ThoughtWorks®

TECHNOLOGY RADAR

Una guía con opiniones sobre las tecnologías de vanguardia

Volume 23

#TWTechRadar

thoughtworks.com/es/radar

Contribuyentes

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

La Junta Asesora de Tecnología (TAB) es un grupo de 21 tecnologistas senior de ThoughtWorks. El TAB se reúne presencialmente dos veces al año y quincenalmente por teléfono. Su función principal es ser un grupo asesor para ThoughtWorks CTO,Rebecca Parsons.

La TAB actúa como un solo individuo que puede analizar temas que influyen en la tecnología y a tecnologistas de ThoughtWorks. Con la pandemia mundial de hoy, se volvió a crear este volumen del Technology Radar a través de un evento virtual.

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Crispim

Cassie Shum

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Lakshminarasimhan Sudarshan

Mike Mason

Neal Ford

Ni Wang

Perla Villarreal

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

Sobre el Radar

Nuestros Thoughtworkers son apasionados por la tecnología. La construimos, investigamos, probamos, liberamos su código fuente, escribimos sobre el y constantemente queremos mejorarlo para todas las personas. Nuestra misión es liderar la excelencia tecnológica y revolucionar las TI. En soporte de esta misión, 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 tecnológicas que 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. El contenido pretende ser un resumen consiso.

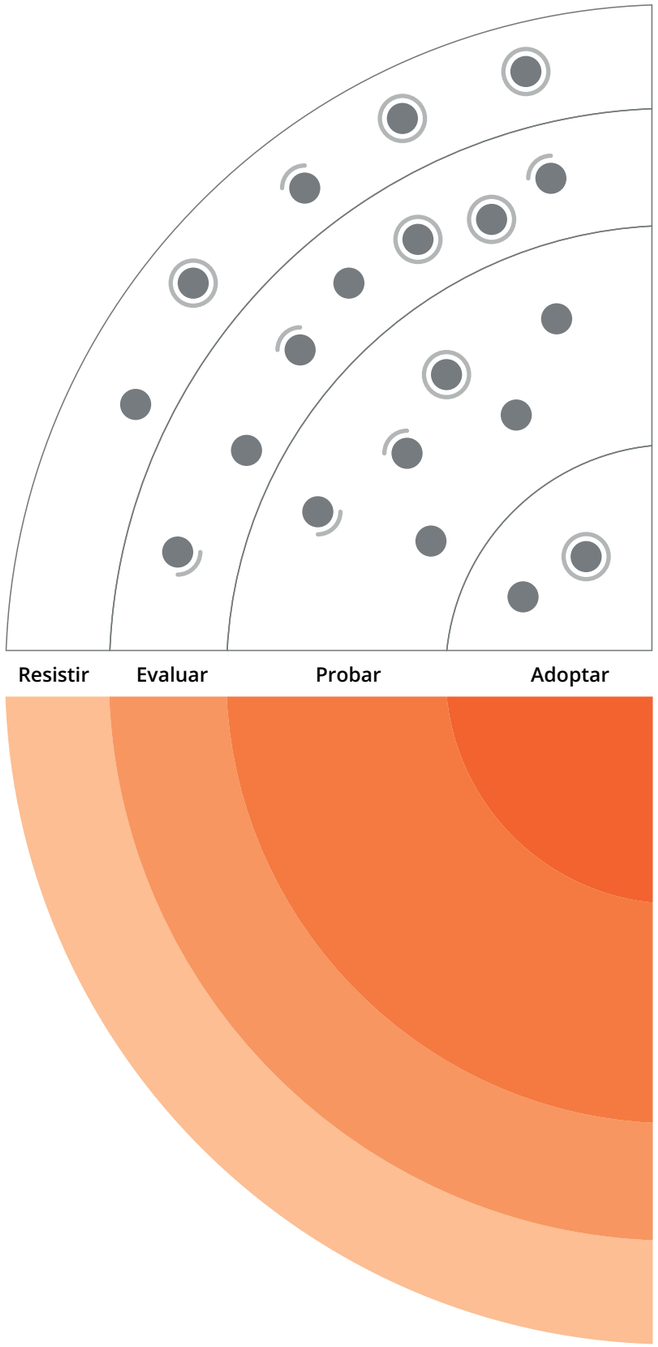
Te alentamos a explorar estas tecnologías. El Radar es gráfico por naturaleza, agrupando items en técnicas, herramientas, plataformas y lenguajes & frameworks. Cuando items del Radar llegan a aparecer en múltiples cuadrantes, elegimos el que parezca más apropiado. Después agrupamos estos items en cuatro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

Probar

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

Evaluar

Vale la pena explorar, con la comprensión de cómo podría afectar a su empresa.

Resistir

Proceder con precuación.

Temas de esta edición

La Grandiosidad de GraphQL

Vemos que ha aumentado la adopción de GraphQL en muchos equipos, además de la existencia de un próspero ecosistema de soporte. GraphQL resuelve algunos problemas comunes que se manifiestan en las arquitecturas distribuidas modernas como los microservicios. Cuando se dividen las cosas en partes pequeñas, a menudo es necesario volver a consolidar la información para resolver los requerimientos del negocio. Esta herramienta ofrece capacidades convenientes que ayudan a resolver este problema cada vez más común. Como todas las abstracciones poderosas, puede ser necesario realizar compromisos y efectuar evaluaciones cuidadosas por parte de los equipos para evitar efectos negativos a largo plazo. Por ejemplo, hemos visto casos donde se provee demasiados detalles de implementación subyacentes a través de una herramienta de agregación, lo que genera una fragilidad innecesaria en la arquitectura. También vimos que facilidades de corto plazo se convirtieron en dolores de cabeza en el largo plazo, como cuando los equipos intentaron utilizar una herramienta de agregación para crear un modelo de datos canónico, universal y centralizado. Alentamos a que los equipos utilicen GraphQL y las herramientas emergentes que lo rodean, pero insistimos en que sean cautelosos en no generalizar tecnologías cuyo enfoque es restringido para resolver demasiados problemas.

Los Desafíos con el Navegador Continúan

El navegador web fue diseñado originalmente para explorar documentos, y hoy en día permite principalmente la ejecución de aplicaciones, con lo que el desfase en la abstracción continúa siendo un reto para las personas desarrolladoras. Para superar los múltiples problemas inherentes a este desfase, los equipos de desarrollo siguen repensando y volviendo a desafiar a los métodos establecidos para ejecutar pruebas en el navegador, gestionar el estado y, en general, construir aplicaciones ricas y rápidas. Hemos visto varias de estas tendencias en el Radar. Primero, movimos a Redux a “Adoptar” en 2017 y lo presentamos como la solución por defecto para gestionar el estado en aplicaciones React, y ahora vemos que los equipos de desarrollo buscan algo distinto (Recoil) o retrasan la decisión de adoptar una librería de gestión del estado. Segundo, Svelte ha estado ganando más interés y desafía uno de los conceptos establecidos y aplicados por los *frameworks* populares como React y Vue.js: el DOM virtual. Tercero, seguimos viendo nuevas herramientas para ejecutar pruebas en el navegador: Playwright es otro nuevo intento de mejorar las pruebas de interfaz de usuario mientras que Mock Service Worker es un nuevo método para desacoplar las pruebas de las interacciones con el back-end. Cuarto, continuamos observando el reto de balancear la productividad de las personas desarrolladoras con el rendimiento: con los polyfills adaptados al navegador se intenta cambiar la escala en este compromiso.

Visualizar Todo

Esta edición del Radar presenta varios *blips* a través de todo el escenario tecnológico con una cosa en común: visualización. Encontrarás blips sobre infraestructura, ciencia de datos, recursos en la nube y un conjunto de innovadoras herramientas de visualización, incluyendo varias formas efectivas para ver abstracciones complicadas. También encontrarás discusiones sobre herramientas interactivas para la visualización de datos y tableros como Dash, Bokeh y Streamlit, así como tambien herramientas para la visualización de la infraestructura, como Kiali, utilizada para visualizar mallas de servicio en arquitecturas de microservicios. A medida que los ecosistemas de desarrollo se hacen más complejos una imagen generalmente ayuda a dominar la inevitable sobrecarga cognitiva.

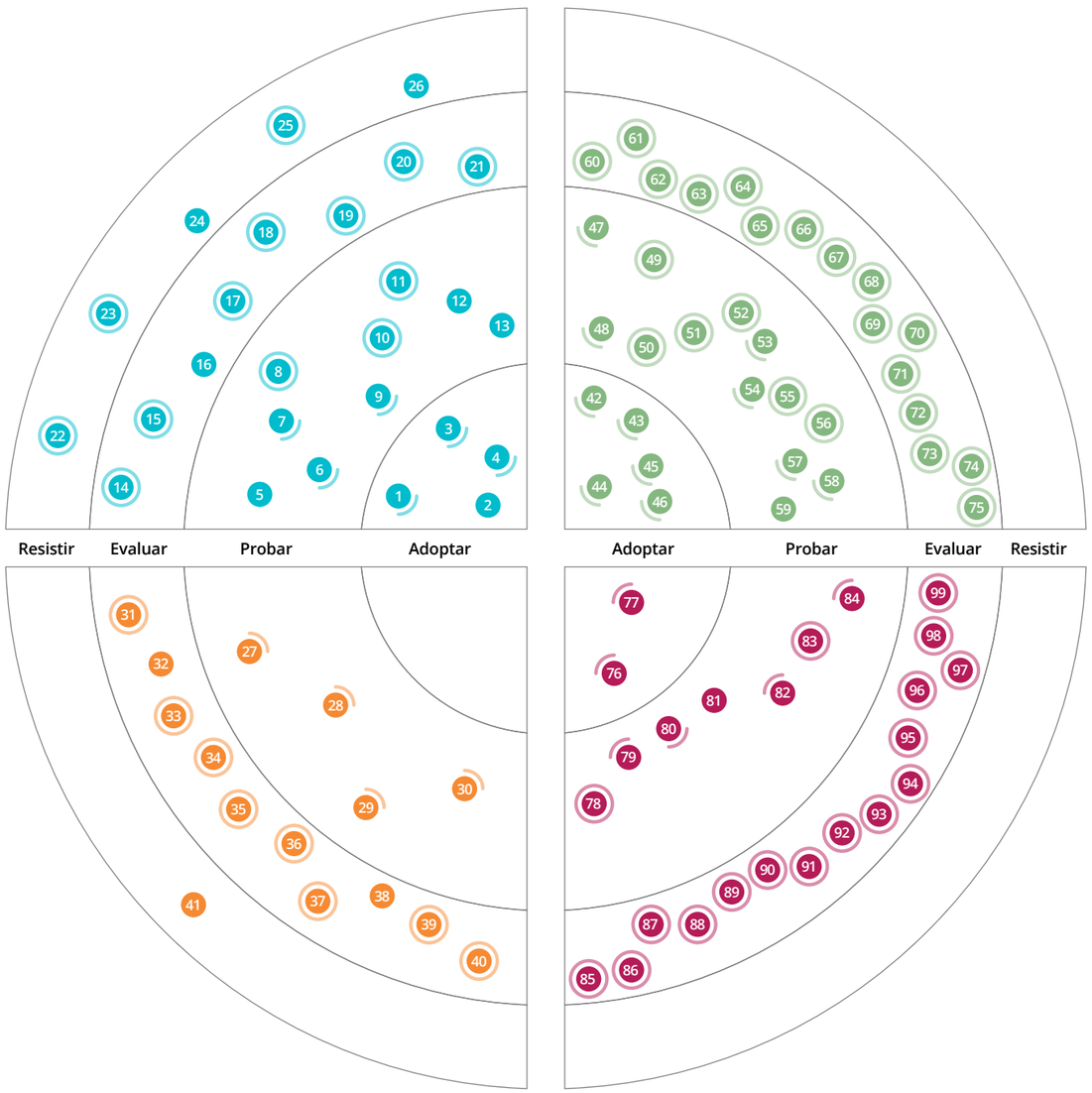
Adolescencia de la Infraestructura como Código

Administrar la infraestructura como código se ha hecho más común a medida que las organizaciones ven los beneficios de automatizar su infraestructura creando, por consiguiente, un ciclo de retroalimentación de innovación-adopción para los creadores de herramientas y *frameworks*. Herramientas como el CDK de AWS y Pulumi, entre otras, ofrecen capacidades más extensas que sus predecesoras, mejorando a tal punto que creemos que la práctica de la infraestructura como código ha llegado a su adolescencia, con sus consecuencias tanto positivas como negativas. Estamos gratamente sorprendidos del número de blips en todos los cuadrantes que reflejan positivamente la mejora en madurez de este ecosistema. Sin embargo, también discutimos los desafíos en cuanto a la falta de patrones maduros y los problemas que muchas compañías enfrentan cuando tratan de encontrar el mejor uso de esta práctica, todos los cuales evidencian el avance en el camino hacia la madurez. Esperamos que la comunidad de infraestructura siga aprendiendo lecciones del diseño de software, especialmente en términos de la creación de una infraestructura poco acoplada que sea capaz de ser desplegada.

Democratizando la Programación

Varias de nuestras discusiones giran en torno a las herramientas y técnicas que promueven la democratización de la programación: habilitar a cualquier persona para ejecutar tareas que anteriormente solo las personas con conocimientos de programación podían realizar. Por ejemplo, soluciones como IFTTT y Zapier han sido muy populares en este espacio por mucho tiempo. Hemos observado un incremento en el uso de herramientas como Amazon Honeycode, una herramienta que permite crear aplicaciones de negocio simples sin saber programar. A pesar de que estas herramientas proveen un ambiente de programación de propósito específico, los desafíos surgen cuando se mueven estas soluciones a ambientes de producción de gran escala. Las personas desarrolladoras y las expertas con las hojas de cálculo, por mucho tiempo han buscado la forma de encontrar un punto medio entre los ambientes de programación específicos a un dominio y los tradicionales. El advenimiento de nuevas y modernas herramientas renueva la discusión a través de dominios más amplios, donde se presentan muchos de los mismos pros y contras.

El Radar



Técnicas

Adoptar
1. Funciones de aptitud de desfase de dependencias
2. Coste de ejecución como función de aptitud de arquitectura
3. Políticas de seguridad como código
4. Plantillas de servicio a la medida

Probar
5. Entrega continua para aprendizaje automático (CD4ML)
6. Malla de datos
7. Definición declarativa para pipelines de datos
8. Diagramas como código
9. Imágenes Docker sin distribución
10. Intercepción de eventos
11. Ejecución paralela con conciliación
12. Usar procesos y enfoques “remotos nativos”
13. Arquitectura de Confianza Cero

Evaluar
14. Plataformas delimitadas de poco código
15. Pollyfills adaptados al navegador
16. Identidad descentralizada
17. Servicios en la nube gestionados por Kubernetes
18. Modelo Abierto de Aplicaciones (OAM)
19. Enclaves seguros
20. Experimentación de avance y retroceso
21. Credenciales verificables

Resistir
22. Apollo Federation
23. ESBs disfrazados de API Gateways
24. Agregación de registros para análisis de negocios
25. Anarquía de los “micro-frontends”
26. Llevar “notebooks” a producción

Plataformas

Adoptar

Probar
27. Azure DevOps
28. Debezium
29. Honeycomb
30. JupyterLab

Evaluar
31. Amundsen
32. Kit de Desarrollo en la Nube
33. Backstage
34. Dremio
35. DuckDB
36. K3s
37. Materialize
38. Pulumi
39. Tekton
40. Stack de Trust over IP

Resistir
41. Node en exceso

Herramientas

Adoptar
42. Airflow
43. Bitrise
44. Dependabot
45. Helm
46. Trivy

Probar
47. Bokeh
48. Concourse
49. Dash
50. jscodeshift
51. Kustomize
52. MLflow
53. Pitest
54. Sentry
55. ShellCheck
56. Stryker
57. Terragrunt
58. tfsec
59. Yarn

Evaluar
60. CML
61. Eleventy
62. Flagger
63. gossm
64. Great Expectations
65. k6
66. Katran
67. Kiali
68. LGTM
69. Litmus
70. Opacus
71. OSS Index
72. Playwright
73. pnpm
74. Sensei
75. Zola

Resistir

Lenguajes & Frameworks

Adoptar
76. Arrow
77. jest-when

Probar
78. Fastify
79. Immer
80. Redux
81. Rust
82. single-spa
83. Strikt
84. XState

Evaluar
85. Babylon.js
86. Blazor
87. Flutter Driver
88. Sentinel
89. Hermes
90. io-ts
91. Kedro
92. LitElement
93. Mock Service Worker
94. Recoil
95. Snorkel
96. Streamlit
97. Svelte
98. SWR
99. Testing Library

Resistir

TECHNOLOGY RADAR

Técnicas

Técnicas

Funciones de aptitud de desfase de dependencias

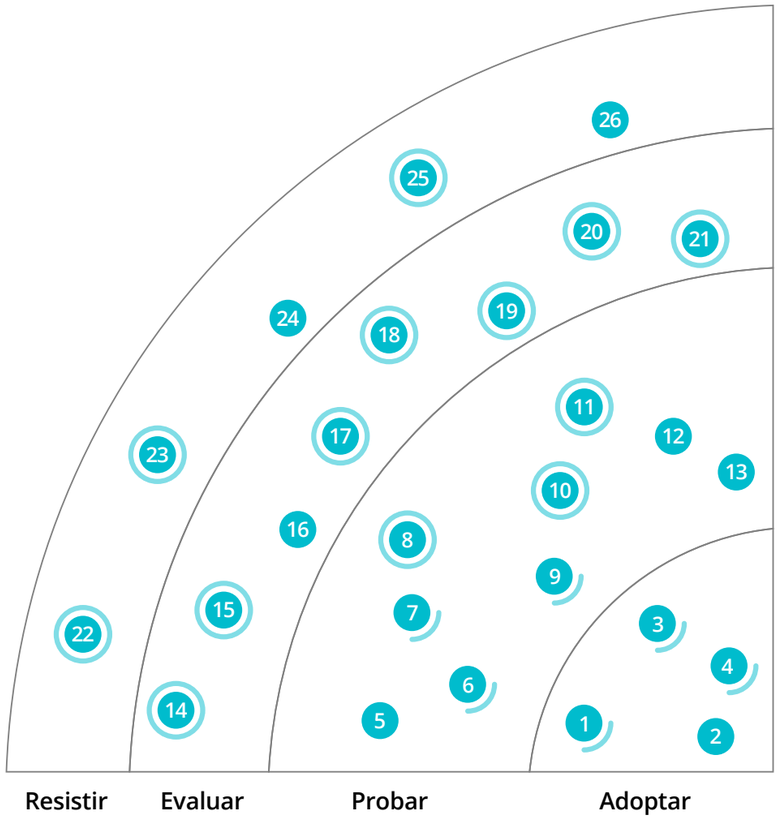
Adoptar

Las funciones de aptitud introducidas por la arquitectura evolutiva, tomadas de la computación evolutiva, son funciones ejecutables que informan si las aplicaciones y la arquitectura se están alejando objetivamente de sus características deseadas. Básicamente, son pruebas que se pueden incorporar en los *pipelines*. Una de las principales características de una aplicación es la frescura de sus dependencias hacia bibliotecas, APIs o componentes del entorno que una función de aptitud de desfase de dependencias monitorea para marcar a aquellas que están obsoletas y que requieren de actualización. Con el incremento en cantidad y madurez de las herramientas que detectan desfases en las dependencias, como Dependabot o Snyk, podemos incorporar fácilmente funciones de aptitud de desfase de dependencias en el proceso de lanzamiento del software y así tomar las medidas más oportunas para mantener actualizadas las dependencias de nuestras aplicaciones.

Coste de ejecución como función de aptitud de arquitectura

Adoptar

Automatizar la estimación, seguimiento y proyección del coste de ejecución de una infraestructura en la nube es necesario para las organizaciones de hoy. Los modelos inteligentes de precios de los proveedores de la nube, combinados con la proliferación de los parámetros de precios y la naturaleza dinámica de la arquitectura de hoy pueden llevar a costos de ejecución sorprendentemente altos. Por ejemplo, los precios de serverless basados en llamadas API, de soluciones de streaming de eventos enfocadas en el tráfico o de procesamientos de grupos de datos basados en tareas corridas, tienen una naturaleza dinámica que cambia a lo largo del tiempo a medida que la arquitectura evoluciona. Cuando nuestros equipos manejan infraestructura en la nube, implementar el coste de ejecución como función de la aptitud de arquitectura es una de sus primeras tareas. Esto quiere decir que nuestros equipos pueden observar el costo de ejecutar los servicios en comparación al valor entregado; cuando ven desviaciones respecto a lo que se espera o es aceptable, discutirán si es momento de evolucionar la arquitectura. La observación y cálculo del coste de ejecución se implementa como una función automatizada.



Políticas de seguridad como código

Adoptar

A medida que el terreno de la tecnología se vuelve más complejo, áreas como la seguridad requieren de mayor automatización y prácticas de ingeniería. Cuando construimos sistemas, debemos

Adoptar
1. Funciones de aptitud de desfase de dependencias
2. Coste de ejecución como función de aptitud de arquitectura
3. Políticas de seguridad como código
4. Plantillas de servicio a la medida

Probar
5. Entrega continua para aprendizaje automático (CD4ML)
6. Malla de datos
7. Definición declarativa para pipelines de datos
8. Diagramas como código
9. Imágenes Docker sin distribución
10. Intercepción de eventos
11. Ejecución paralela con conciliación
12. Usar procesos y enfoques “remotos nativos”
13. Arquitectura de Confianza Cero

Evaluar
14. Plataformas delimitadas de poco código
15. Pollyfills adaptados al navegador
16. Identidad descentralizada
17. Servicios en la nube gestionados por Kubernetes
18. Modelo Abierto de Aplicaciones (OAM)
19. Enclaves seguros
20. Experimentación de avance y retroceso
21. Credenciales verificables

Resistir
22. Apollo Federation
23. ESBs disfrazados de API Gateways
24. Agregación de registros para análisis de negocios
25. Anarquía de los “micro-frontends”
26. Llevar “notebooks” a producción

Técnicas

La malla de datos marca el inicio de un cambio en los paradigmas arquitectónicos y organizacionales sobre cómo se gestionan los datos analíticos masivos. Sin embargo, hay una gran brecha en las herramientas de código abierto para acelerar la implementación de mallas de datos.

(Malla de datos)

considerar a las políticas de seguridad, aquellas reglas y procedimientos que protegen a los sistemas de peligros e interrupciones. Por ejemplo, las políticas de control de acceso definen e imponen quién puede acceder a cada servicio y recurso y bajo qué circunstancias; en contraste, las políticas de seguridad de redes pueden limitar dinámicamente el nivel de tráfico hacia un servicio en particular.

Varios de nuestros equipos han tenido buenos resultados al tratar a las políticas de seguridad como código. Cuando decimos como código, no nos limitamos a escribir estas políticas en un archivo, si no que también aplicamos técnicas como mantener ese código versionado, introducir validaciones automatizadas en el pipeline, despliegues automatizados a ambientes así como observar y monitorear su funcionamiento. Basados en nuestra experiencia y en la madurez de las herramientas y plataformas existentes, como Open Policy Agent e Istio, que proveen mecanismos para definir e imponer políticas flexibles que soportan la práctica de políticas de seguridad como código, recomendamos mucho utilizar esta técnica en tu entorno.

Plantillas de servicio a la medida

Adoptar

Desde la última vez que hablamos de las plantillas de servicio personalizadas , hemos visto una mayor adopción de este patrón para ayudar a allanar el camino de las organizaciones que se mueven a microservicios. Con los constantes avances en herramientas de observabilidad, orquestación de contenedores y sidecars de mallas de servicios, una plantilla proporciona definiciones predeterminadas para simplificar la creación de nuevos servicios, eliminando gran parte de la configuración necesaria para que el servicio funcione bien con la infraestructura circundante.

Hemos tenido éxito aplicando los principios de gestión de productos a las plantillas, tratando a las personas desarrolladoras internas como clientes y facilitándoles llevar su código a producción y realizar su operación con apropiados niveles de observabilidad. Esto tiene el beneficio añadido de actuar como un mecanismo ligero de gobernanza para centralizar las decisiones técnicas por defecto.

Entrega continua para aprendizaje automático (CD4ML)

Probar

Hace aproximadamente una década introdujimos la entrega continua, nuestra manera por defecto de entregar soluciones de software. Las soluciones actuales incluyen, cada vez más, modelos de aprendizaje automático y no los consideramos una excepción para la adopción de prácticas de entrega continua. Lo llamamos entrega continua para aprendizaje automático (CD4ML). A pesar de que los principios de la entrega continua siguen siendo los mismos, las herramientas y prácticas para implementar el proceso de principio a fin de entrenamiento, pruebas, despliegue y monitoreo de modelos requiere de algunas modificaciones. Por ejemplo: el control de versiones no solo incluye el código si no también los datos, los modelos y sus parámetros; la pirámide de pruebas se extiende para incluir análisis y validación de sesgos, de equidad, de datos y características; el proceso de despliegue debe considerar como promover y evaluar el rendimiento de los nuevos modelos contra los actuales modelos campeones. Mientras la industria está celebrando el nuevo término de moda, MLOps, creemos que CD4ML es la aproximación holística para implementar un proceso integral para entregar de forma fiable y mejorar continuamente los modelos de aprendizaje automático, desde la ideación hasta producción.

Malla de datos

Probar

La malla de datos marca el inicio de un cambio en los paradigmas arquitectónicos y organizacionales sobre cómo se gestionan los datos analíticos masivos. El paradigma se fundamenta en cuatro principios: (1) descentralización orientada al dominio de la propiedad de los datos y de su arquitectura; (2) datos orientados al dominio servidos como un producto; (3) auto servicio de infraestructura de datos como plataforma, para impulsar la autonomía de los equipos orientados al dominio; y (4) gobernanza federada para impulsar ecosistemas y la interoperabilidad. Si bien los principios son intuitivos y pretenden abordar muchos de los desafíos ya conocidos de la gestión centralizada de datos analiticos, estos trascienden las tecnologías actuales para datos analiticos. Luego de construir mallas de datos en muchos clientes con las herramientas existentes hemos aprendido dos cosas: (a) hay una gran brecha en las herramientas de código abierto o comerciales para acelerar la implementación de mallas de datos (por ejemplo, la implementación de un modelo de acceso universal a datos políglota basados en tiempo, que actualmente construimos a la medida para nuestros clientes) y (b) a pesar de la brecha, es factible usar tecnologías existentes como elementos básicos.

Naturalmente, la idoneidad tecnológica es un componente importante en la implementación de una estrategia de datos de una organización basada en una malla de datos. Sin embargo, el éxito requiere de una reorganización estructural para separar al equipo de la plataforma de datos, crear el rol del *product owner* de datos para cada dominio e introducir las estructuras necesarias de incentivos para que los dominios se apropien y compartan los datos analíticos como productos.

Definición declarativa para pipelines de datos

Probar

Muchos pipelines de datos se definen en un script largo, más o menos imperativo, escrito en Python o Scala. El script contiene tanto la lógica con los pasos individuales así como el código que enlaza los pasos entre sí. Cuando encontramos situaciones similares en pruebas de Selenium, los equipos de desarrollo descubrieron el patrón Objeto de Página (Page Object), y luego, varios frameworks para el desarrollo dirigido por comportamientos (BDD) implementaron una división entre las definiciones de pasos y su composición. Algunos equipos están aún experimentando para traer el mismo razonamiento a la ingeniería de datos. Una definición declarativa para *pipelines* de datos separada, escrita tal vez en YAML, contiene únicamente la declaración y la secuencia de pasos. En esta se declaran los conjuntos de datos de entrada y salida pero se recurre a los scripts si y cuando se requiere de lógica más compleja. A La Mode es una herramienta relativamente nueva que toma un enfoque de DSL para realizar la definición de *pipelines*, pero airflow-declarative, una herramienta que convierte directamente los grafos acíclicos en YAML en tareas programadas para Airflow, parece tener mayor impulso en este espacio.

Diagramas como código

Probar

Estamos viendo más y más herramientas que te permiten crear diagramas como código para arquitectura de software y otros usos. Existen beneficios al utilizar estas herramientas en lugar de las alternativas más pesadas, como poder incluirlos fácilmente en el control de versiones y la habilidad de generar los DSLs desde varios orígenes. Entre las herramientas que nos gustan están Diagrams, Structurizr DSL, AsciiDoctor Diagram además de herramientas bien establecidas como WebSequenceDiagrams, PlantUML y el venerable Graphviz. También, como es bastante sencillo generar SVG hoy en día, no hay que descartar inmediatamente la opción de escribir una herramienta propia. Por ejemplo, uno de nuestros autores hizo un pequeño programa en Ruby para generar SVGs rápidamente.

Imágenes Docker sin distribución

Probar

Cuando se construyen imágenes de Docker para las aplicaciones, usualmente nos preocupan dos cosas: la seguridad y su tamaño. Tradicionalmente hemos usado herramientas de escaneo de seguridad en contenedores para detectar y corregir vulnerabilidades y riesgos comunes y distribuciones pequeñas como Alpine Linux para resolver el tema del tamaño de la imagen y del rendimiento de la distribución. Hemos ganado más experiencia con imágenes Docker sin distribución y estamos listos para recomendar esta estrategia como otra importante medida de seguridad para aplicaciones contenerizadas. Este tipo de imágenes para Docker reducen el tamaño y las dependencias suprimiendo las distribuciones de sistemas operativos completos. Esta técnica reduce el ruido en los escaneos de seguridad y la superficie de ataque a la aplicación: hay menos vulnerabilidades que hay que corregir y, como extra, estas imágenes son más eficientes. Google ha publicado un conjunto de imágenes de contenedor sin distribución para diferentes lenguajes. Se pueden crear imágenes para aplicación sin distribución utilizando la herramienta de construcciónl Bazel de Google o utilizando simplemente Dockerfiles multistage. Hay que notar que los contenedores sin distribución no tienen un shell para depurar. Sin embargo es fácil encontrar en línea versiones de contenedores sin distribución habilitadas para la depuración que incluyen el shell BusyBox. Google ha sido pionero en esta técnica y, según nuestra experiencia, sigue estando confinada a imágenes generadas por Google. Esperamos que esta técnica se extienda por fuera de este ecosistema.

Intercepción de eventos

Probar

A medida que más compañías están migrando sus sistemas legados, sentimos que vale la pena destacar una alternativa a la captura de cambios en los datos (CDC) como el mecanismo para obtener datos de estos sistemas. Martin Fowler describió la intercepción de eventos en 2004. En términos modernos, implica bifurcar las peticiones que ingresan a un sistema para que sea posible construir un reemplazo gradualmente. A menudo, esto es resultado de copiar eventos o mensajes, pero bifurcar peticiones HTTP es también válido. Algunos ejemplos son bifurcar eventos desde sistemas de puntos de venta antes de que se escriban en el sistema central y bifurcar transacciones de pago antes de que se escriban en los sistemas centrales de un banco. Ambos conducen al reemplazo gradual de las partes de los sistemas legados. Percibimos que la técnica de obtención de cambios del estado en su origen ha sido subestimada (ya que se ha preferido volver a crear estos eventos usando CDC, luego que los originales han sido procesados), por lo que la destacamos en esta edición del Radar.

Técnicas

Estamos viendo una proliferación de herramientas que permiten crear diagramas de arquitectura de software y otros diagramas como código. Los beneficios incluyen un fácil control de versiones y la habilidad de generar los DSLs desde varios orígenes

(Diagramas como código)

La intercepción de eventos es una alternativa que vale la pena para capturar cambios en los datos cuando estas migrando tus sistemas legados. Se ha pasado por alto el hecho de obtener los cambios de estado en la fuente, en lugar de tratar de recrearlos después de procesarlos utilizando los CDC.

(Intercepción de eventos)

Técnicas

Las plataformas de poco o ningún código pueden resolver problemas muy específicos en dominios muy limitados. Sin embargo, seguimos siendo escépticos sobre su amplia aplicabilidad.

(Plataformas delimitadas de poco código)

Ejecución paralela con conciliación

Probar

Reemplazar código legado en escala siempre es un esfuerzo complicado, el cual se beneficia a menudo de realizar una ejecución paralela con conciliación. En la práctica, esta técnica se fundamenta en ejecutar el mismo flujo en producción a través del código antiguo y nuevo, retornar la respuesta del código legado pero comparar los resultados para ganar mayor confianza en la nueva implementación. A pesar de ser una técnica antigua, en años recientes hemos visto implementaciones más robustas, basadas en prácticas de entrega continua como canary releases y feature toggles, y que se amplían añadiendo una capa extra de experimentación y análisis de datos para comparar los resultados en vivo. Incluso, hemos usado este enfoque para comparar resultados sobre aspectos no funcionales como el tiempo de respuesta. Aunque hemos utilizado esta técnica varias veces con herramientas hechas a la medida, sin duda le debemos un reconocimiento a la herramienta Scientist, de GitHub, que fue utilizada para actualizar una pieza crítica de su aplicación y que ahora ha sido portada a múltiples lenguajes de programación.

Usar procesos y enfoques “remotos nativos”

Probar

Mientras la pandemia se extiende, parece que, por el momento, los equipos con una alta distribución serán parte de la “nueva normalidad”. En los últimos 6 meses hemos aprendido mucho acerca del trabajo remoto efectivo. En el lado positivo, las herramientas de visualización de colaboración y gestión de trabajo han simplificado más que nunca el trabajo remoto en conjunto entre colegas. Las personas desarrolladoras, por ejemplo, pueden apoyarse en Visual Studio Live Share y en GitHub Codespaces para facilitar el trabajo en equipo e incrementar la productividad. La mayor desventaja del trabajo remoto podría ser el agotamiento: demasiadas personas tienen video llamadas programadas consecutivamente durante todo el día, y esto ya está generando consecuencias. Mientras las herramientas visuales que tenemos en línea mejoran la posibilidad de colaborar, también es posible crear grandes y complejos diagramas que terminan siendo muy difíciles de usar, y los aspectos de seguridad por la proliferación de herramientas también debe ser administrado con mucho cuidado. Nuestro consejo es acordarse de dar un paso hacia atrás, hablar con los equipos, evaluar qué funciona y qué no y cambiar los procesos y herramientas según sea necesario.

Arquitectura de Confianza Cero

Probar

Mientras que la infraestructura computacional y de datos sigue cambiando en las empresas (de aplicaciones monolíticas a microservicios, de lagos de datos centralizados a mallas de datos, de alojamiento en servidores propios a usar las nubes de varios proveedores, con una proliferación creciente de dispositivos conectados), el enfoque para asegurar los activos empresariales sigue sin mayores cambios, con gran dependencia y confianza en el perímetro de la red: las organizaciones siguen haciendo grandes inversiones para asegurar sus activos fortaleciendo las perímetros virtuales de sus empresas, utilizando enlaces privados y configuraciones de cortafuegos, y reemplazando procesos de seguridad estáticos y engorrosos que ya no aplican en la realidad de hoy. Esta tendencia nos obliga a destacar nuevamente la arquitectura de confianza cero (ZTA). ZTA representa un cambio para los paradigmas de arquitectura y en las estrategias de seguridad. Se basa en el supuesto de que el perímetro de una red ya no representa un límite seguro y no se debe otorgar confianza implícita a los usuarios o servicios basándose únicamente en su ubicación física o de red. La cantidad de recursos, herramientas y plataformas disponibles para implementar aspectos de ZTA sigue creciendo e incluye: hacer cumplir políticas como código basadas en los principios de menor privilegio y de la mayor granularidad posible además del monitoreo continuo y la mitigación automatizada de amenazas; usar mallas de servicios para hacer cumplir los controles de seguridad de aplicación a servicio y de servicio a servicio; implementar certificación de los archivos binarios para verificar su origen; e incluir enclaves seguros además del cifrado tradicional para hacer cumplir los tres pilares de la seguridad de los datos: en tránsito, en reposo y en la memoria. Para obtener más información de este tema, consulta la publicación sobre ZTA del NIST y el artículo de Google sobre BeyondProd.

Plataformas delimitadas de poco código

Evaluar

Una de las decisiones más complejas a las que se enfrentan las compañías en este momento es la adopción de plataformas de poco o ningún código, es decir, plataformas que resuelven problemas muy específicos en dominios bastante limitados. Muchos proveedores están presionando agresivamente hacia este espacio.
Los problemas que vemos con estas plataformas se relacionan típicamente con la imposibilidad de aplicar buenas prácticas de ingeniería como el versionamiento. Además, realizar pruebas es generalmente muy difícil. Sin embargo, hemos notados la existencia de algunos nuevos e interesantes participantes en el mercado, como Amazon Honeycode que facilita la creación de aplicaciones simples de gestión de tareas o eventos y Parabola para flujos de trabajo en la nube similares a los de IFTTT, por lo

que estamos incluyendo a las plataformas delimitadas de poco código en esta edición del Radar. No obstante, seguimos siendo profundamente escépticos acerca de su mayor aplicabilidad, ya que estas herramientas tienen el poder de escapar de sus límites y enredarlo todo, como lo hace la maleza. Es por eso que seguimos aconsejando tener mucho cuidado en su adopción.

Pollyfills adaptados al navegador

Evaluar

Los polyfills son muy útiles para ayudar a la evolución de la web ya que proveen implementaciones alternativas para funcionalidades modernas en navegadores que aún no las soportan. Sin embargo, muy a menudo, las aplicaciones web envían polyfills a navegadores que no los necesitan, provocando descargas innecesarias y gasto de recursos al procesarlos. Esta situación se ha hecho más notoria ahora que quedan pocos motores de renderizado y la mayoría de los polyfills apuntan solo a uno de ellos: el motor Trident de IE11. Además, la cuota de mercado de IE11 está decreciendo a medida que se acerca el fin de su soporte, en menos de un año. Por lo tanto, sugerimos que se utilicen polyfills adaptados al navegador , que entreguen solo los *polyfills* que en realidad se necesitan para cada navegador. Esta técnica se podría implementar como un servicio con Polyfill.io.

Identidad descentralizada

Evaluar

En 2016, Christopher Allen, uno de los contribuyentes principales a SSL/TLS, nos inspiró con una introducción de 10 principios que apuntaban a una nueva forma de identidad digital así como una forma de llegar a ella: el camino hacia la auto-identidad soberana. La auto-identidad soberana, también conocida como la identidad descentralizada , es una “identidad portable y para toda la vida de una persona, organización o cosa que no depende de una autoridad centralizada y que nunca se puede confiscar”, de acuerdo al estandar Trust over IP. La adopción e implementación de identidades descentralizadas está ganando impulso y convirtiéndose en algo asequible. Vemos que se está adoptando en aplicaciones médicas, infraestructura sanitaria pública e identidad empresarial legal que respetan la privacidad. Si quieres comenzar pronto con la identidad descentralizada, puedes evaluar proyectos de código abierto como Sovrin Network, Hyperledger Aries e Indy OSS, así como los estándares de identificadores descentralizados y de credenciales verificables. Estamos muy pendientes de lo que sucede en este campo mientras ayudamos a nuestros clientes con su posicionamiento estratégico en la nueva era de confianza digital..

Servicios en la nube gestionados por Kubernetes

Evaluar

Los proveedores de la nube han comenzado lentamente a dar soporte a APIs del estilo de Kubernetes, mediante definiciones de recursos personalizados (CRDs) para gestionar sus servicios en la nube. En la mayoría de los casos, estos servicios en la nube son una parte central de la infraestructura y hemos visto a equipos utilizar herramientas como Terraform o Pulumi para provisionarlos. Con estos nuevos CRDs (ACK for AWS, Azure Service Operator para Azure y Config Connectors for GCP) puedes utilizar Kubernetes para provisionar y gestionar estos servicios en la nube. Una de las ventajas de estos servicios en la nube gestionados por Kubernetes es que puedes utilizar el mismo nivel de control de Kubernetes para garantizar el estado declarativo tanto de tu aplicación como de su infraestructura. El inconveniente es que acopla estrechamente el clúster de Kubernetes con la infraestructura, por lo que lo estamos evaluando cuidadosamente y pensamos que deberías hacerlo igual.

Modelo Abierto de Aplicaciones (OAM)

Evaluar

Hemos hablado mucho sobre los beneficios de crear equipos de producto de ingeniería de plataformas para dar soporte a los demás equipos de producto, pero en realidad es difícil hacerlo. Parece que la industria aún está buscando la abstracción correcta en el mundo de la infraestructura como código. Aunque herramientas como Terraform y Helm son pasos en la dirección correcta, el enfoque sigue estando en la gestión de la infraestructura en lugar del desarrollo de aplicaciones. También hay cambios hacia el concepto de infraestructura como software con la aparición de nuevas herramientas como Pulumi y CDK. El Modelo Abierto de Aplicaciones (OAM) es un intento de traer alguna estandarización a este espacio. Utilizando las abstracciones de componentes, configuraciones de aplicaciones, ámbitos o contextos (*scopes*) y aspectos o atributos (*traits*), las personas desarrolladoras pueden describir sus aplicaciones independientemente de la plataforma, mientras que las personas implementadoras de plataforma definen su plataforma en términos de cargas de trabajo, aspectos o atributos (*traits*) y ámbitos o contextos (*scopes*). Queda por ver si el OAM será ampliamente adoptado, pero recomendamos estar atento a esta idea interesante y necesaria.

Enclaves seguros

Evaluar

Los enclaves seguros , también conocidos como entornos de ejecución confiable (*trusted execution environments, TEE*), hacen referencia a una técnica que aísla un entorno (procesador,

Técnicas

La mayoría de las credenciales digitales actuales son registros de datos simples en sistemas de información que son fáciles de modificar y que a menudo, exponen información innecesaria En los últimos años, hemos visto el crecimiento de las credenciales verificables para resolver este problema.

(Credenciales verificables)

Técnicas

Independientemente de como se llamen, poner la lógica de negocio en una herramienta centralizada crea acoplamiento en la arquitectura, reduce la transparencia e incrementa la dependencia en un proveedor sin añadir ventajas.

(ESBs disfrazados de API Gateways)

memoria y almacenamiento) con un mayor nivel de seguridad y limita el intercambio de información con otros contextos no fiables. Por ejemplo, un enclave seguro a nivel de hardware y sistema operativo podría crear y almacenar claves privadas y realizar operaciones con ellas como el cifrado de información o verificación de firmas sin permitir que las claves privadas salgan del enclave seguro o que sean cargadas en memoria de algún proceso no fiable. Un enclave seguro provee un conjunto limitado de instrucciones para ejecutar operaciones confiables, de manera aislada a un contexto de aplicación no fiable.

La técnica ha sido ampliamente soportada por muchos proveedores de hardware y proveedores de sistemas operativos (incluído Apple), y las personas desarrolladoras lo han usado en la Internet de las Cosas y para edge applications. Sin embargo, sólo recientemente ha ganado atención tanto en aplicaciones empresariales como en aplicaciones en la nube. Los proveedores de la nube han empezado a introducir funcionalidades de computación confidencial en la forma de enclaves seguros basados en hardware: La infraestructura de computación confidencial de Azure promete máquinas virtuales con entornos de ejecución confiables y accesibles mediante el Open Enclave SDK que es una biblioteca de código abierto para ejecutar operaciones confiables. De igual manera, GCP Confidential VMs and Compute Engine, que todavía está en beta, permite el uso de máquinas virtuales con cifrado de datos en memoria y AWS Nitro Enclaves está siguiendo esa tendencia en su siguiente entrega. Con la introducción de enclaves seguros basados en la nube y la computación confidencial, podemos añadir un tercer pilar a la protección de datos: en reposo, en transmisión y ahora en la memoria.

Si bien nos encontramos todavía en los primeros días de los enclaves seguros para aplicaciones empresariales, animamos a considerar esta técnica teniendo en cuenta las posibles vulnerabilidadades que pueden comprometer el enclave seguro de los proveedores de hardware usados.

Experimentación de avance y retroceso

Evaluar

Los experimentos controlados usando pruebas A/B son una gran manera de contextualizar decisiones alrededor del desarrollo de productos. Pero no funcionan muy bien cuando no podemos establecer independencia entre los dos grupos involucrados en la prueba A/B, por ejemplo, al añadir alguien al grupo “A” impacta al grupo “B” y viceversa. Una técnica para resolver este tipo de problema es la experimentación de avance y retroceso. El concepto central en esto es cambiar entre los modos “A” y “B” continuamente en ciertas regiones en periodos de tiempos alternativos en vez de correr ambos modos durante el mismo periodo de tiempo. Entonces comparamos la experiencia de la clientela y otras métricas clave entre los dos periodos de tiempo. Hemos probado esta técnica con buenos resultados en algunos de nuestros proyectos y creemos que es una buena herramienta para incluir en nuestra caja de herramientas de experimentación.

Credenciales verificables

Evaluar

Las credenciales están en todas partes en nuestras vidas e incluyen pasaportes, licencias de conducir y certificados académicos. Sin embargo, la mayoría de las credenciales digitales actuales son registros de datos simples en sistemas de información que son fáciles de modificar y falsificar, y que a menudo, exponen información innecesaria. En los últimos años, hemos visto cómo la madurez continuada de Credenciales verificables resuelve este problema. El estándar de la W3C lo define de una manera que es criptográficamente segura, respetuosa de la privacidad y verificable por máquina. El modelo coloca a los titulares de las credenciales en el centro, que es similar a nuestra experiencia cuando usamos credenciales físicas: los usuarios pueden poner sus credenciales verificables en sus propias billeteras digitales y mostrarlas a cualquier persona en cualquier momento sin el permiso del emisor de las credenciales. Este enfoque descentralizado también permite a los usuarios administrar mejor su propia información y revelar selectivamente cierta información y mejora en gran medida la protección de la privacidad de los datos. Por ejemplo, debido a la tecnología de prueba de conocimiento cero, se puede construir una credencial verificable para demostrar que uno es un adulto sin revelar su fecha de nacimiento. La comunidad ha desarrollado muchos casos de uso en torno a credenciales verificables. Hemos implementado nuestra propia certificación de salud COVID con referencia a la Iniciativa de Credenciales para COVID-19 (CCI). Aunque las credenciales verificables no dependen de la tecnología blockchain o identidad descentralizada, esta técnica a menudo funciona con DID en la práctica y utiliza blockchain como registro de datos comprobables. Muchos *frameworks* de identidad descentralizada también están integrados con credenciales verificables.

Apollo Federation

Resistir

Cuando hablamos por primera vez de GraphQL en el Radar, advertimos que su mal uso podria llevar a situaciones que tienen mas desventajas que beneficios en el largo plazo. No obstante, hemos visto un aumento en el interés en GraphQL entre nuestros equipos por su habilidad de agregar información provista por diferentes recursos. Esta vez queremos advertir sobre el uso de Apollo Federation y su soporte sólido para un grafo de datos único y unificado para tu compañía. Si bien, a primera vista, la idea de tener conceptos ubicuos para toda la organización es tentadora, debemos tener en cuenta intentos similares ya propuestos en la industria, como MDM y modelos de datos canónicos, entre otros, que han expuesto las dificultades de este

enfoque. Los retos pueden ser significativos, especialmente cuando el dominio en el que nos encontramos es lo suficientemente complejo como para crear un modelo único y unificado.

ESBs disfrazados de API Gateways

Resistir

Hace tiempo que advertimos en contra de los ESB centralizados y definimos que los "endpoints inteligentes y canales (pipes) tontos" eran una de las características base para las arquitecturas de microservicios. Desafortunadamente, estamos viendo como los ESBs tradicionales están cambiando su imagen y creando ESBs disfrazados de API Gateways , que naturalmente incentivan a crear API Gateways demasiado ambiciosos. Hay que evitar el engaño de la publicidad: independientemente de como se llamen, poner la lógica de negocio (incluyendo la orquestación y transformación) en una herramienta centralizada crea acoplamiento en la arquitectura, reduce la transparencia e incrementa la dependencia en un proveedor sin añadir ventajas. Los API gateways pueden servir como una abstracción útil para aplicar características horizontales (crosscutting concerns), pero estamos convencidos que la inteligencia debe estar en las propias APIs.

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 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.

Anarquía de los "micro-frontends"

Resistir

Desde que introdujimos originalmente el término en 2016, los *micro-frontends* han ganado popularidad y han sido aceptados de forma generalizada. Pero como suele ocurrir con cualquier nueva técnica con un nombre fácil de recordar, en ocasiones ha sido mal utilizada o de forma abusiva. Es especialmente preocupante la tendencia a utilizar esta arquitectura como una excusa para mezclar una variedad de tecnologías, herramientas o frameworks competidores en la misma página, dando lugar a la anarquía de los micro-frontends. Una forma particularmente escandalosa de este síndrome consiste en el uso de múltiples frameworks de frontend, como React.js y Angular,en la misma aplicación de página única. Si bien esto puede ser técnicamente posible, no es en absoluto aconsejable, excepto cuando forma parte de una estrategia de transición deliberada. Otras propiedades que deberían ser consistentes entre equipos incluyen la técnica de aplicación de estilos (por ejemplo, CSS-en-JS or CSS modules) y los métodos utilizados para integrar los componentes individuales (por ejemplo, iFrames o web components). Además, las organizaciones deberían decidir si estandarizar usando enfoques consistentes o dejar a sus equipos decidir sobre cómo gestionar el estado, el acceso a los datos, las herramientas de construcción, las analíticas y una serie de otras opciones en una aplicación micro-frontend

Llevar "notebooks" a producción

Resistir

En las últimas décadas, los *notebooks* computacionales, mostrados por primera vez por Wolfram Mathematica, han evolucionado para dar soporte a los flujos de trabajo de investigación científica, exploración y educación. De forma natural, al soportar flujos de trabajo de ciencia de datos y mediante herramientas como Jupyter notebooks y Databricks notebooks, se han convertido en un gran complemento al proporcionar un entorno interactivo de computación simple e intuitivo para combinar código que analiza datos junto con texto enriquecido y visualización para contar una historia de datos. Los notebooks se diseñaron para ser el mejor medio para la comunicación y la innovación científica moderna. Sin embargo, en los últimos años, hemos visto una tendencia a utilizar los notebooks como medio para la ejecución de código de calidad de producción utilizado generalmente para guiar operaciones empresariales. Hemos visto a proveedores de plataformas de *notebooks* publicitando el uso de notebooks exploratorios en producción. Este es un caso de buenas intenciones, por intentar democratizar la programación para científicas de datos, con una implementación incorrecta y a cambio de escalabilidad, mantenibilidad, adaptabilidad y otras cualidades que un código de producción longevo necesita proporcionar. Por ello, no recomendamos llevar notebooks a producción y, en su lugar, animamos a empoderar a las personas científicas de datos para que construyan código de calidad para producción con los frameworks de programación adecuados, y así simplificar el uso de herramientas de entrega continua y abstraer la complejidad a través de plataformas de aprendizaje automático de punta a punta.

Técnicas

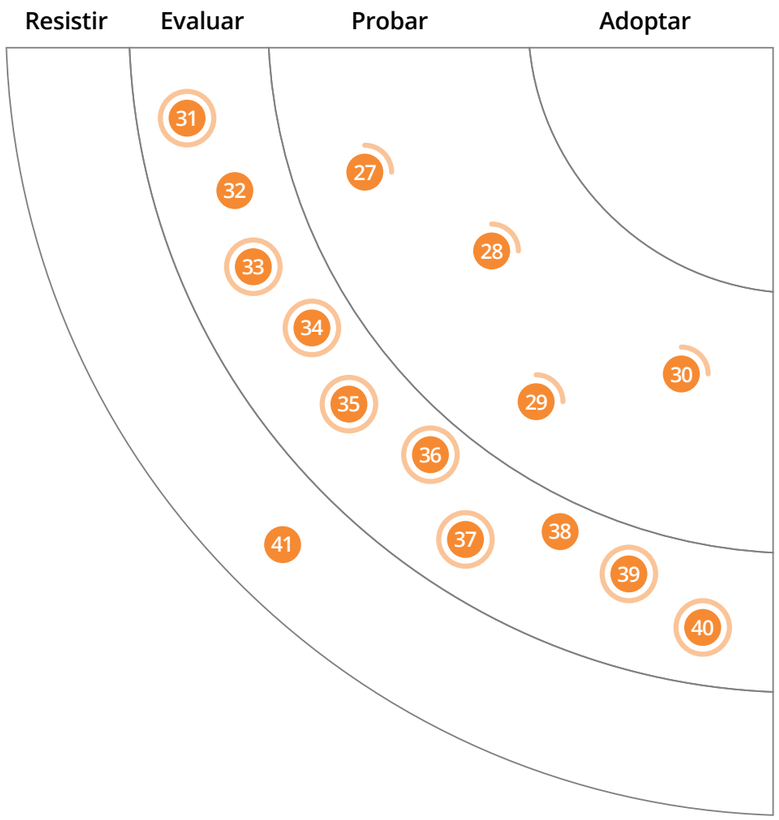
Es especialmente preocupante la tendencia a utilizar la arquitectura de micro-frontends como una excusa para mezclar una variedad de tecnologías, herramientas o frameworks competidores en la misma página. Si bien esto puede ser técnicamente posible, no es en absoluto aconsejable

(Anarquía de los "micro-frontends")

TECHNOLOGY RADAR

Plataformas

Plataformas



Azure DevOps
Probar

Azure DevOps es una colección de servicios administrados que incluye repositorios Git, pipelines de CI/CD, herramientas para pruebas automatizadas, herramientas para la gestión de tareas y planeación ágil y repositorio de paquetes. Hemos visto a nuestros equipos ganar más experiencia en esta plataforma y obtener buenos resultados, lo que significa que Azure DevOps está madurando. Nos gusta especialmente su flexibilidad ya que permite utilizar los servicios que se necesite, incluso si son de distintos proveedores. Por ejemplo, se podría usar un repositorio Git externo con los servicios de *pipeline* de Azure DevOps. Nuestros equipos están especialmente entusiasmados con Azure DevOps Pipelines. No obstante, todos los servicios ofrecen una buena experiencia de desarrollo que ayuda a nuestros equipos a entregar valor.

Debezium
Probar

Debeziumes una plataforma para la captura de cambios en los datos (*change data capture*, CDC) que puede transmitir las modificaciones que ocurren en las bases de datos a tópicos de Kafka. CDC es una técnica popular con muchos casos de uso como la replicación de datos a otras bases de datos, la alimentación de sistemas analíticos, la extracción de microservicios a partir de aplicaciones monolíticas y la invalidación de cachés. Debezium reacciona a cambios en los archivos de registro de la base de datos y cuenta con conectores de CDC para múltiples bases de datos, incluyendo Postgres, MySQL, Oracle y MongoDB. Estamos usando Debezium en varios proyectos y los resultados han sido muy buenos.

Honeycomb
Probar

Honeycomb es un servicio de observabilidad que recoge datos enriquecidos de los sistemas de producción y permite su gestión mediante un muestreo dinámico. Las personas desarrolladoras pueden registrar grandes cantidades de eventos enriquecidos para decidir posteriormente como dividirlos y relacionarlos. Este enfoque interactivo es útil cuando se trabaja con los grandes sistemas distribuidos actuales, porque hemos pasado el punto en donde podemos anticipar de forma razonable qué preguntas querríamos hacer a los sistemas en producción. El equipo de Honeycomb está desarrollando activamente complementos para varios lenguajes y frameworks, y ya hay versiones disponibles

Adoptar

Probar
27. Azure DevOps
28. Debezium
29. Honeycomb
30. JupyterLab

Evaluar
31. Amundsen
32. Kit de Desarrollo en la Nube de AWS
33. Backstage
34. Dremio
35. DuckDB
36. K3s
37. Materialize
38. Pulumi
39. Tekton
40. Trust over IP stack

Resistir
41. Node en exceso

Plataformas

El entorno interactivo de JupyterLab es una evolución de Jupyter Notebooks: extiende sus capacidades originales con autocompletado y celdas de arrastrar y soltar, entre otras nuevas funcionalidades

(JupyterLab)

Backstage de Spotify es una plataforma de código abierto para la creación de portales para desarrolladores, que puede ayudar a estandarizar el número de herramientas y tecnologías que utilizan los equipos y evitar que el ecosistema de software se vuelva fragmentado y complejo.

(Backstage)

para Go, Node, Java Java y Rails, entre otros; además, nuevas funcionalidades están siendo añadidas con frecuencia. El modelo de precios también se ha simplificado para hacerlo más atractivo. A nuestros equipos les encanta.

JupyterLab
Probar

Desde que presentamos a JupyterLab en el anillo Evaluar de nuestra última edición, esta aplicación se ha convertido en la interfaz web preferida para el proyecto Jupyter por muchas de nuestras practicantes de datos. La adopción de JupyterLab está superando rápidamente a Jupyter Notebooks, a la que terminará reemplazando. Si todavía estás utilizando Jupyter Notebooks deberías probar JupyterLab. Su entorno interactivo es una evolución de Jupyter Notebooks: extiende sus capacidades originales con autocompletado y celdas de arrastrar y soltar, entre otras nuevas funcionalidades.

Amundsen
Evaluar

Las personas científicas de datos utilizan gran parte de su tiempo en el descubrimiento de datos, lo que significa que las herramientas que contribuyen en este ámbito generan cierto entusiasmo. Si bien el proyecto Apache Atlas se ha convertido en la herramienta de referencia para la gestión de metadatos, el descubrimiento de datos aún no es fácil de realizar. Para ello aparece Amundsen, una herramienta que se puede desplegar conjuntamente con Apache Atlas para proporcionar una interfaz de búsqueda más apropiada para el descubrimiento de datos.

Kit de Desarrollo en la Nube de AWS
Evaluar

Para muchos de nuestros equipos, Terraform se ha convertido en la opción por defecto para definir infraestructura en la nube. Sin embargo, algunos de nuestros equipos han estado experimentando con el kit de desarrollo en la nube de AWS (CDK de AWS) y les gusta lo que han visto hasta el momento. En particular, les gusta el uso de lenguajes de programación de primera clase en vez de archivos de configuración, lo que les permite utilizar las herramientas existentes, las técnicas para pruebas y sus habilidades. Al igual que con herramientas similares, se debe tener cuidado para asegurar que los despliegues sigan siendo fáciles de entender y mantener. Actualmente el CDK soporta TypeScript, JavaScript, Python, Java y C# con .NET. Seguiremos pendientes de este desarrollo, especialmente porque los equipos de AWS y HashiCorp recientemente presentaron un avance del CDK para Terraform que permite generar configuraciones de Terraform y habilitar el aprovisionamiento con esta tecnología.

Backstage
Evaluar

Las organizaciones están buscando soportar y simplificar los entornos de desarrollo a través de portales para desarrolladores o plataformas. A medida que el número de herramientas y tecnologías se incrementa, se hace cada vez más necesaria alguna forma de estandarización para lograr consistencia, de forma que las personas desarrolladoras puedan enfocarse en la innovación y el desarrollo de productos en lugar de atascarse en reinventar la rueda.

Un portal centralizado para desarrolladores puede ofrecer descubrimiento fácil de servicios y buenas prácticas. Backstage es una plataforma de código abierto creada por Spotify para la creación de portales para desarrolladores. Se basa en plantillas de software, unificando herramientas de infraestructura y documentación técnica consistente y centralizada. Su arquitectura de componentes permite su extensibilidad y adaptabilidad al ecosistema de infraestructura de la organización.

Dremio
Evaluar

Dremio es un motor de lago de datos en la nube que proporciona consultas interactivas sobre almacenes de lagos de datos alojados en la nube. Con Dremio no hay que manejar pipelines de datos para extraer y transformar datos dentro de un almacén de datos separado para alcanzar un rendimiento predictivo. Dremio crea conjuntos de datos virtuales a partir de los datos ingeridos dentro del lago de datos y proporciona una visión uniforme a los consumidores. Presto popularizó la técnica de separar el almacenamiento de la capa de computación y Dremio la lleva más lejos mejorando el rendimiento y optimizando los costos operativos..

DuckDB
Evaluar

DuckDB es una base de datos embebida, basada en columnas, para cargas de trabajo analíticas y de ciencia de datos. Las personas analistas pasan una cantidad significativa de tiempo limpiando y visualizando los datos localmente antes de llevarlos a los servidores. A pesar de

que hemos contado con sistemas de bases de datos por décadas, la mayoría están diseñadas para casos de uso cliente-servidor y, por lo tanto, no son adecuadas para consultas locales interactivas. Para superar esta limitación, las personas analistas normalmente recurren a herramientas de procesamiento de datos en memoria, como Pandas or data.table. Aunque estas herramientas son efectivas, limitan el alcance del análisis al volumen de datos que cabe en la memoria. Nos parece que DuckDB llena bien este vacío proporcionando un motor embebido basado en columnas, optimizado para funciones analíticas en conjuntos de datos locales y de tamaños mayores que la memoria disponible.

K3s
Evaluar

K3s es una distribución ligera de Kubernetes construida para IoT y edge computing. Está empaquetada como un único binario y tiene pocas o ninguna dependencias con el sistema operativo, por lo que es realmente fácil de operar y usar. Utiliza sqlite3 como el motor de almacenamiento predeterminado en lugar de etcd. Tiene un uso de memoria reducido, debido a que ejecuta todos los componentes relevantes en un solo proceso. También consigue ser un binario más pequeño al excluir los controladores de almacenamiento de terceros y de los proveedores de nube que no son relevantes para los casos de uso de K3s. Creemos que vale la pena considerar esta herramienta cuando se cuenta con ambientes con recursos limitados.

Materialize
Evaluar

Materialize es una base de datos en streaming que permite realizar cálculos incrementales sin pipelines de datos complicados. Se debe simplemente describir los cálculos mediante vistas SQL estándar y conectar Materialize al stream de datos. El motor de flujo de datos diferencial subyacente ejecuta cálculos incrementales para proporcionar resultados consistentes y correctos con mínima latencia. A diferencia de las bases de datos tradicionales, no hay restricciones para definir estas vistas materializadas y los cálculos se ejecutan en tiempo real.

Pulumi
Evaluar

Hemos visto un creciente pero lento interés en Pulumi. Esta utilidad llena una brecha en el mundo de la infraestructura como código, donde Terraform está muy afianzado. Si bien Terraform ha sido ampliamente usado y probado, su naturaleza declarativa presenta capacidades inadecuadas de abstracción y carece de la capacidad de ser probado. Terraform es adecuado cuando la infraestructura es completamente estática, pero para la definición de infraestructuras dinámicas se necesita un lenguaje de programación real. Pulumi se distingue por permitir que las configuraciones se escriban en TypeScript/JavaScript, Python y Go sin ser necesario un lenguaje de marcado o de plantillas. Pulumi está altamente enfocado en arquitecturas nativas a la nube, incluyendo contenedores, funciones serverless y servicios de datos, proporcionando buen soporte para Kubernetes. Recientemente, el CDK de AWS se ha posicionado como contendor, pero Pulumi se mantiene como la única herramienta neutral del área, en lo que respecta a los proveedores. Anticipamos una amplia adopción de Pulumi en el futuro y esperamos la aparición de herramientas viables y de ecosistemas de conocimiento que le den soporte.

Tekton
Evaluar

Tekton es una nueva plataforma nativa de Kubernetes-para la gestión de pipelines de integración y entrega continua (CI/CD). Esta plataforma no solamente se instala y ejecuta sobre Kubernetes sino que también define sus pipelines de CI/CD como recursos personalizados de Kubernetes. Esto significa que el pipeline puede ser controlado por las aplicaciones cliente nativas de Kubernetes (CLI o APIs) y que se puede aprovechar las funcionalidades de gestión de recursos subyacentes como los rollbacks. El formato de declaración de pipelines es flexible y permite definir flujos de trabajo condicionales, rutas de ejecución paralelas y manejar tareas finales para hacer limpieza, entre otras funcionalidades. Como resultado, Tekton puede soportar flujos de despliegue complejos e híbridos con rollbacks, canary releases y más. Tekton es de código abierto y también se ofrece como un servicio administrado en GCP. A pesar de que la documentación podría ser mejor y la comunidad está creciendo, hemos estado usando Tekton exitosamente con cargas de trabajo en producción para AWS.

Plataformas

K3s es una distribución ligera de Kubernetes construida para IoT y edge computing, que elimina los controladores de almacenamiento de terceros y los proveedores de nube que no son relevantes para estos casos de uso.

(K3s)

Materialize es una base de datos en streaming que permite hacer cálculos incrementales sin complicados pipelines de datos.

(Materialize)

Plataformas

Este stack técnico y de gobernanza de cuatro capas tiene por objeto establecer una base de referencia para una forma de gestión descentralizada de la identidad altamente interoperable.

(Trust over IP stack)

Node.js es muy popular. Pero eso no lo hace adecuado para todo, es una mala elección, para cargas de trabajo pesadas. Advertimos contra la tendencia a usar Node. js indiscriminadamente o por razones equivocadas.

(Node en exceso)

Trust over IP stack
Evaluar

Los retos continuos respecto a la forma en que los individuos y las organizaciones establecen confianza digitalmente en internet está dando lugar a un nuevo enfoque sobre cómo comprobar la identidad, cómo compartir y verificar los atributos necesarios para establecer la confianza y cómo asegurar las transacciones. Nuestro Radar presenta algunas de las tecnologías fundamentales como la identidad descentralizada y credenciales verificables que posibilitan esta nueva era en la confianza digital.
Sin embargo, este cambio de escala global no sería posible sin la estandarización de las tecnologías de gobernanza que permitan la interoperabilidad. La nueva *Trust over IP Foundation*, perteneciente a Linux Foundation, está destinada a conseguirlo. Tomando en cuenta la forma en que la estandarización de TCP/IP, como el puente estrecho de Internet, ha favorecido la interoperabilidad entre millones de dispositivos, el grupo está definiendo Trust over IP Stack mediante cuatro capas técnicas y de gobierno. El stack incluye servicios públicos como identificadores descentralizados, comunicaciones descentralizadas de identidad para protocolos estandarizados para que agentes, como las billeteras digitales, se comuniquen; protocolos de intercambio de datos como flujos para proporcionar y verificar credenciales verificables, así como ecosistemas de aplicaciones tales como educación, finanzas, salud, etc. Si estás revisando los sistemas de identidades y cómo se establecen los mecanismos de confianza en tu ecosistema, te sugerimos considerar el stack de ToIP y sus herramientas de soporte conocidas como Hyperledger Aries.

Node en exceso
Resistir

Las tecnologías, especialmente las ampliamente populares, tienen tendencia a ser sobreutilizadas. 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.js 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

TECHNOLOGY RADAR

Herramientas

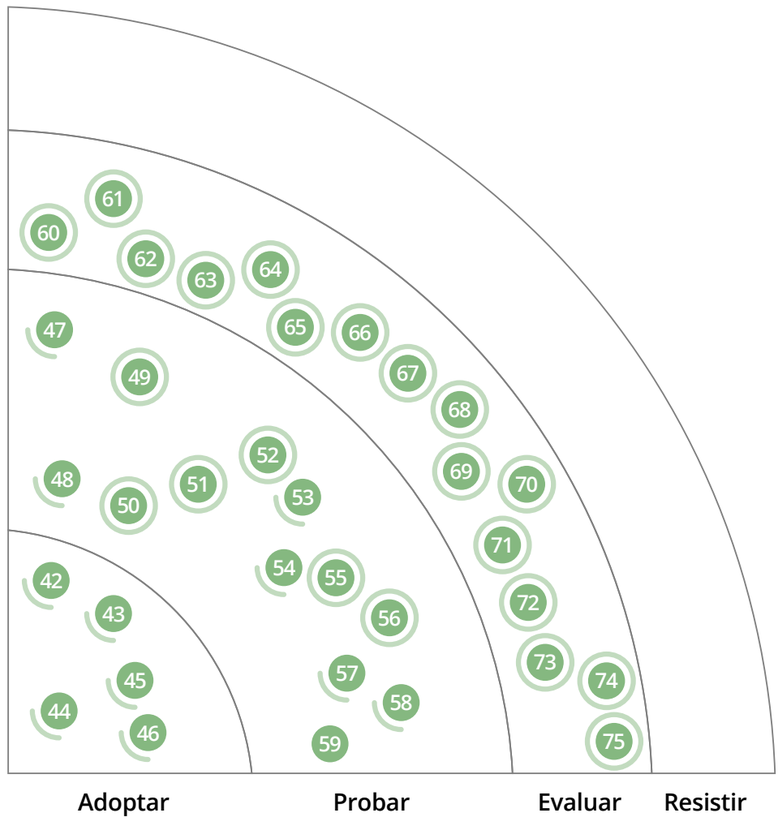
Herramientas

Airflow
Adoptar

Airflow sigue siendo nuestra herramienta de código abierto favorita y más utilizada para la gestión de flujos de trabajo para pipelines de procesamiento de datos como grafos acíclicos dirigidos (DAGs). Esta es un área en crecimiento con herramientas de código abierto como Luigi and Argo y herramientas más específicas como Azure Data Factory or AWS Data Pipeline. Sin embargo, Airflow se diferencia porque su definición programática de flujos de trabajo se realiza en archivos de configuración con poco código, provee soporte para pruebas automatizadas, es de código abierto, es multiplataforma, ofrece muchas posibilidades de integración con el ecosistema de datos y tiene soporte de una gran comunidad. No obstante, en arquitecturas de datos descentralizadas, como la malla de datos, esta herramienta se puede usar como un orquestador centralizado de flujos de trabajo.

Bitrise
Adoptar

Bitrise, una herramienta de entrega continua de dominio específico para aplicaciones móviles, sigue siendo útil para el flujo de trabajo para el desarrollo móvil y los equipos deberían utilizarlo. Bitrise puede construir, probar y desplegar aplicaciones móviles desde la computadora portátil de la persona desarrolladora hasta la publicación en la tienda de aplicaciones. Es fácil de configurar y proporciona un conjunto completo de pasos listos para usar para la mayoría de las necesidades del desarrollo móvil.



Dependabot
Adoptar

Entre las herramientas disponibles para mantener las dependencias actualizadas, Dependabotes la opción predeterminada en nuestra opinión. La integración de Dependabot con GitHub es fluida y automáticamente envía pull requests para actualizar las dependencias a sus últimas versiones. Esto se puede habilitar a nivel de organización, haciendo también muy fácil para los equipos recibir estos pull requests. Si no estás usando GitHub, es posible usar las bibliotecas de Dependabot dentro del pipeline de construcción. Si te interesa una herramienta alternativa, también podrías considerar Renovate, que soporta una gama más amplia de servicios, incluyendo GitLab, Bitbucket y Azure DevOps.

Helm
Adoptar

Helm es un gestor de paquetes para Kubernetes. Viene con un repositorio de aplicaciones curadas de Kubernetes que se mantienen en el repositorio Charts oficial. Desde la última vez que hablamos sobre Helm, Helm 3

Adoptar
42. Airflow
43. Bitrise
44. Dependabot
45. Helm
46. Trivy

Probar
47. Bokeh
48. Concourse
49. Dash
50. jscodeshift
51. Kustomize
52. MLflow
53. Pitest
54. Sentry
55. ShellCheck
56. Stryker
57. Terragrunt
58. tfsec
59. Yarn

Evaluar
60. CML
61. Eleventy
62. Flagger
63. gossm
64. Great Expectations
65. k6
66. Katran
67. Kiali
68. LGTM
69. Litmus
70. Opacus
71. OSS Index
72. Playwright
73. pnpm
74. Sensei
75. Zola

Resistir

Herramientas

Trivy un escáner de vulnerabilidades para contenedores que publica un binario independiente, haciendo más fácil configurar y ejecutar el análisis localmente.

(Trivy)

jscodeshift ayuda a mitigar el malestar de mantener bases de código JavaScript a gran escala. Lo hemos encontrado particularmente útil para mantener los sistemas de diseño (design systems).

(jscodeshift)

ha sido publicado, y el cambio más significativo es la eliminación de Tiller, el componente de servidor de Helm 2. El beneficio de un diseño sin Tiller es que solo se puede realizar cambios al clúster de Kubernetes desde el lado cliente, es decir, solo es posible modificar el clúster de acuerdo a los permisos que se tenga como usuario del comando Helm. Hemos usado Helm en varios proyectos de clientes y su gestión de dependencias, plantillas y mecanismo de *hook* ha simplificado enormemente la gestión del ciclo de vida de las aplicaciones en Kubernetes.

Trivy

Adoptar

Los pipelines de construcción que crean y despliegan contenedores deberían incluir el escaneo de seguridad de los contenedores. A nuestros equipos les gusta especialmente Trivy, un escáner de vulnerabilidades para contenedores. Hemos probado Clair y Anchore Engine among entre otras buenas herramientas en esta área. A diferencia de Clair, Trivy no solo verifica los contenedores, sino también las dependencias en el código. Además, como Trivy se publica como un binario independiente, es más fácil configurar y ejecutar el análisis localmente. Otros beneficios de Trivy son que es de código abierto y soporta contenedores sin distribución (*distroless*).

Bokeh

Probar

Bokeh es una de las principales bibliotecas en Python para crear gráficos científicos y visualizaciones de datos que se renderizan en el navegador a través de JavaScript. Estas herramientas, en comparación con las de escritorio que generan imágenes estáticas, facilitan la reutilización de código a la hora de realizar trabajo exploratorio en aplicaciones web. Bokeh hace este trabajo especialmente bien: es una biblioteca consolidada y completa, y nos gusta que concentra su funcionalidad en la capa de presentación sin involucrarse en otros temas como la agregación de datos (ver ggplot) o en el desarrollo web (como Shiny o Dash). Por ello, si la separación de responsabilidades es prioritaria, es un placer usar Bokeh. Cabe decir que esta herramienta sí que proporciona componentes de UI para web y puede ejecutarse en modo servidor, pero estas funcionalidades son opcionales y dependen de la necesidad. Bokeh es flexible y no hace supuestos sobre cómo se utilizará o si tiene demasiadas dependencias (como pandas o *notebooks*).

Concourse

Probar

Implementar pipelines para entrega continua sostenibles, que puedan construir y desplegar software de producción en múltiples ambientes, requiere de una herramienta que trate a los pipelines de construcción y a los artefactos como protagonistas. Cuando empezamos a evaluar a Concourse, nos gustó su modelo de operación simple y flexible, el principio de compilación basada en contenedores y el hecho de que obliga a definir los pipelines como código. Desde entonces su usabilidad ha mejorado y su modelo simple ha resistido el paso del tiempo. Muchos de nuestros equipos y clientes han usado Concourse con éxito en instalaciones de pipelines de gran magnitud durante períodos de tiempo prolongados. A menudo aprovechamos la flexibilidad de Concourse para ejecutar agentes donde se requiera, por ejemplo, cuando las pruebas de integración de hardware requieren de una instalación local.

Dash

Probar

Esta edición del Radar presenta varias herramientas nuevas para la creación de aplicaciones web que ayudan a los usuarios finales a visualizar e interactuar con datos. A diferencia de simples bibliotecas de visualización, como D3. lestas herramientas minimizan el esfuerzo necesario para construir aplicaciones analíticas independientes para manipular conjuntos de datos existentes. Dash, de Plotly, está ganando popularidad entre científicas de datos porque permite crear aplicaciones analíticas ricas en funcionalidades usando Python. Dash mejora las bibliotecas para datos de Python y, al igual que Shiny, está basada en R. Estas aplicaciones son a veces conocidas como tableros o dashboards, a pesar que estos nombres no hacen justicia a las posibles funcionalidades que proveen. Dash es particularmente adecuado para construir aplicaciones escalables y listas para producción, a diferencia de Streamlit, otra herramienta similar. Puede considerarse utilizar Dash si se necesita presentar a usuarias de negocio, análisis más sofisticados que herramientas de poco o ningún código pueden ofrecer, como Tableau.

jscodeshift

Probar

Mantener grandes bases de código JavaScript nunca es fácil, y es especialmente desafiante cuando se migran cambios importantes. Los IDEs con capacidades de refactorización pueden ayudar en escenarios simples. Sin embargo, cuando la base de código es una biblioteca de la que existen múltiples dependencias,

cada vez que se realiza un cambio importante se debe revisar varias bases de código dependientes para realizar las actualizaciones adecuadas, lo que requiere supervisión humana y debe realizarse manualmente. jscodeshift, un conjunto de herramientas para refactorizar JavaScript y TypeScript, ayuda a aliviar este dolor. Esta herramienta analiza el código usando árboles de sintaxis abstracta (AST) y proporciona un API para manipular el árbol con varias transformaciones (por ejemplo, agregar, renombrar y eliminar propiedades de componentes existentes) y luego exportar el árbol como código fuente final. jscodeshift también incluye una utilidad simple para pruebas unitarias que puede aplicar el desarrollo dirigido por pruebas (TDD) para escribir codemods de migración. Hemos encontrado que jscodeshift es bastante útil para el mantenimiento de *design systems*.

Kustomize
Probar

Kustomize es una herramienta para administrar y personalizar los archivos de manifiesto de Kubernetes. Permite seleccionar y parchar los recursos base de Kubernetes antes de aplicarlos a diferentes ambientes y ahora es compatible de forma nativa con kubectl. Nos gusta porque ayuda a mantener el código sin repeticiones y a diferencia de Helm (que está tratando de hacer demasiadas cosas: administración de paquetes, administración de versiones, etc.), encontramos que Kustomize sigue la filosofía de Unix: hacer una cosa bien y esperar que la salida de cada programa sea la entrada de otro.

MLflow
Probar

MLflow es una herramienta de código abierto para el seguimiento de experimentos de aprendizaje automático y la gestión del ciclo de vida. El flujo de trabajo para desarrollar y evolucionar continuamente un modelo de aprendizaje automático incluye una serie de experimentos (una colección de ejecuciones), el seguimiento del rendimiento de estos experimentos (una colección de métricas) y el seguimiento y ajuste de modelos (proyectos). MLflow facilita este flujo de trabajo muy bien al apoyar los estándares abiertos existentes y se integra bien con muchas otras herramientas en el ecosistema. MLflow, como servicio gestionado por Databricks en la nube, disponible en AWS and Azure, está madurando rápidamente y lo hemos utilizado con éxito en nuestros proyectos. Consideramos que MLflow es una gran herramienta para la gestión y el seguimiento de modelos, que soporta tanto modelos de interacción basados en UI como en API. Nuestra única y creciente preocupación es que MLflow está tratando de entregar respuestas a demasiados temas vinculados, como una sola plataforma, como el servicio de modelos y la puntuación.

Pitest
Probar

Los métodos de prueba tradicionales se centran en evaluar si nuestro código de producción hace lo que debería hacer. Sin embargo, podemos cometer errores en el código de prueba introduciendo pruebas incompletas o sin valor que crean una falsa sensación de confianza. Es aquí donde entran en juego las pruebas de mutación: comprueban la calidad de las pruebas en sí, encontrando casos extremos difíciles de detectar. Nuestros equipos usan Pitest desde hace tiempo, y recomendamos su uso en proyectos Java para medir la salud del conjunto de pruebas. En resumen, las pruebas de mutación introducen cambios en el código de producción y ejecutan las mismas pruebas una segunda vez; si los resultados aún son positivos significa que las pruebas no están bien y necesitan mejorar. Si usas un lenguaje de programación distinto a Java, Stryker es una buena elección en este espacio.

Sentry
Probar

Sentry es una aplicación de monitoreo multiplataforma que se enfoca en el reporte de errores. Las herramientas como Sentry se diferencian de las soluciones tradicionales para el registro de logs, como ELK Stack por su foco en descubrir, investigar y corregir errores. Sentry ya lleva un tiempo en el mercado y soporta varios lenguajes y frameworks. En ThoughtWorks lo hemos utilizado en muchos proyectos y ha sido realmente útil para rastrear errores, verificar si un commit realmente resolvió una incidencia y para generar alertas en caso que alguna incidencia resurja debido a problemas de regresión.

ShellCheck
Probar

Aún cuando las herramientas han mejorado considerablemente en el área de infraestructura, en algunas ocasiones aún tiene sentido escribir un script para el shell. Claramente, la sintaxis de los scripts para el shell podría describirse como arcaica y, tomando en cuenta que últimamente hemos tenido menos práctica escribiéndolos, nos ha comenzado a gustar ShellCheck, un *linter* para este tipo de *scripts*. ShellCheck puede usarse desde la línea de comandos, como

Herramientas

Kustomize es una herramienta para administrar y personalizar los archivos de manifiesto de Kubernetes. Permite seleccionar y parchar los recursos base de Kubernetes antes de aplicarlos a diferentes ambientes.

(Kustomize)

Sentry es una aplicación de monitoreo multiplataforma que se enfoca en el reporte de errores. Ha ayudado a nuestros equipos a rastrear errores, verificar si un commit realmente resolvió una incidencia y para generar alertas en caso que alguna incidencia resurja debido a problemas de regresión.

(Sentry)

Herramientas

Flagger es una herramienta útil para ajustar la porción de tráfico que se enruta a una nueva versión de un servicio - lo cual es útil cuando se trabaja con mallas de servicio y API gateways.

(Flagger)

parte de un proceso de compilación, o aún mejor, como una extensión para varios IDEs populares. Su wiki contiene una descripción detallada de varios cientos de problemas que ShellCheck puede detectar y la mayoría de herramientas e IDEs proveen una forma conveniente de acceder a la página de la wiki correspondiente cuando encuentra un problema.

Stryker
Probar

Stryker es un participante relativamente nuevo en el mundo de las pruebas de mutación. Al igual que Pitest, Stryker nos permite evaluar la calidad de nuestras pruebas. Hemos obtenido buenos resultados con proyectos en JavaScript, y también es compatible con proyectos en C# y Scala. Stryker es fácil de utilizar, muy personalizable y nos ha permitido aumentar la cobertura de código así como la confianza en las aplicaciones que desarrollamos para nuestros clientes.

Terragrunt
Probar

Hemos usado Terraform extensivamente para crear y manejar infraestructuras en la nube. En nuestra experiencia con instalaciones más grandes, donde el código se encuentra dividido en módulos y las configuraciones se incluyen de diferentes maneras, los equipos eventualmente se tropiezan con un muro inevitable de repetición ocasionado por la falta de flexibilidad. Hemos resuelto esto utilizando Terragrunt, una delgada capa para Terraform que implementa las prácticas propuestas por Yevgeniy Brikman en su libro *Terraform: Up and Running*. Hemos encontrado útil a Terragrunt porque fomenta la creación de módulos versionados y la reutilización para distintos entornos. Los hooks para el ciclo de vida son otra funcionalidad útil que entrega flexibilidad adicional. En términos de empaquetado, Terragrunt tiene las mismas limitaciones que Terraform: no hay una forma apropiada para definir paquetes o dependencias entre paquetes. Como alternativa, puedes utilizar módulos y versiones específicas asociadas a una etiqueta de Git.

tfsec
Probar

La seguridad es una preocupación general y capturar los riesgos con anticipación siempre es mejor que enfrentar los problemas más adelante. En el espacio de la infraestructura como código, donde Terraform se ha establecido como la opción obvia para gestionar ambientes en la nube, ahora también podemos contar con tfsec, una herramienta de análisis estático que escanea las plantillas de Terraform para encontrar posibles problemas de seguridad. Nuestros equipos han usado tfsec con bastante éxito: la herramienta es fácil de configurar y utilizar, lo que la convierte en una excelente opción para cualquier equipo de desarrollo decidido a mitigar riesgos y prevenir brechas de seguridad antes de que sucedan. Sus reglas preestablecidas para diferentes proveedores de la nube, como AWS y Azure, complementa los beneficios que brinda tfsec a los equipos que usan Terraform.

Yarn
Probar

Yarn sigue siendo el gestor de paquetes elegido por muchos equipos. Estamos emocionados por Yarn 2, una nueva versión mayor de la herramienta que trae una larga lista de cambios y mejoras. Además de los ajustes de usabilidad y mejoras en el apartado de los espacios de trabajo, Yarn 2 presenta el concepto de “cero instalaciones”, que permite a las personas desarrolladoras ejecutar un proyecto directamente después de clonarlo. Sin embargo, Yarn 2 incluye algunos cambios de importancia que hacen de la actualización una tarea no trivial. También, recurre por defecto a los entornos plug’n’play (PnP) y al mismo tiempo no soporta React Native en entornos PnP. Por supuesto que los equipos pueden optar por no utilizar PnP o mantenerse en Yarn 1. Sin embargo, deben tener en cuenta que Yarn 1 está ahora en modo de mantenimiento.

CML
Evaluar

Hemos incluido a la entrega continua para aprendizaje automático como una técnica en ediciones anteriores del Radar y ahora queremos destacar una nueva herramienta prometedora llamada Continuous Machine Learning (or CML) de las mismas personas que crearon DVC. CML tiene como objetivo traer las mejores prácticas de ingeniería de CI y CD a los equipos de AI y ML y puede ayudar a organizar la infraestructura de MLOps soportada por prácticas y herramientas de ingeniería de software tradicional, en lugar de crear plataformas de AI separadas. Nos gusta que hayan priorizado el soporte para DVC y lo vemos como una buena señal para esta nueva y creciente herramienta.

Eleventy
Evaluar

Desde hace tiempo nos gusta la idea del uso de generadores de sitios estáticos para evitar complejidad y mejorar el rendimiento, siempre que el caso de uso lo permita. Si bien Eleventy ha existido por algún tiempo, no es sino recién que nos ha llamado la atención por su madurez y porque otras alternativas favoritas ya existentes, como Gatsby.js, mostraron ciertos problemas de

escalabilidad. Eleventy es fácil de aprender y de utilizar para construir sitios. Nos gusta además que permite crear, con facilidad, marcado semántico (y por lo tanto más accesible) con sus plantillas, además de su soporte simple y robusto para la paginación.

Flagger
Evaluar

Las mallas de servicios y los API gateways proveen una forma conveniente de enrutar el tráfico a los distintos microservicos que implementan la misma interfaz de API. Flagger utiliza esta característica para ajustar dinámicamente la cantidad de tráfico que es dirigida a una nueva versión de un servicio. Esta es una técnica frecuente para los *canary releases* o los despliegues blue-green (blue-green deployments). Flagger funciona junto a una variedad de proxies populares (como Envoy y Kong) para incrementar progresivamente el número de peticiones que se envían a un servicio y para reportar métricas sobre la carga, y así obtener retroalimentación rápida sobre el funcionamiento de una nueva versión. Nos gusta que Flagger simplifica esta valiosa práctica lo cual ayuda a extender su adopción. Aunque Weaveworks la patrocina, Flagger es una utilidad independiente y no hay obligación de usarla en conjunto con las otras herramientas de esta empresa.

gossm
Evaluar

Cuando se requiere conectarse a instancias de servidores en AWS, se recomienda hacerlo mediante un servidor bastión en vez de realizar una conexión directa. Sin embargo, provisionar un servidor de este tipo para este único propósito puede ser frustrante. Por eso el AWS Systems Manager's Session Manager facilita el uso de túneles para conectarse a las instancias de servidores con más comodidad. gossm ies una herramienta de código abierto para la línea de comandos que hace aún más conveniente el uso de Session Manager. gossm permite aprovechar la seguridad suministrada por Session Manager y las políticas de IAM desde la terminal usando herramientas como ssh y scp. También proporciona algunas funcionalidades que la herramienta de línea de comandos propia de AWS no tiene, incluyendo el descubrimiento de servidores e integración SSH.

Great Expectations
Evaluar

Con el surgimiento de CD4ML, los aspectos operacionales de la ingeniería y la ciencia de datos han recibido más atención. La gobernanza de datos automatizada es un aspecto de este desarrollo. Great Expectations es un *framework* que permite crear controles integrados que señalan anomalías o problemas de calidad en los *pipelines* de datos. Al igual que las pruebas unitarias corren en un pipeline de compilación, Great Expectations realiza verificaciones durante la ejecución de un *pipeline* de datos. Esto es útil no solo para implementar una especie de Andon para *pipelines* de datos sino también para garantizar que los algoritmos basados en modelos permanezcan dentro del rango operativo determinado por sus datos de entrenamiento. Los controles automatizados como este pueden ayudar a distribuir, democratizar y custodiar el acceso a los datos. Great Expectations también contiene una herramienta de generación de perfiles para ayudar a comprender las cualidades de un conjunto de datos en particular y establecer límites apropiados.

k6
Evaluar

Estamos entusiasmados con k6, una herramienta relativamente nueva en el ecosistema de las pruebas de rendimiento con un fuerte foco en la experiencia para las personas desarrolladoras. La interfaz de línea de comandos (CLI) de k6 ejecuta scripts JavaScript y permite configurar el tiempo de ejecución y el número de usuarios virtuales. La CLI tiene varias características avanzadas que permiten, entre otras cosas, ver las estadísticas actuales antes de que las pruebas hayan terminado de ejecutarse, escalar el número de usuarios virtuales originalmente establecidos e incluso pausar y continuar la ejecución de una prueba. La salida que presenta la aplicación proporciona un conjunto de métricas personalizables con transformadores que permiten visualizar los resultados en Datadog y en otras herramientas de observabilidad. La inclusión de checks en los scripts simplifica la adición de pruebas de rendimiento en los pipelines de CI/CD. Para pruebas de rendimiento avanzadas vale la pena probar k6 Cloud, la versión comercial, que provee escalamiento en la nube y visualizaciones adicionales.

Katran
Evaluar

Katran es un balanceador de carga de capa 4 de alto rendimiento. No es para todo el mundo, pero si el requerimiento es proporcionar redundancia para balanceadores de carga de capa 7 (como HAProxy o NGINX) o si se necesita escalar los balanceadores de carga a dos o más servidores, entonces recomendamos evaluar Katran. Lo vemos como una opción flexible y eficiente frente a técnicas como el uso de balanceadores de carga round robin de capa 7 para DNS o el modelo de Kernel IPVS que los ingenieros de

Herramientas

Great Expectations es un framework que permite crear controles integrados que señalan anomalías o problemas de calidad en los pipelines de datos.

(Great Expectations)

Herramientas

Es una herramienta de análisis estático de código con foco en la seguridad, que está respaldada por un catálogo de reglas de programación segura

(LGTM)

redes usualmente adoptan para resolver situaciones similares.

Kiali
Evaluar

Debido al aumento en el uso de mallas de servicio para desplegar colecciones de microservicios dentro de contenedores, esperamos ver cómo surgen herramientas que automatizan y simplifican las tareas de administración relacionadas con este estilo arquitectónico. Kiali es una de ellas. Kiali ofrece una interfaz de usuario gráfica para observar y controlar redes de servicios desplegados con Istio. Hemos encontrado útil a Kiali para visualizar la topología de los servicios en una red y entender cómo se dirige el tráfico entre ellos. Por ejemplo, cuando se usa en conjunto con Flagger, Kiali nos puede mostrar las peticiones que han sido dirigidas a cada servicio en un esquema de canary release. Nos gusta especialmente la habilidad de Kiali para inyectar artificialmente errores de red en una malla de servicio para probar la resiliencia en caso de interrupciones a nivel de red. Esta práctica es frecuentemente ignorada por la complejidad de configurar y ejecutar pruebas de fallos en una malla compleja de microservicios.

LGTM
Evaluar

Escribir código seguro es más importante que nunca, aunque es apenas una de las tantas cosas que las personas desarrolladoras tienen que priorizar. LGTM proporciona una red de seguridad y los medios para beneficiarse de una base de conocimiento sobre prácticas de programación segura. Es una herramienta de análisis estático de código con foco en la seguridad, que está respaldada por un catálogo de reglas de programación segura (parcialmente de código abierto). Las reglas están implementadas como consultas sobre el código usando el lenguaje de consulta CodeQL LGTM puede utilizarse para integrar comprobaciones de seguridad de caja blanca en el pipeline de entrega continua para Java, JavaScript, Python, C# y C/C++. LGTM y CodeQL son parte del Github Security Lab.

Litmus
Evaluar

Litmus es una herramienta de ingeniería del caos que presenta una baja barrera de entrada y que permite inyectar fácilmente diferentes escenarios de error en los clusters de Kubernetes. Nos entusiasma sobre todo las posibilidades que Litmus ofrece además del típico eliminado aleatorio de pods, como la simulación de problemas de red, de CPU, de memoria y de E/S. Litmus también permite realizar experimentos a la medida para simular errores para servicios comunes como Kafka y Cassandra.

Opacus
Evaluar

El concepto de privacidad diferencial apareció por primera vez en el Radar en 2016. Si bien el problema de romper la privacidad mediante consultas sistemáticas al modelo de inferencia ya se había reconocido entonces, se consideraba fundamentalmente un problema teórico dado que no existían suficientes recursos. La industria ha carecido de herramientas para prevenir que esto ocurriera. Opacus es una nueva biblioteca escrita en Python que puede utilizarse conjuntamente con PyTorch para ayudar a frustrar un tipo de ataque de privacidad diferencial. Aunque se trata de un desarrollo prometedor, ha sido un reto encontrar el modelo correcto y el conjunto de datos en el que éste aplica. La biblioteca es aún bastante nueva por lo que esperamos ver cómo será aceptada en el futuro.

OSS Index
Evaluar

Para un equipo de desarrollo es importante identificar si las dependencias de su aplicación tienen vulnerabilidades conocidas. OSS Index puede usarse para este propósito. OSS Index es un catálogo gratuito de componentes de código abierto y herramientas de escaneo diseñadas para ayudar a las personas desarrolladoras a identificar vulnerabilidades, entender los riesgos y mantener su software seguro. Nuestros equipos ya han integrado este índice en sus pipelines usando herramientas comoAuditJS y el complemento Sonatype Scan para Gradle. La velocidad es rápida, las vulnerabilidades se identifican con precisión y pocos falsos positivos ocurren.

Playwright
Evaluar

El campo de las pruebas para interfaces de usuario Web sigue bastante activo. Algunas de las personas que construyeron Puppeteer se mudaron a Microsoft y actualmente están aplicando sus conocimientos y lecciones aprendidas en Playwright, una herramienta que nos permite escribir pruebas para Chromium y Firefox, e incluso WebKit, todo a través de la misma API. Playwright ha ganado algo de atención por dar soporte a todos los principales motores de navegación, lo que consigue en la actualidad mediante versiones parchadas de Firefox y WebKit.

Queda por ver cuán rápido pueden ponerse al día otras herramientas, con el soporte cada vez mayor del protocolo de Chrome DevTools, como un API común para la automatización de navegadores.

pnpm
Evaluar

pnpm es un gestor de paquetes prometedor para Node.js que tenemos en la mira por la gran velocidad y eficacia que tiene en comparación con otros gestores de paquetes. Las dependencias se guardan en un único lugar en el disco y se enlazan a los directorios node_modules respectivos. pnpm también permite la optimización incremental a nivel de archivo, cuenta con un API estable que facilita la extensión y personalización e incluye el modo de almacenamiento en servidor, lo cual acelera aún más la descarga de dependencias. En caso de tener muchos proyectos con las mismas dependencias, puede ser una buena idea explorar pnpm más a profundidad.

Sensei
Evaluar

Sensei de Secure Code Warrior, es un complemento para los IDEs para Java que simplifica crear y distribuir lineamientos de calidad para codificación segura. En ThoughtWorks promovemos con frecuencia el uso de “herramientas en lugar de reglas”, es decir, conseguir que sea fácil hacer lo correcto en lugar de tener que aplicar reglamentos y procedimientos de gobierno en forma de listas de control, y esta herramienta encaja en esta filosofía. Las personas desarrolladoras crean recetas que pueden ser compartidas fácilmente con el resto del equipo. Éstas pueden ser simples o complejas y se implementan como consultas sobre el AST (Abstract Syntax Tree) de Java. Algunos ejemplos son la generación de advertencias sobre inyección de SQL, debilidades criptográficas, etc. Otra característica que nos gusta: Sensei se ejecuta cada vez que el código cambia en el IDE y ofrece una retroalimentación más rápida que las herramientas tradicionales de análisis estático.

Zola
Evaluar

Zola es un generador de sitios estáticos escrito en Rust. Por ende está disponible como un ejecutable sólo y sin dependencias, es muy rápido y soporta todas las cosas usuales que se espera como Sass, contenido en markdown y carga automática de cambios en caliente. Hemos tenido éxito construyendo sitios estáticos con Zola y apreciamos lo intuitivo que es para usar.

Herramientas

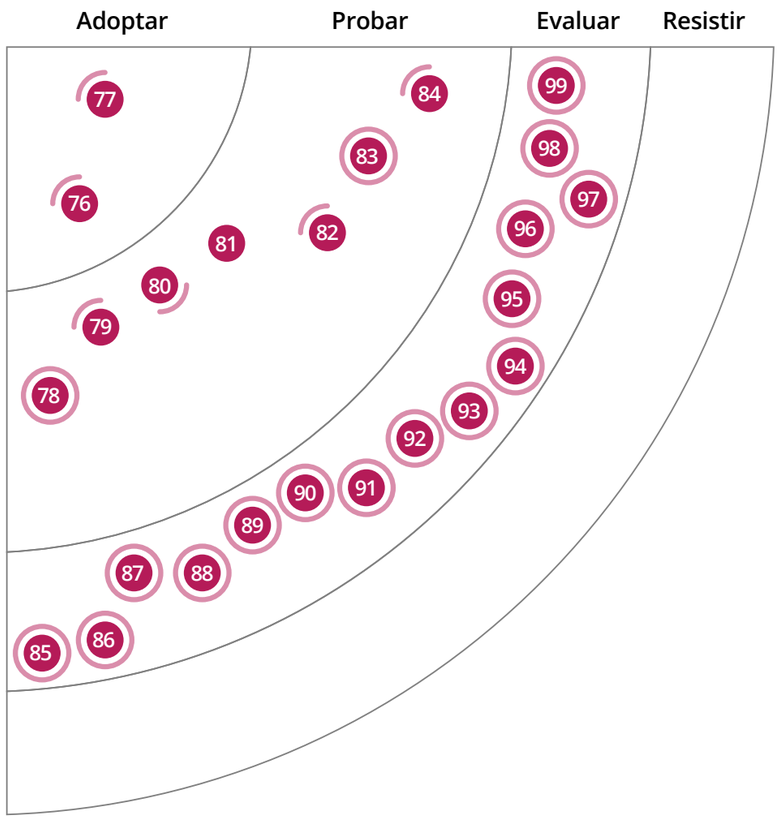
Sensei facilita la creación y distribución de guías de calidad para código seguro.

(Sensei)

TECHNOLOGY RADAR

Lenguajes & Frameworks

Lenguajes & Frameworks



Arrow
Adoptar

Arrow es promocionado como el complemento funcional de la biblioteca estándar de Kotlin. De hecho, el paquete de abstracciones de alto nivel listas para usar que ofrece Arrow ha demostrado ser tan útil que nuestros equipos ahora lo tienen como su elección por defecto para trabajar con Kotlin. Recientemente, en preparación para el lanzamiento de la versión 1.0, el equipo de Arrow introdujo varios cambios, entre los que se incluyen nuevos módulos y también algunas deprecaciones y eliminaciones.

jest-when
Adoptar

jest-when es una biblioteca ligera en JavaScript que complementa a Jest y ayuda a comparar los argumentos de las invocaciones a funciones simuladas (*mocked functions*). Jest es una gran herramienta para probar la tecnología en general; jest-when permite definir expectativas sobre argumentos específicos en las funciones simuladas, lo que posibilita escribir pruebas unitarias más robustas de módulos con muchas dependencias. Es fácil de usar y provee gran soporte para múltiples comparadores (*matchers*), razón por la cual nuestros equipos han hecho de jest-when su opción por defecto en este campo.

Fastify
Probar

En el caso en el que sea necesario realizar una implementación usando Node.js vemos que Fastify es una opción con la cual nuestros equipos se sienten contentos de trabajar. Este framework web ofrece facilidad en el manejo de validaciones de peticiones y respuestas, soporte para TypeScript y un ecosistema de complementos que brinda a nuestros equipos una experiencia más sencilla para el desarrollo de software. Si bien es una buena opción en el ecosistema de Node.js, nos mantenemos firmes en nuestro consejo anterior: evitar los escenarios de uso excesivo de Node.

Immer
Probar

Con la creciente complejidad de las aplicaciones de página única (single-page applications, SPAs) con JavaScript, manejar la previsibilidad del estado es cada vez más importante. La inmutabilidad puede ayudar a garantizar que nuestras aplicaciones se comporten de forma consistente, pero por desgracia, JavaScript no ofrece soporte integrado para estructuras de datos profundamente inmutables (véase la propuesta de registros y tuplas para ECMAScript). Immer (“siempre” en Alemán) es un pequeño paquete que nos permite trabajar con estado inmutable de forma

Adoptar
76. Arrow
77. jest-when

Probar
78. Fastify
79. Immer
80. Redux
81. Rust
82. single-spa
83. Strikt
84. XState

Evaluar
85. Babylon.js
86. Blazor
87. Flutter Driver
88. Sentinel
89. Hermes
90. io-ts
91. Kedro
92. LitElement
93. Mock Service Worker
94. Recoil
95. Snorkel
96. Streamlit
97. Svelte
98. SWR
99. Testing Library

Resistir

Lenguajes & Frameworks

Ya no consideramos que Redux sea la solución predeterminada para gestión del estado en las aplicaciones de React. Sigue siendo un marco de trabajo valioso, pero puede conducir a un código más verboso y difícil de seguir que las alternativas.

(Redux)

Es un framework simple en JavaScript y TypeScript para crear máquinas de estado finito y visualizarlas como gráficos de estado.

(XState)

más conveniente. Está basado en el mecanismo de copia en escritura (copy-on-write), tiene una API mínima y opera sobre objetos y arreglos normales de JavaScript. Esto significa que el acceso a los datos es transparente y no se requiere de grandes esfuerzos de refactorización para introducir inmutabilidad en una base de código existente. Muchos de nuestros equipos ya lo utilizan en sus bases de código JavaScript y les gusta más que Immutable.js, razón por la cual lo promovemos al anillo Probar.

Redux
Probar

Hemos decidido regresar a Redux al anillo de Probar para mostrar que ya no es considerada como la solución predeterminada para la gestión del estado en aplicaciones React. Nuestra experiencia muestra que Redux aún es un framework valioso en muchos casos aunque lleva a escribir un código más verboso y difícil de mantener, en comparación con otros enfoques. El añadir Redux Sagas a la mezcla generalmente contribuye a este problema. Como alternativa se pueden utilizar las características incorporadas en las versiones recientes de React para gestionar el estado sin necesidad de frameworks adicionales. No obstante, nos gustaría resaltar que una vez alcanzado el punto en el que la gestión del estado comienza a hacerse compleja, podría ser apropiado usar Redux o tal vez incluso Recoil, publicada recientemente por Facebook.

Rust
Probar

La popularidad del lenguaje de programación Rust sigue aumentando, al punto de haber sido nombrado, por las personas desarrolladoras, como el lenguaje “más querido” en Stack Overflow durante cinco años consecutivos. A nosotros también nos gusta. Es un lenguaje expresivo, rápido y seguro que va aumentando su utilidad a medida que su ecosistema crece. Por ejemplo, Rust se está empezando a utilizar para aplicaciones de ciencia de datos y aprendizaje automático, entregando un aumento significativo del rendimiento. También, Materialize es una base de datos orientada a la transmisión de flujos de datos con baja latencia escrita en Rust.

single-spa
Trial

single-spa es un framework para JavaScript que permite consolidar múltiples *micro-frontends* en una aplicación individual. Aunque advertimos sobre la anarquía de los micro-frontends (el uso de micro-frontends como excusa para mezclar y combinar múltiples frameworks) single-spa soporta precisamente eso. Entendemos que existen casos legítimos tales como la actualización a una nueva versión de un framework a través de múltiples micro-frontends, en los cuales la integración a través de múltiples frameworks es necesaria. Nuestros equipos han considerado a single-spa como la mejor opción para la integración de micro-frontends, y se ha encontrado que trabaja bien con SystemJS y administrando diferentes versiones de una dependencia en particular.

Strikt
Probar

El ecosistema de Kotlin sigue creciendo y más bibliotecas de componentes están aprovechando las características de este lenguaje para reemplazar a sus alternativas basadas en Java. Strikt es una una biblioteca de aserciones que permite escribir comprobaciones en las pruebas con un estilo muy fluido. Utiliza bloques de Kotlin y funciones lambda para ayudar a que las pruebas sean menos verbosas mientras mantienen su legibilidad. Strikt también permite la construcción de aserciones personalizadas para hacer que las pruebas sean más específicas al dominio.

XState
Probar

En ediciones anteriores del Radar, hemos hablado de varias bibliotecas para la gestión del estado, pero XState toma un enfoque ligeramente distinto. Es un framework simple en JavaScript y TypeScript para crear máquinas de estado finito y visualizarlas como gráficos de estado. Se integra con los frameworks reactivos más populares de JavaScript (Vue.js, Ember.js, React.js y RxJS) y está basado en el estándar W3C para máquinas de estado finito. Otra característica notable es la serialización de las definiciones de las máquinas. Algo que nos ha sido útil al crear máquinas de estado finito en otros contextos (particularmente al escribir lógica de juegos) es la habilidad de visualizar los estados y sus posibles transiciones; nos gusta que es realmente sencillo hacerlo con el visualizador de XState.

Babylon.js
Evaluar

Cuando escribimos sobre la realidad virtual más allá de los juegos hace algunos años, no intentamos predecir la velocidad ni el nivel en el que podríamos encontrar soluciones de realidad virtual en otros campos además de los videojuegos. En retrospectiva, hemos visto interés e incremento en la adopción, que sin embargo ha sido más lenta de

lo que algunas personas pensamos. Una de las razones podría tener que ver con las herramientas: Unity y Unreal son dos motores muy maduros y capaces para desarrollar aplicaciones de realidad virtual; también hemos hablado de Godot. Sin embargo, estos motores son bastante diferentes de lo que la mayoría de los equipos de desarrollo web y empresarial conocen. Tras mucho indagar, nos hemos dado cuenta de que las soluciones de realidad virtual basadas en la web han hecho grandes avances y nuestra experiencia con Babylon.js ha sido positiva. Babylon.js está escrito en TypeScript y renderiza sus aplicaciones en el navegador, lo que permite ofrecer una experiencia familiar y conocida para muchos equipos de desarrollo. Además, Babylon.js es un software de código abierto, estable y es un proyecto bien financiado, lo que le da estabilidad en el largo plazo y lo convierte en una opción aún más interesante.

Blazor
Evaluar

Si bien JavaScript y su ecosistema dominan el espacio del desarrollo de interfaces de usuario web, están surgiendo nuevas oportunidades con la llegada de WebAssembly. Blazor nos parece una opción interesante para construir interfaces de usuario web interactivas con C#. Nos gusta especialmente este marco de trabajo de código abierto porque permite ejecutar código C# en el navegador mediante WebAssembly, aprovechando el motor en tiempo de ejecución .NET Standard y su ecosistema, además de librerías personalizadas desarrolladas en ese lenguaje de programación. Además, Blazor permite la interacción bidireccional con código en JavaScript en el navegador si se requiere.

Flutter Driver
Evaluar

Flutter Driver es una biblioteca para pruebas de integración para aplicaciones hechas con Flutter. Con Flutter Driver puedes configurar y manejar las pruebas en dispositivos reales o en emuladores. Nuestros equipos continúan escribiendo pruebas unitarias y pruebas de dispositivos para asegurar que se implementan la mayoría de las funcionalidades de negocio en las aplicaciones de Flutter. Sin embargo, para probar las interacciones reales, como lo harían las usuarias, estamos evaluando Flutter Driver y creemos que también deberías hacerlo.

HashiCorp Sentinel
Evaluar

Si bien somos grandes promotores en lo que respecta a definir políticas de seguridad como código, las herramientas disponibles en este ámbito han sido bastante limitadas. Si estás utilizando productos de HashiCorp (como Terraform o Vault) y no te importa pagar por las versiones empresariales, tienes la opción de usar HashiCorp Sentinel. Sentinel es, de hecho, un lenguaje de programación completo para definir e implementar decisiones sobre políticas basadas en contexto. Por ejemplo, en Terraform puede utilizarse para comprobar violaciones de políticas de seguridad antes de aplicar los cambios de infraestructura. En Vault, Sentinel puede usarse para definir un control de acceso más granular a las APIs. Este enfoque tiene todos los beneficios de encapsulamiento, mantenibilidad, legibilidad y extensibilidad que ofrecen los lenguajes de programación de alto nivel, creando una alternativa atractiva a las políticas de seguridad declarativas tradicionales. Sentinel pertenece a la misma clase de herramientas que Open Policy Agent pero es propietaria, de código cerrado y solo funciona con los productos de HashiCorp.

Hermes
Evaluar

Hermes es un motor de JavaScript optimizado para la inicialización rápida de aplicaciones React Native en Android. Los motores de JavaScript, como V8, tienen compiladores (just in time, JIT) que perfilan el código en tiempo de ejecución para producir instrucciones optimizadas. Hermes, sin embargo, utiliza un enfoque distinto compilando el código JavaScript en bytecode optimizado de manera anticipada (ahead of time, AOT). Como resultado se obtiene un tamaño de imagen APK más pequeño, menos consumo de memoria y tiempos de arranque reducidos. Estamos evaluando a Hermes cuidadosamente en algunas apps de React Native y recomendamos realizar lo mismo para sus desarrollos.

io-ts
Evaluar

Hemos disfrutado mucho usando TypeScript desde hace ya algún tiempo y nos encanta la seguridad que ofrece el tipado fuerte. Sin embargo, introducir datos dentro de los límites impuestos por el sistema de tipos, por ejemplo desde una llamada a un servicio de back-end, puede llevar a obtener errores en tiempo de ejecución. Una biblioteca que ayuda a resolver este problema es io-ts. Cubre el vacío existente entre la comprobación de tipos en tiempo de compilación y el consumo de datos externos en tiempo de ejecución, ofreciendo funciones para

Lenguajes & Frameworks

Sentinel es un lenguaje de programación completo para definir e implementar decisiones sobre políticas.

(HashiCorp Sentinel)

Lenguajes & Frameworks

Kedro es un framework de desarrollo para flujos de trabajo en proyectos de ciencia de datos que trae un enfoque estandarizado para la construcción de pipelines de datos y para aprendizaje automático.

(Kedro)

codificar y decodificar datos. También puede usarse como una protección personalizada de tipos. Según nuestros equipos, es una solución elegante a un problema bastante escurridizo.

Kedro
Evaluar

En el pasado hemos hablado sobre el perfeccionamiento de las herramientas que permiten aplicar buenas prácticas de ingeniería en proyectos de ciencia de datos. Kedro es otra buena adición a este espacio. Es un *framework* de desarrollo para flujos de trabajo en proyectos de ciencia de datos que trae un enfoque estandarizado para la construcción de pipelines de datos y para aprendizaje automático listos para producción. Nos gusta su enfoque en las prácticas de ingeniería de software y en el buen diseño, con énfasis en el desarrollo dirigido por pruebas (TDD), modularidad, versionamiento y prácticas de buena higiene como lo es el mantener las credenciales fuera de la base de código.

LitElement
Evaluar

Se ha hecho un progreso constante desde la primera vez que escribimos sobre Componentes Web en el 2014. LitElement, que forma parte del Polymer Project, es una biblioteca simple que se puede usar para crear componentes web ligeros. En realidad, es solo una clase base que elimina la necesidad de escribir gran parte del código repetitivo más común, lo que simplifica el desarrollo de componentes web. Hemos tenido éxito relativamente rápido al usarlo en proyectos y estamos emocionados de ver madurar la tecnología.

Mock Service Worker
Evaluar

Las aplicaciones web, especialmente las que son creadas para uso interno en las empresas, suelen ser construidas en dos partes. La primera, que se ejecuta en el navegador, corresponde a la interfaz de usuario y a alguna lógica de negocio. La segunda, que corre en el servidor, corresponde al resto de la lógica de negocio, la autenticación y persistencia de datos. Normalmente, ambas partes se comunican utilizando JSON sobre HTTP. Los *endpoints* no deberían confundirse por una API real ya que son apenas un detalle de implementación de una aplicación que está dividida en dos ambientes. Al mismo tiempo, ofrecen un punto de unión a través del cual se puede probar cada parte de manera individual. Cuando se prueba el código JavaScript, el servidor puede ser simulado a nivel de red mediante herramientas como Mountebank. Un método alternativo es interceptar las peticiones en el navegador. Nos gusta lo que propone Mock Service Worker porque con los *service workers* se provee una abstracción que es familiar para las personas desarrolladoras. Este enfoque permite una configuración más simple y mayor rapidez en la ejecución de las pruebas. Sin embargo, como estas pruebas no validan el funcionamiento de la capa de red, sugerimos implementar algunas pruebas de punta a punta (*end-to-end*) como parte de una pirámide de pruebas saludable.

Recoil
Evaluar

Cada vez más equipos que utilizan React están reevaluando sus opciones para el manejo del estado, algo que también mencionamos en nuestra reevaluación de Redux. Facebook (creadora de React) ha publicado recientemente Recoil, un nuevo framework para el manejo del estado que surgió de una aplicación interna que tenía que tratar con grandes cantidades de datos. A pesar que actualmente no tenemos mucha experiencia práctica con Recoil, vemos su potencial y promesa. Su API es sencilla y fácil de aprender y se siente como React idiomático. A diferencia de otros enfoques, Recoil proporciona una forma eficiente y flexible de tener un estado compartido dentro de una aplicación: soporta estados creados dinámicamente mediante consultas y datos derivados así como la observación del estado en toda la aplicación sin afectar la separación del código.

Snorkel
Evaluar

Los modelos modernos de ML son muy complejos y requieren de gran cantidad de conjuntos de datos para entrenamiento etiquetados para que puedan aprender de ellos. Snorkel comenzó en el laboratorio de IA de Stanford con la consideración de que etiquetar datos manualmente es muy costoso y, muchas veces, no es factible. Snorkel nos permite etiquetar datos de entrenamiento de forma programada a través de la creación de funciones de etiquetado. Snorkel utiliza técnicas de aprendizaje supervisadas para evaluar la precisión y correlación entre estas funciones de etiquetado y luego las vuelve a examinar y combina sus etiquetas de salida, produciendo como resultado etiquetas de entrenamiento de alta calidad. Desde entonces, los creadores de Snorkel han creado una plataforma comercial llamada Snorkel Flow. Si bien Snorkel ya no se desarrolla

activamente, sigue siendo importante por sus ideas sobre el uso de métodos poco supervisados para el etiquetado de datos.

Streamlit
Evaluar

Streamlit es un *framework* de código abierto para Python utilizado por especialistas en ciencia de datos para construir aplicaciones de visualización de datos que tengan un excelente aspecto. Streamlit destaca sobre competidores como Dash por su foco en el prototipado rápido y por el soporte a una amplia variedad de bibliotecas de visualización, como por ejemplo Plotly y Bokeh. Esta herramienta es una excelente alternativa si se necesita realizar presentaciones rápidas durante el ciclo de experimentación. En ThoughtWorks la estamos utilizando en algunos proyectos y nos encanta la manera en que podemos construir visualizaciones interactivas con muy poco esfuerzo.

Svelte
Evaluar

Siguen apareciendo nuevos frameworks para el desarrollo de interfaz de usuario y Svelte se perfila como un prometedor nuevo framework basado en componentes. A diferencia de otros frameworks que aprovechan el DOM virtual, Svelte compila el código a JavaScript estándar que actualiza quirúrgicamente el DOM directamente. No obstante, Svelte es simplemente un framework basado en componentes; si el plan es construir una aplicación con muchas funcionalidades, considera utilizarlo en conjunto con Sapper.

SWR
Evaluar

SWR es una biblioteca de React Hooks para obtener datos remotos. Implementa la estrategia para el almacenamiento de datos en caché de HTTP denominada *stale-while-revalidate*. SWR primero devuelve los datos desde la caché (obsoletos), luego envía la petición para obtener información (revalidar) y finalmente actualiza los valores a partir de la respuesta actualizada. Los componentes reciben un flujo de datos, primero obsoletos, luego actualizados, de forma constante y automática. Nuestros equipos de desarrollo han tenido buenas experiencias utilizando SWR, mejorando de forma dramática la experiencia de uso al tener siempre datos en la pantalla. Sin embargo, advertimos a los equipos para que sólo utilicen la estrategia de almacenamiento en caché de SWR cuando sea aceptable que la aplicación devuelva datos obsoletos. Ten en cuenta que el protocolo HTTP exige que los cachés respondan a una petición con la respuesta más actualizada que sea apropiada para la petición, y sólo en circunstancias cuidadosamente estudiadas se permite la devolución de una respuesta obsoleta.

Testing Library
Evaluar

Testing Library es una familia de paquetes para probar aplicaciones en numerosos frameworks como React, Vue, React Native y Angular, entre otros. Estas bibliotecas ayudan a probar los componentes de interfaz de usuario desde la perspectiva del usuario, animando a probar su comportamiento en lugar de los detalles de implementación, como la presencia de elementos en la interfaz de usuario en un momento determinado. Uno de los beneficios de este enfoque es que las pruebas son más confiables, y estamos seguros que esto es su principal diferenciador. Recomendamos evaluar esta familia de bibliotecas para probar las aplicaciones web con cualquier *framework*. Aunque nuestra experiencia directa se limita a React Testing Library y a Angular Testing Library, en general, nos ha impresionado lo que hemos visto.

Lenguajes & Frameworks

Creado en la Universidad de Stanford, Snorkel nos permite etiquetar programáticamente los conjuntos de datos masivos que se utilizan para entrenar los algoritmos de aprendizaje de las máquinas.

(Snorkel)

ThoughtWorks®

Somos una consultora de software global y una comunidad de individuos apasionados guiados por propósitos, 7,000+ 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

Coordinadoras de traducción: María José Lalama, Natalia Rivera y Paula Marín

Traductores: Abel Guillén, Aldemaro Díaz, Alexandra Granda, Ana Silvia Telleria Martínez, Angel Garbayo, Araceli Correa, Bárbara Deluchi, Byron Torres, Carlos Medina, Cecilia Barudi, Cecilia Geraldo, Cintya Aguirre, Daniel Santibañez, Daniela Mora, David Montaño, David Corrales, David Salazar, Diana Suasnavas, Eduard Maura i Puig, Eduardo Winpenny Tejedor, Jesús Cardenal, Jorge Agudo Praena, José Herdoíza, Jose Puebla, Julio López Guzmán, María Fernanda Escudero, Marcelo Morales Padilla, Marcos Mercuri, María Córdova, María Fernanda Yépez, Milber Champutiz, Pamela Guamán, Paola Cajilema, Reynier Pupo, Ricardo Rodriguez, Rosa Palli, Sergio Delgado, Victor Cosqui, Yago Pereiro Estevan y Yeraldine Mendoza.

Editores técnicos: Carlos Oquendo y Fausto De La Torre

ThoughtWorks®

thoughtworks.com/es/radar

#TWTechRadar