Volume
30

Abril 2024

Radar Tecnológico

Una guía de opinión sobre el entorno tecnológico actual

/thoughtworks

Strategy. Design. Engineering.

Sobre el Radar

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

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

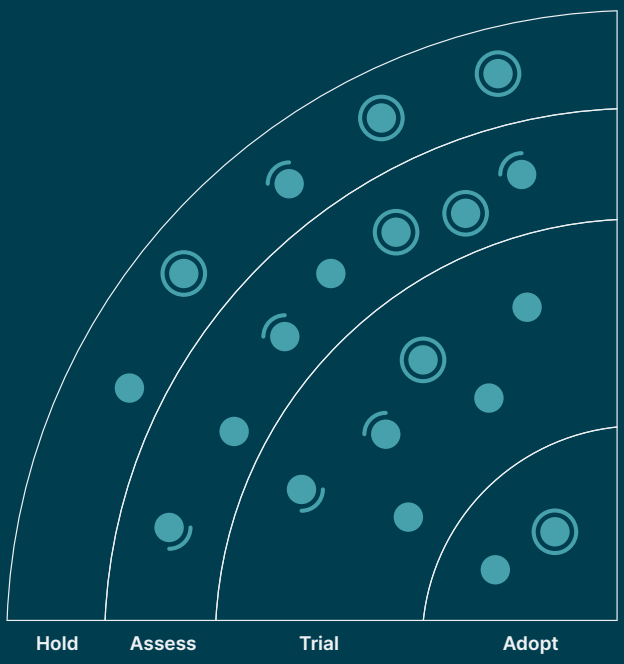
Te animamos a explorar estas tecnologías. El Radar es de naturaleza gráfica y agrupa los elementos en técnicas, herramientas, plataformas y lenguajes y frameworks. Cuando los elementos del Radar podían aparecer en varios cuadrantes, elegimos el que nos pareció más apropiado. Además, agrupamos estos elementos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

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

Un vistazo al Radar

El Radar se dedica a rastrear cosas interesantes, a las que nos referimos como blips. Organizamos los blips en el Radar utilizando dos elementos de categorización: cuadrantes y anillos. Los cuadrantes representan los diferentes tipos de blips. Los anillos indican nuestra recomendación para utilizar esa tecnología.

Un blip es una tecnología o técnica que desempeña un papel en el desarrollo de software. Los blips son cosas que están en movimiento", es decir, que su posición en el Radar cambia a menudo, lo que suele indicar nuestra creciente confianza en recomendarlos a medida que avanzan por los anillos..



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

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

Evaluar: Vale la pena explorar con el objetivode comprender cómo afectará a su empresa.e.

Resistir: Proceder con precaución..

Nuevo **Deplazado adentro /afuera** **Ningún cambio**

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

Contribuyentes

El Technology Advisory Board (TAB) es un grupo de 22 personas tecnólogas senior de Thoughtworks. El TAB se reúne dos veces al año en persona y virtualmente cada dos semanas. Su función principal es ser un grupo de asesoramiento para el CTO de Thoughtworks, Rachel Laycock, y CTO Emerita, Rebecca Parsons.

El TAB actúa como un organismo amplio que puede examinar los temas que afectan a la tecnología y a las personas tecnólogas en Thoughtworks. Esta edición del Radar Tecnológico de Thoughtworks es en base a la reunión del TAB en Nueva York en febrero de 2024.


Rebecca Parsons (CTO Emerita)


Rachel Laycock (CTO)


Martin Fowler (Chief Scientist)


Bharani Subramaniam


Birgitta Böckeler


Brandon Byars


Camilla Falconi Crispim


Erik Dörnenburg


Fausto de la Torre


Hao Xu


James Lewis


Marisa Hoenig


Maya Ormaza


Mike Mason

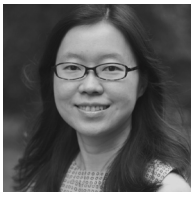

Neal Ford


Pawan Shah


Scott Shaw


Selvakumar Natesan


Shangqi Liu


Sofia Tania


Vanya Seth


Will Amaral

Contribuyentes

Equipo de traducción:

Catalina Margozzini, Tex Albuja, Neysa Alarcón, Milber Champutiz, Israel Pasaca, Erika Vacacela, Paola Cajilema, Anna Corral, Ulises Harris, Miguel Acevedo, Ana María Zúñiga, Arnaldo Garat, Antonia Castells, Eduard Maura i Puig, Angelina Ziran, Jhosep Marin, Jorge Palacios, Cristian Montero, Santiago Fuentes, Francisco Trujillo, Andrés Rojas, Gabriela Marques, Daniel Santibañez, Jennifer Guayta, Eddin Ramírez , Jose Fernandez, Joaquin Hervas, Esteban Mata, Gabriel Frias Rivas, Gisela Yumi, José Herdoíza, Carlos Barroso, Ana Lallena Arquillo, Zully Arellano, Jose María Alonso Montero, Elvira Segarra Ruiz, Enrique Aracil Mas, Irene Torres, Ana López Estrella, Monica Giraldo Rivera, Isabel Hong, Galo Becerra, Esteban Chacon, Araceli Correa, Judit Navarro, Juan Romero, Andrea Astorga, Diana Pila, Carlos Fuentes, Carolina Melgarejo Pezoa, Jesús Cardenal, Kelly Castro, Jorge Domínguez, Naileth Rivero, Juan Pablo Jorquera, Sendami Luque Liñan, Jorge Agudo Praena, Eumir Ollarvez y Nahuel Delgado.

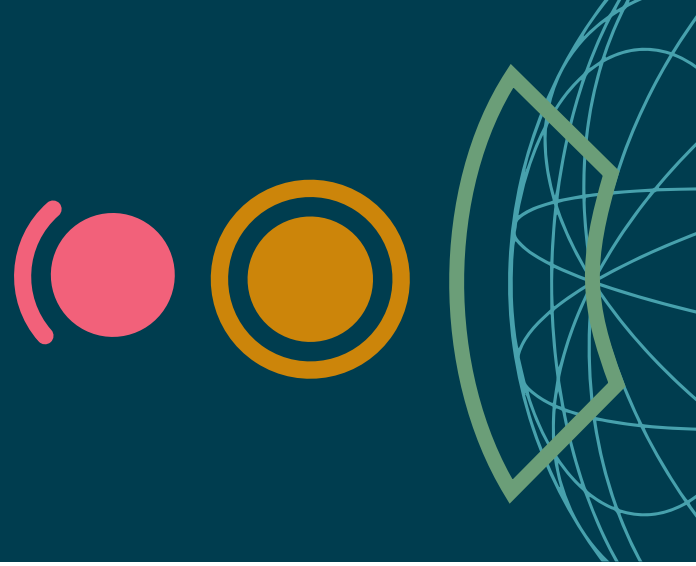
Equipo de Edición:

Ruben Trujillo, Carlos Oquendo, Fernando Tamayo, Octavio Valenzuela Beltrán e Ignacio Gutierrez

Equipo de Marketing:

Elizabeth Parra.

Temas



Licencias de código (casi) abierto

En la preparación de esta edición, tuvimos dos tipos de debate sobre licencias. En primer lugar, durante muchos años, el ecosistema de desarrollo de software de código abierto se apoyó en un conjunto de licencias, catalogadas por la Open Source Initiative (OSI), siendo un pequeño número de licencias populares las preferidas en la mayoría de los casos. Sin embargo, en los últimos tiempos hemos visto algo de movimiento en este panorama antes sereno. Varias herramientas destacadas han tenido mala prensa últimamente, cuando sus mantenedores cambiaron, en varios casos abruptamente, de una licencia de código abierto a un modelo comercial. No tenemos problemas con pagar por el software, y nos parece razonable usar licencias comerciales para funcionalidades adicionales. Sin embargo, consideramos problemático cuando la funcionalidad básica de una herramienta muy utilizada se coloca detrás de un muro de pago, especialmente cuando se ha desarrollado un ecosistema alrededor. En segundo lugar, otra situación interesante tiene que ver con el software que proclama ser de código abierto pero cuyas capacidades fundamentales sólo aparecen luego que los consumidores pagan suscripciones u otros cargos. Si bien este modelo de negocio ha existido antes, parece que se explota más con muchas de las nuevas y relucientes herramientas de IA, que ofrecen capacidades asombrosas un poco escondidas bajo la letra pequeña. Aconsejamos revisar con detenimiento las cuestiones de licenciamiento. Presta atención a las salvedades y asegúrate de que todos los archivos de un repositorio están cubiertos por la licencia del nivel superior.

Equipos de desarrollo de software asistidos por IA

El tema de la IA obviamente dominó nuestras conversaciones; un tercio de nuestros puntos del Radar se referían a algún aspecto de ella. Mientras discutíamos varias herramientas de IA centradas en los desarrolladores como GitHub Copilot, CodiumAI, aider y Continuar, también tuvimos numerosas conversaciones sobre cómo el uso holístico de la IA en todo un equipo cambia aspectos del desarrollo de software. Hablamos sobre una variedad de herramientas que no llegaron a la versión final, incluyendo terminales asistidas por IA como Warp, la capacidad de convertir capturas de pantalla en código, ChatOps respaldado por LLMs y muchos otros temas. Aunque las herramientas de desarrollo tienden a alcanzar un mayor grado de madurez, sospechamos que todos los aspectos del desarrollo de software pueden beneficiarse gradualmente del uso pragmático de la IA y las herramientas derivadas, y estamos siguiendo activamente las innovaciones en el panorama del desarrollo. Por

supuesto, las nuevas capacidades casi mágicas que ofrece la IA conllevan nuevos riesgos para la calidad y la seguridad del software. Esto requiere que los equipos permanezcan atentos, incluso manteniendo informados a los no desarrolladores sobre los peligros potenciales.

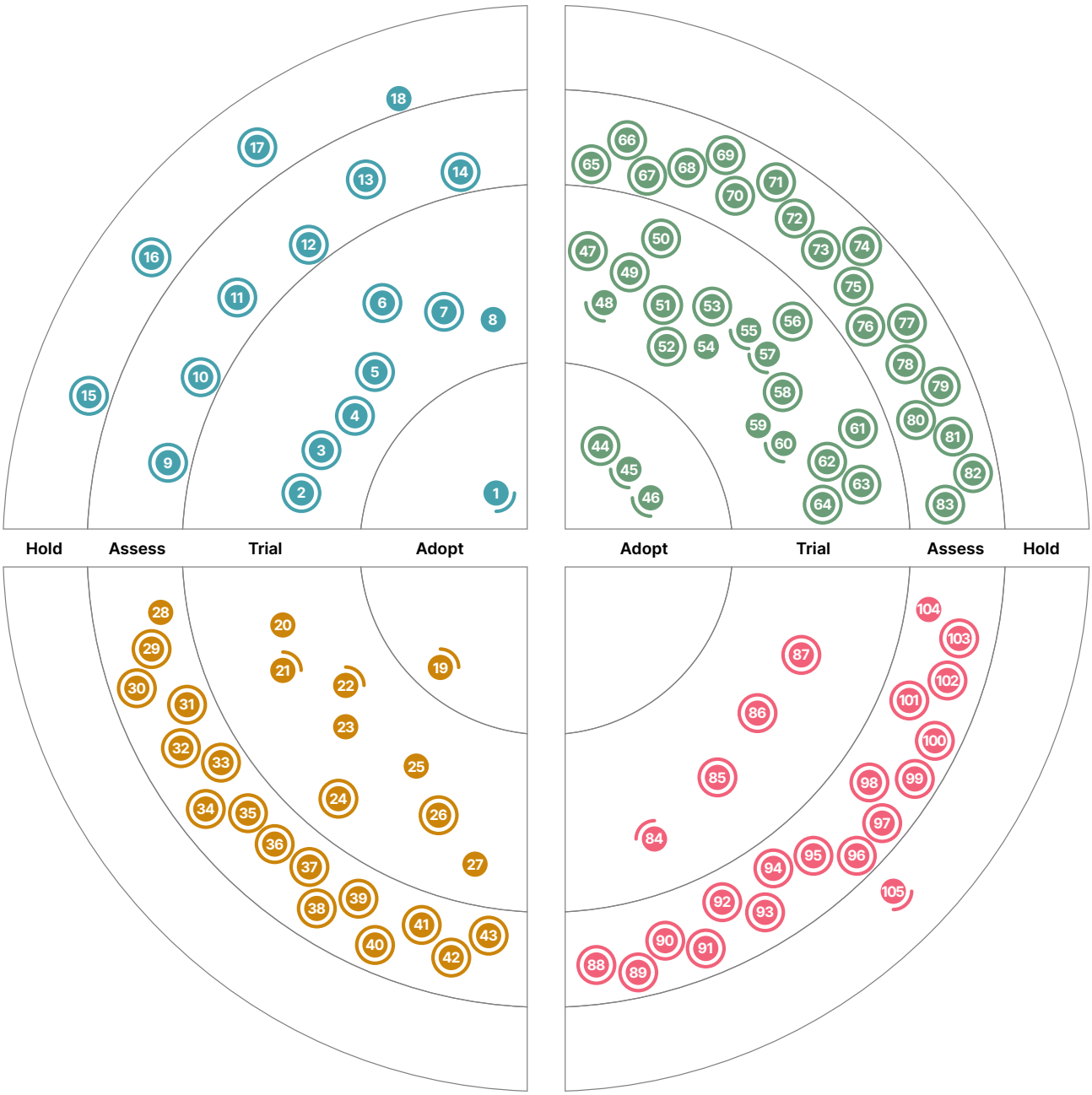
Patrones de arquitectura emergentes para LLM

Los patrones son populares en el mundo de la tecnología porque proporcionan un nombre conciso para una solución a un problema particular. Con el creciente uso de los modelos grandes de lenguajes (Large Language Models, LLM), empezamos a ver el surgimiento de patrones de arquitectura específicos para apoyar contextos comunes. Por ejemplo, hemos hablado de NeMo Guardrails, que permite a las personas desarrolladoras construir políticas de gobernanza sobre la utilización de LLM. También hemos hablado de herramientas tal y como Langfuse que permite mayor observabilidad en los pasos que conducen a las salidas de un LLM y cómo lidiar con (y validar) bases de código excesivamente llenas de código generado. Hemos discutido cómo la retrieval-augmented generation (RAG) es nuestro patrón preferido para mejorar la calidad de las salidas de un LLM, especialmente en el ecosistema empresarial. Adicionalmente, hemos discutido técnicas como por ejemplo utilizar un LLM menos potente (y de menor coste) para producir material que es revisado de forma selectiva por un LLM más potente (y caro). Los patrones forman un vocabulario importante para las tecnologías y esperamos ver una explosión de patrones (y de los inevitables anti-patrones) conforme la IA generativa continúe extendiéndose en el desarrollo de software.

Acercando los PRs a la CI real

Thoughtworks siempre ha sido una firme defensora de los ciclos de retroalimentación rápida durante el desarrollo de software y, por eso mismo, una gran partidaria de la integración continua (CI, del inglés: Continuous Integration). Para apoyar la adopción, construimos el primer servidor de CI — CruiseControl — que fue convertido en código abierto a finales de los años 90. Recientemente, nuestro director científico Martin Fowler actualizó la definición canónica de integración continua en su bliki para volver a poner el foco sobre esta práctica. Sin embargo, muchos de nuestros equipos se ven obligados a ignorar la parte CI de CI/CD porque se encuentran en situaciones en las que las pull requests (PRs) son obligatorias. Aunque la práctica con PRs se desarrolló originalmente para gestionar equipos de código abierto ampliamente distribuidos y contribuyentes poco confiables, se han convertido en un sinónimo de revisión por pares (del inglés: peer review) habitual incluso en equipos de desarrollo pequeños y muy cohesionados. En estas circunstancias, muchos desarrolladores anhelan la misma sensación de fluidez que obtienen al practicar CI real. Examinamos varias herramientas que intentan aliviar los problemas de los procesos de revisión de PRs, incluidas gitStream y Github merge queue. También discutimos técnicas como stacked diffs que prometen alinearse con los principios básicos de la CI al permitir un control más granular sobre el proceso de integración y discutimos métodos de métricas derivadas de PRs para identificar ineficiencias y cuellos de botella durante la entrega de software. Estas herramientas son de gran ayuda en este espacio debido a la tendencia hacia la IA generativa para el desarrollo de software. Con los asistentes de IA en la programación, el rendimiento del desarrollo aumenta, lo que lleva a una tendencia a crear PRs más grandes. Esto ejerce aún más presión sobre los procesos de revisión de código asíncrono. Aunque todavía preferimos la práctica original de CI, alentamos a los equipos que no pueden utilizarla debido a limitaciones externas a encontrar formas de mejorar la precisión de la integración y la velocidad de su ciclo de retroalimentación.

El Radar



Nuevo

Desplazado adentro/afuera

Ningún cambio

El Radar

Técnicas

Adoptar

1. Generación mejorada por recuperación (RAG)

Probar

2. Generación automática de descriptores de entidades de Backstage
3. NLPs tradicionales combinados con LLMs
4. Cumplimiento continuo
5. Edge functions
6. Campeones de la seguridad
7. Texto a SQL
8. Seguimiento de la salud sobre la deuda técnica

Evaluar

9. Asistentes de equipo con IA
10. Análisis gráfico para LLM-Backed chats
11. ChatOps impulsada por LLMs
12. Agentes autónomos impulsados por LLMs
13. Usar GenAI para entender las bases de código heredado
14. VISS

Resistir

15. Amplias pruebas de integración
16. Uso excesivamente entusiasta de LLM
17. Prisa por afinar LLMs
18. Componentes Web para aplicaciones renderizadas en el servidor (SSR)

Plataformas

Adoptar

19. CloudEvents

Probar

20. Arm en la nube
21. Azure Container Apps
22. Azure OpenAI Service
23. DataHub
24. Plataformas de orquestación de infraestructura
25. Pulumi
26. Rancher Desktop
27. Weights & Biases

Evaluar

28. Bun
29. Chronosphere
30. DataOS
31. Dify
32. Elasticsearch Relevance Engine
33. FOCUS
34. Gemini Nano
35. HyperDX
36. IcePanel
37. Langfuse
38. Qdrant
39. RISC-V for embedded
40. Tigerbeetle
41. WebTransport
42. Zarf
43. ZITADEL

Resistir

—

Herramientas

Adoptar

44. Conan
45. Kaniko
46. Karpenter

Probar

47. 42Crunch API Conformance Scan
48. actions-runner-controller
49. Android Emulator Container
50. AWS CUDOS
51. aws-nuke
52. Bruno
53. Develocity
54. GitHub Copilot
55. Gradio
56. Catálogos de versiones de Gradle
57. Maestro
58. Herramienta SBOM de Microsoft
59. Open Policy Agent (OPA)
60. Ejecutores de GitHub auto-alojados de Philips
61. Pop
62. Renovate
63. Terrascan
64. Velero

Evaluar

65. aider
66. Akvorado
67. Baichuan 2
68. Cargo Lambda
69. Codium AI
70. Continue
71. Fern Docs
72. Granted
73. LinearB
74. LLaVA
75. Marimo
76. Mixtral
77. NeMo Guardrails
78. Ollama
79. OpenTofu
80. QAnything
81. System Initiative
82. Tetragon
83. Winglang

Resistir

—

Lenguajes y Frameworks

Adoptar

—

Probar

84. Astro
85. DataComPy
86. Pinia
87. Ray

Evaluar

88. Android Adaptability
89. Concrete ML
90. Crabviz
91. Crux
92. Databricks Asset Bundles
93. Electric
94. LiteLLM
95. LLaMA-Factory
96. MLX
97. Mojo
98. Otter
99. Pkl
100. Rust para el desarrollo de UI
101. vLLM
102. Voyager
103. WGPU
104. Zig

Resistir

105. LangChain

Técnicas

Adoptar

1. Generación mejorada por recuperación (RAG)

Probar

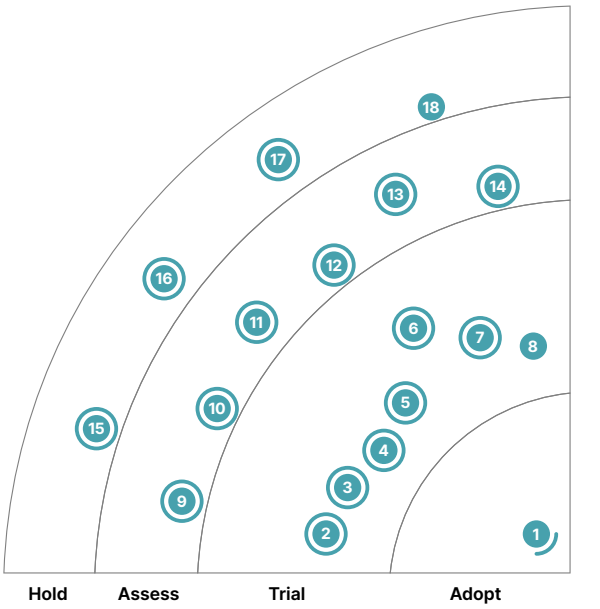
2. Generación automática de descriptores de entidades de Backstage
3. NLPs tradicionales combinados con LLMs
4. Cumplimiento continuo
5. Edge functions
6. Campeones de la seguridad
7. Texto a SQL
8. Seguimiento de la salud sobre la deuda técnica

Evaluar

9. Asistentes de equipo con IA
10. Análisis gráfico para LLM-Backed chats
11. ChatOps impulsada por LLMs
12. Agentes autónomos impulsados por LLMs
13. Usar GenAI para entender las bases de código heredado
14. VISS

Resistir

15. Amplias pruebas de integración
16. Uso excesivamente entusiasta de LLM
17. Prisa por afinar LLMs
18. Componentes Web para aplicaciones renderizadas en el servidor (SSR)



Nuevo

Desplazado dentro/afuera

Ningún cambio

1. Generación mejorada por recuperación (RAG)

Adoptar

Generación mejorada por recuperación (RAG por sus siglas en inglés) ies el patrón preferido por nuestros equipos para mejorar la calidad de las respuestas generadas por un modelo lingüístico grande (LLM por sus siglas en inglés). Lo hemos utilizado con éxito en varios proyectos, incluyendo el popular Jugalbandi AI Platform. Con RAG, la información sobre documentos relevantes y fiables -en formatos como HTML y PDF- se almacena en una base de datos que admita un tipo de datos vectorial o una búsqueda eficiente de documentos, como pgvector, Qdrant o Elasticsearch Relevance Engine. Para una consulta determinada, la base de datos se consulta para recuperar documentos relevantes, que luego se combinan con la consulta para proporcionar un contexto más enriquecido al LLM. De este modo se obtienen resultados de mayor calidad y se reducen considerablemente las alucinaciones. La ventana de contexto -que determina el tamaño máximo de entrada del LLM- es limitada, lo que significa que seleccionar los documentos más relevantes es crucial. Mejoramos la relevancia del contenido que se añade a la consulta mediante re-ranking. Del mismo modo, los documentos suelen ser demasiado grandes para calcular una incrustación, lo que significa que deben dividirse en fragmentos más pequeños. Suele ser un problema difícil, y una solución es que los fragmentos se solapen hasta cierto punto.

2. Generación automática de descriptores de entidades de Backstage

Probar

Backstage de Spotify ha sido ampliamente adoptado por nuestra base de clientes como la plataforma preferida para alojar portales de experiencia de desarrolladores. Backstage, por sí solo, es un cascarón que permite almacenar plugins y que provee una interfaz para manejar el catálogo de recursos que conforman el ecosistema de una plataforma. Cualquier entidad expuesta o manejada por Backstage está configurada en el archivo de información del catálogo, el cual contiene información tal como estado, ciclo de vida, dependencias y APIs, entre otros detalles. Por defecto, los descriptores de entidades individuales son escritos a mano y usualmente mantenidos y versionados por el equipo responsable del componente en cuestión. Mantener los descriptores actualizados puede ser tedioso y crea una barrera a la adopción por parte de los desarrolladores. También, siempre está la posibilidad de que los cambios sean pasados por alto, o que algunos componentes sean completamente olvidados. Encontramos que **generar automáticamente los descriptores de entidades de Backstage** es más eficiente y menos propenso a errores. La mayoría de las organizaciones tienen fuentes existentes de información que pueden dar el primer empujón al proceso de generar las entradas del catálogo. Buenas prácticas de desarrollo, por ejemplo, usando tags apropiadas o agregando metadata a archivos fuente, pueden simplificar el descubrimiento de entidades y la generación de descriptores. Estos procesos automatizados pueden ejecutarse regularmente, por ejemplo una vez al día, para mantener el catálogo actualizado.

3. NLPs tradicionales combinados con LLMs

Probar

Los modelos de lenguaje grandes (LLMs: Large Language Models en inglés) son la navaja suiza del Procesamiento del Lenguaje Natural (PLN, del inglés NLP: Natural Language Processing). Sin embargo, también son bastante costosos y no siempre son la mejor herramienta para tus tareas — a veces es más efectivo usar simplemente un sacacorchos. De hecho, hay mucho potencial en **combinar el NLP tradicional con los LLMs** , o construir múltiples estrategias NLP en conjunto con LLMs para implementar casos de uso y aprovechar los LLMs para los pasos en los que realmente

necesitamos sus capacidades. Es menos costoso, y puede ser más efectivo para solventar parte de tu caso de uso, usar métodos tradicionales de ciencias de datos y NLP para agrupar documentos (clustering), identificar temas y clasificar, e incluso resumir. Cuando necesitamos generar y resumir textos más largos, o combinar varios documentos grandes, es cuando usamos LLMs para aprovechar la capacidad de atención y memoria superiores de éstos. Por ejemplo, hemos usado exitosamente esta combinación de técnicas para generar un informe exhaustivo de tendencias de un dominio. En este caso, hemos partido de una recopilación de una enorme cantidad de documentos de tendencias individuales y hemos usado el clustering tradicional junto con el poder generativo de los LLMs.

4. Cumplimiento continuo

Probar

Cumplimiento continuo es la práctica de garantizar que los procesos y tecnologías de desarrollo de software cumplan con las regulaciones de la industria y los estándares de seguridad de manera continua, aprovechando en gran medida la automatización. Verificación manual de las vulnerabilidades de seguridad y el cumplimiento de las regulaciones pueden ralentizar el desarrollo e introducir errores. Como una alternativa, las organizaciones pueden automatizar las verificaciones y auditorías de cumplimiento. Pueden integrar herramientas en los pipelines de desarrollo de software, permitiendo a los equipos detectar y abordar los problemas de cumplimiento al inicio del proceso de desarrollo. Codificar las reglas de cumplimiento y las mejores prácticas ayuda a reforzar las políticas y los estándares uniformente en los equipos. Esto permite analizar los cambios de código en busca de vulnerabilidades, aplicar estándares de codificación y rastrear los cambios en la configuración de la infraestructura para garantizar que cumplan con los requisitos de cumplimiento. Por último, los informes automatizados de lo mencionado anteriormente simplifican las auditorías y proporcionan evidencia clara del cumplimiento. Hemos conversado acerca de técnicas como publicación SBOMs y aplicación de recomendaciones de SLSA — estos son muy buenos puntos de partida. Los beneficios de esta técnica son múltiples. En primer lugar, la automatización promueve un software más seguro al identificar y mitigar las vulnerabilidades de manera temprana y, en segundo lugar, los ciclos de desarrollo se aceleran a medida que se eliminan las tareas manuales. Reducción de costos y mayor consistencia son ventajas adicionales. Para industrias críticas como vehículos autónomos, el cumplimiento continuo automatizado puede mejorar la eficiencia y la confiabilidad del proceso de certificación, lo que en última instancia conduce a vehículos más seguros y confiables en las carreteras.

5. Edge functions

Probar

Aunque no es un concepto nuevo, hemos observado la creciente disponibilidad y uso de la ejecución descentralizada de código a través de redes de distribución de contenidos (CDNs). Servicios como Cloudflare Workers o Amazon CloudFront Edge Functions proveen mecanismos para ejecutar fragmentos de código serverless cerca de la ubicación geográfica del cliente. Las **Edge functions** no solo ofrecen una latencia más baja si se puede generar una respuesta en el borde, sino que también ofrecen la oportunidad de re-escribir peticiones y respuestas dependiendo de la ubicación del servidor regional. Por ejemplo, puede reescribir una URL de petición para dirigirla a un servidor específico que tenga datos locales relevantes para un campo encontrado en el cuerpo de la petición. Este enfoque es el más adecuado para procesos cortos, sin estado y de ejecución rápida debido a que el poder computacional en el borde es limitado.

6. Campeones de la seguridad

Probar

Security champions son miembros del equipo que reflexionan de forma crítica sobre las repercusiones en la seguridad de las decisiones de entrega, tanto técnicas como no técnicas. Plantean estas preguntas y preocupaciones a los líderes de equipo y conocen a fondo las directrices y requisitos básicos de seguridad. Ayudan a los equipos de desarrollo a enfocar todas las actividades durante la entrega del software con una mentalidad de seguridad, reduciendo así los riesgos generales para la seguridad de los sistemas que desarrollan. Security champion no es un puesto aparte, sino una responsabilidad asignada a un miembro del equipo que recibe la formación adecuada de profesionales de la seguridad. Equipados con esta formación, los security champions mejoran la concienciación sobre seguridad del equipo difundiendo conocimientos y actuando como puente entre los equipos de desarrollo y seguridad. Un gran ejemplo de actividad de los security champions es que pueden ayudar a impulsar dentro del equipo el modelado de amenazas, que ayuda a los equipos a pensar en los riesgos de seguridad desde el principio. Designar y formar a un security champion en un equipo es un gran primer paso, pero confiar únicamente en los security champions sin el compromiso adecuado de los líderes puede acarrear problemas. Según nuestra experiencia, crear una mentalidad de seguridad requiere el compromiso de todo el equipo y de los directivos.

7. Texto a SQL

Probar

Texto a SQL es una técnica que convierte consultas escritas en lenguaje natural a consultas SQL que pueden ser ejecutadas en una base de datos. Aunque los grandes modelos de lenguaje (LLM, large language models) pueden entender y transformar el lenguaje natural, la creación de consultas SQL precisas para un esquema propio puede convertirse en una tarea desafiante. Aquí entra Vanna, un marco de trabajo de generación aumentada por recuperación (RAG, retrieval-augmented generation) escrito en Python, de código abierto, para la generación de SQL. Vanna opera en dos pasos: primero se debe crear embeddings usando sentencias del lenguaje de definición de datos (DDL) y consultas SQL de ejemplo para ese esquema, y luego se puede hacer preguntas en lenguaje natural. Aunque Vanna puede funcionar con cualquier LLM, recomendamos evaluar NSQL, un LLM específico de dominio para tareas de conversión de texto a SQL.

8. Seguimiento de la salud sobre la deuda técnica

Probar

Seguimos observando las mejoras de los equipos en sus ecosistemas al tratar el índice de salud de la misma forma que los otros objetivos de nivel de servicio (SLOs) y priorizando mejoras en consecuencia, en vez de solamente enfocarse en hacer seguimiento de la deuda técnica. Al asignar recursos eficientemente para abordar los problemas con mayor impacto en la salud, equipos y organizaciones pueden reducir costos de mantenimiento a largo plazo y evolucionar productos de manera más eficiente. Este enfoque también mejora la comunicación entre stakeholders tanto técnicos como no técnicos, fomentando un entendimiento común del estado del sistema. A pesar de que las métricas pueden variar entre organizaciones (Ver artículo para ejemplos) al final contribuyen a una sostenibilidad a largo plazo y se aseguran de que el software se mantenga adaptable y competitivo. En un panorama digital que cambia rápidamente, enfocarse en el **seguimiento de la salud sobre la deuda técnica** de los sistemas proporciona una estrategia estructurada y basada en evidencias para mantenerlos y mejorarlos.

9. Asistentes de equipo con IA

Evaluar

Actualmente, se habla principalmente de herramientas de asistencia de codificación IA como GitHub Copilot en el contexto de ayudar y mejorar el trabajo individual. Sin embargo, la entrega de software es y seguirá siendo un trabajo en equipo, por lo que deberías buscar formas de crear **asistentes de equipo con IA** para ayudar a formar elequipo 10x, en lugar de un montón de silos de ingenieros 10x. asistidos por IA. Hemos comenzado a utilizar un enfoque de asistencia en equipo que puede incrementar la amplificación del conocimiento, el perfeccionamiento de habilidades y la alineación mediante una combinación de prompts y fuentes de conocimiento. Los prompts estandarizados facilitan la aplicación de las mejores prácticas acordadas dentro del contexto de equipo, tales como técnicas y plantillas para la redacción de historias de usuario o la implementación de prácticas como el modelado de amenazas. Además de los prompts, las fuentes de conocimiento disponibilizadas a través de la generación aumentada por recuperación proporcionan información contextualmente relevante proveniente de directrices organizacionales o bases de conocimiento específicas de la industria. Este enfoque otorga a los miembros del equipo acceso al conocimiento y los recursos que necesitan justo a tiempo.

10. Análisis gráfico para LLM-Backed chats

Evaluar

Chatbots producidos por large language models (LLMs) están obteniendo gran popularidad hoy en día, y estamos observando técnicas emergentes alrededor de su uso en producción. Uno de los desafíos para esto es entender cómo los usuarios están interactuando con un chatbot el cual es manejado por un ente genérico como un LLM, donde la conversación puede ir en distintas direcciones. Entender la realidad de los flujos de conversación es crucial para perfeccionar el producto y mejorar los rangos de conversión. Una técnica para superar este problema es utilizar **análisis gráfico para LLM-Backed chats.** Los agentes que mantienen un chat con un resultado específico - tales como una acción de compra o una resolución exitosa ante un problema de consumidor - usualmente son representados por un estado deseado de máquina. Al cargar todas los diálogos en una gráfica, se puede analizar patrones actuales y observar discrepancias para el estado esperado de la máquina. Esto ayuda a encontrar fallas y oportunidades para mejoras de producto.

11. ChatOps impulsada por LLMs

Evaluar

ChatOps impulsada por LLMs (Modelos de Lenguaje Grandes, LLM por sus siglas en inglés) es una aplicación emergente de los modelos de lenguaje grandes mediante una plataforma de chat (principalmente Slack) que permite a los ingenieros construir, desplegar y operar software empleando el lenguaje natural. Tiene el potencial de optimizar los flujos de trabajo de ingeniería mejorando las posibilidades de descubrimiento de los servicios de la plataforma y haciendo que sean más amigables para el usuario. En el momento de escribir estas líneas, dos ejemplos tempranos son PromptOps y Kubiya. Sin embargo, considerando la delicadeza necesaria para los entornos de producción, las organizaciones deben evaluar exhaustivamente estas herramientas antes de permitirles acercarse a producción.

12. Agentes autónomos impulsados por LLMs

Evaluar

Los agentes autónomos impulsados por LLMs (Modelos de Lenguaje Grandes, LLM por sus siglas en inglés) están evolucionando más allá de sistemas de un solo agente y sistemas multiagente estáticos con la aparición de frameworks como Autogen y CrewAI. Estos frameworks permiten a los usuarios definir agentes con roles específicos, asignarles tareas y permiten que los agentes colaboren para completar esas tareas mediante la delegación o la conversación. De manera similar a los sistemas de un solo agente que surgieron antes, como AutoGPT, los agentes individuales pueden descomponer tareas, utilizar herramientas preconfiguradas y solicitar opinión humana. Aunque todavía se encuentra en sus primeras fases de desarrollo, este área se está desarrollando rápidamente y tiene un gran potencial para la exploración.

13. Usar GenAI para entender las bases de código heredado

Evaluar

Las IA Generativas (GenAI) y modelos de lenguaje grandes (LLMs) pueden ayudar a los desarrolladores a escribir y entender código. Hasta ahora, en la aplicación práctica, esto ha sido limitado para extractos pequeños de código, pero más desarrollos de productos y tecnologías están emergiendo para **usar GenAI para entender las bases de código heredado.** Esto es particularmente útil en el caso de bases de código heredado que no están bien documentadas o donde la documentación está desactualizada o sea engañosa. Por ejemplo, Driver AI o bloop usan enfoques RAG que combinan inteligencia de lenguaje y búsqueda de código con LLMs para ayudar a los usuarios a encontrar su camino dentro de una base de código. Los modelos emergentes con ventanas de contexto cada vez más grandes también ayudan a hacer estas técnicas más viables para bases de código de tamaño considerable. Otra aplicación prometedora de GenAI para código heredado está en el espacio de la modernización de mainframes, allí los cuellos de botella comúnmente se dan en la ingeniería inversa, en donde está la necesidad de entender la base de código existente y convertir este entendimiento en requisitos para modernizar el proyecto. El uso de GenAI para asistir a los encargados del proceso de ingeniería inversa puede ayudarles a realizar su trabajo más rápidamente.

14. VISS

Evaluar

Recientemente, Zoom liberó como código abierto a su Sistema de Puntuación de Impacto de Vulnerabilidades o VISS. (Vulnerability Impact Scoring System). Éste se enfoca en la puntuación de vulnerabilidades priorizando las medidas de seguridad verdaderamente demostradas. VISS se diferencia del Sistema Común de Puntuación de Vulnerabilidades (CVSS, Common Vulnerability Scoring System) al no enfocarse en los escenarios más pesimistas e intentar medir de forma más objetiva el impacto de las vulnerabilidades desde la perspectiva de la defensa. Para esto, VISS provee una interfaz web donde se puede calcular la puntuación de una vulnerabilidad basada en varios parámetros, categorizados en plataforma, infraestructura y grupos de datos, incluyendo el impacto sobre la plataforma, el número de tenants impactados, el impacto sobre los datos y más. Aunque no tenemos mucha experiencia práctica con esta herramienta en específico, basados en la industria y en el contexto, creemos que vale la pena poner en práctica este método de evaluación adaptado a prioridades.

15. Amplias pruebas de integración

Resistir

Mientras aplaudimos el foco en las pruebas automatizadas, seguimos viendo numerosas organizaciones que invierten excesivamente en lo que creemos son ineficaces **Amplias pruebas de integración.** Como el término "prueba de integración" es ambiguo, tomamos la clasificación amplia de la entrada bliki de Martin Fowler sobre este tema, haciendo referencia a una prueba que requiere versiones en produccion de todas las dependencias en tiempo de ejecución. Tales pruebas son obviamente costosas, ya que requieren entornos de prueba con todas las funcionalidades, con la infraestructura necesaria, datos y servicios. Administrar las versiones correctas de dichas dependencias requiere de un trabajo de coordinación importante, que tiende a ralentizar los ciclos de lanzamiento. Para finalizar, las pruebas por sí mismas son comúnmente frágiles y de poca ayuda. Por ejemplo, necesitamos mucho esfuerzo para determinar si una prueba falló por el código nuevo, la versión incorrecta de dependencias o por el entorno, además, el mensaje de error rara vez ayuda a identificar la fuente del error. Estas críticas no significan que nos opongamos a las pruebas automatizadas de integración "caja negra" en general, más bien, encontramos que un enfoque más útil es aquel que balancea las necesidades de confianza con la frecuencia de lanzamiento. Esto se puede hacer en dos etapas: primero validando el comportamiento del sistema bajo prueba asumiendo cierto conjunto de respuestas de las dependencias en tiempo de ejecución y luego, validando dichas suposiciones. La primera etapa usa la virtualización de servicios para crear pruebas dobles de dependencias en tiempo de ejecución y validar el comportamiento del sistema bajo la prueba. Eso simplifica lo que afecta a la gestión de los datos de prueba y permite tests deterministas. La segunda etapa utiliza pruebas de contrato para validar las suposiciones del entorno con dependencias reales.

16. Uso excesivamente entusiasta de LLM

Resistir

En la carrera por aprovechar lo último en IA, muchas organizaciones están adoptando rápidamente modelos de lenguaje de gran tamaño (LLMs) para una variedad de aplicaciones, desde la generación de contenido hasta procesos complejos de toma de decisiones. El atractivo de los LLMs es innegable; ofrecen una solución aparentemente sin esfuerzo a problemas complejos, y los desarrolladores a menudo pueden crear dicha solución rápidamente y sin necesidad de años de experiencia en aprendizaje automático profundo. Puede ser tentador implementar una solución basada en LLM tan pronto como esté más o menos funcionando y luego continuar. Aunque estas pruebas de valor basadas en LLM son útiles, recomendamos a los equipos que miren cuidadosamente para qué se está utilizando la tecnología y consideren si un LLM es realmente la solución final adecuada. Muchos problemas que un LLM puede resolver — como el análisis de sentimientos o la clasificación de contenido — se pueden resolver de manera más barata y sencilla usando procesamiento de lenguaje natural (NLP) tradicional. Analizar lo que el LLM está haciendo y luego analizar otras soluciones potenciales no sólo mitiga los riesgos asociados con el **uso excesivamente entusiasta de LLM,** sino que también promueve una comprensión y aplicación más matizada de las tecnologías de IA.

17. Prisa por afinar LLMs

Resistir

A medida que las organizaciones buscan formas de hacer que los modelos de lenguaje grandes (LLM por sus siglas en inglés) funcionen en el contexto de su producto, dominio o conocimiento organizativo, estamos viendo **prisa por afinar LLMs.** Aunque afinar un LLM puede ser una herramienta potente para ganar más especialización para un caso de uso, en muchos casos no es necesario. Uno de los casos más comunes de prisa por afinar mal encaminada es hacer a una aplicación respaldada por LLM consciente de conocimiento específico y hechos o de los códigos fuente de una organización. En la gran mayoría de los casos, usar una forma de retrieval-augmented generation (RAG) ofrece una mejor solución y mejor relación coste-beneficio. Afinar requiere de considerables recursos de cómputo y experiencia, e introduce incluso más desafíos en relación con datos sensibles y propietarios que RAG. También existe un riesgo de subajuste cuando no se dispone de suficientes datos para afinar, o, con menos frecuencia, de sobreajuste cuando se tienen demasiados datos y por lo tanto no se encuentra el balance correcto de especificidad de la tarea que se necesita. Esté atento a estos equilibrios y considere las alternativas antes de apresurarse a afinar un LLM para su caso de uso.

18. Componentes Web para aplicaciones renderizadas en el servidor (SSR)

Resistir

Con la adopción de marcos de trabajo como Next.js y htmx, vemos un mayor uso de la renderización en el servidor (SSR, server-side rendering). Como una tecnología natural para el navegador, no es trivial utilizar Componentes Web en el servidor. Han surgido marcos de trabajo para facilitar esta tarea, a veces incluso utilizando un motor de navegador, pero la complejidad sigue ahí. Nuestros equipos se encuentran con que necesitan de soluciones alternativas y de esfuerzos extra para ordenar los componentes para el front-end y para el servidor. Peor que la experiencia de desarrollo es la experiencia de usuario: el rendimiento de carga de la página se ve afectado cuando componentes web personalizados tienen que cargarse e hidratarse en el navegador, e incluso, inevitablemente, contenido sin estilo destella por unos momentos o se producen desplazamientos en el diseño, a pesar de que los componentes ya se han renderizado previamente y se les han realizado minuciosos ajustes. Como ya se mencionó en el Radar anterior, uno de nuestros equipos tuvo que cambiar su sistema de diseño basado en Componentes Web y dejar de usar Stencil debido a estos problemas. Recientemente recibimos informes de otro equipo que acabó sustituyendo los componentes renderizados en el servidor por componentes para el navegador debido a la complejidad de desarrollo. Sugerimos tener cuidado con el uso de **componentes web para aplicaciones renderizadas en el servidor**, incluso si los marcos de trabajo los soportan.

Plataformas

Adoptar

19. CloudEvents

Probar

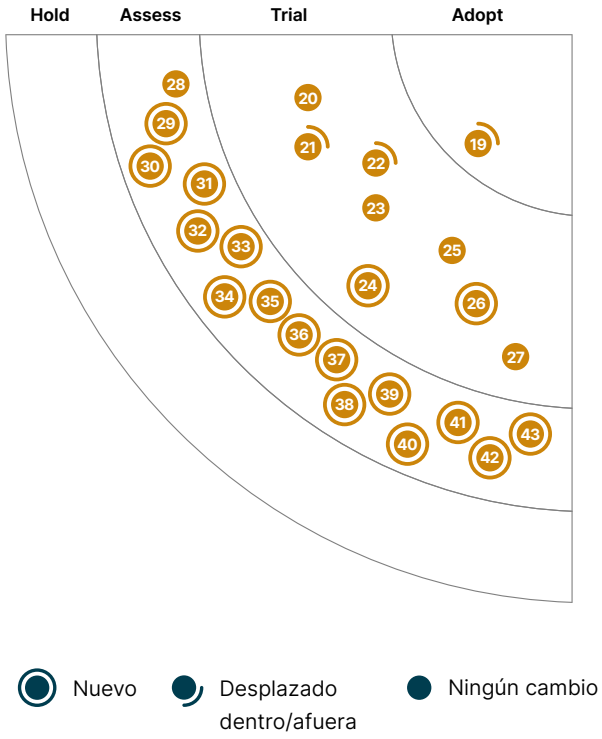
20. Arm en la nube
21. Azure Container Apps
22. Azure OpenAI Service
23. DataHub
24. Plataformas de orquestación de infraestructura
25. Pulumi
26. Rancher Desktop
27. Weights & Biases

Evaluar

28. Bun
29. Chronosphere
30. DataOS
31. Dify
32. Elasticsearch Relevance Engine
33. FOCUS
34. Gemini Nano
35. HyperDX
36. IcePanel
37. Langfuse
38. Qdrant
39. RISC-V para sistemas embebidos
40. Tigerbeetle
41. WebTransport
42. Zarf
43. ZITADEL

Resistir

—



19. CloudEvents

Adoptar

Los eventos son mecanismos comunes de la arquitectura basada en eventos o aplicaciones sin servidor Sin embargo, los productores o proveedores de la nube tienden a soportarlos de diferentes maneras, lo que impide la interoperabilidad entre plataformas e infraestructuras. CloudEvents es una especificación para describir datos de eventos en formatos comunes para proporcionar interoperabilidad entre servicios, plataformas y sistemas. Proporciona SDKs para varios lenguajes de programación para integrar la especificación dentro de la aplicación o cadena de herramientas. Nuestros equipos lo usan no sólo para fines alrededor de las plataformas en la nube, sino también para la especificación de eventos de dominio, entre otros escenarios. CloudEvents está alojado por la Cloud Native Computing Foundation CNCF) y es un proyecto ya graduado. Nuestros equipos utilizan CloudEvents de forma predeterminada para crear arquitecturas basadas en eventos y, por ese motivo, lo trasladamos a Adoptar.

20. Arm en la nube

Probar

Las instancias de cómputo Arm en la nube se han vuelto cada vez más populares en los últimos años debido a su costo y eficiencia enérgetica comparado con las instancias tradicionales basadas en x86. Muchos proveedores de la nube ahora ofrecen instancias basadas en Arm, incluidos AWS, Azure y GCP. El costo beneficio de ejecutar Arm en la nube puede ser particularmente beneficioso para negocios que efectúan grandes cargas de trabajo o tienen necesidad de escalar. Estamos observando a muchos de nuestros equipos migrar a instancias Arm para cargas de trabajo como servicios JVM e incluso bases de datos (incluyendo RDS) sin ningún cambio en el código y cambios mínimos en los scripts de construcción. Las nuevas aplicaciones y sistemas basados en la nube utilizan cada vez más Arm de forma predeterminada. Basados en nuestras experiencias, recomendamos las instancias de cómputo Arm para todas las cargas de trabajo a menos que existan dependencias de arquitectura específicas. Las herramientas que brindan soporte de múltiples arquitecturas, como multi-arch Docker images, también simplifican la construcción y ejecución de los flujos de trabajo.

21. Azure Container Apps

Probar

Azure Container Apps es una plataforma de aplicaciones Kubernetes gestionada que agiliza el despliegue de cargas de trabajo en contenedores. En comparación con Azure Kubernetes Service (AKS), la carga operativa y administrativa de la ejecución de aplicaciones en contenedores se reduce, a expensas de cierta flexibilidad y control, hecho que los equipos deben tener en cuenta. Otro producto en esta área, Azure Container Instances, suele ser demasiado limitado para su uso en producción. Nuestros equipos empezaron a utilizar Azure Container Apps el año pasado, cuando aún estaba en fase de vista previa pública, con buenos resultados incluso al ejecutar contenedores grandes. Ahora que está disponible de forma general, lo estamos considerando para más casos de uso. Tanto Dapr como KEDA Autoscaler son compatibles.

22. Azure OpenAI Service

Probar

Azure OpenAI Service proporciona acceso a GPT-4, GPT-35-Turbo, Embeddings, modelo DALL-E y más de OpenAI a través de una API REST, un SDK de Python y una interfaz web. Los modelos se pueden adaptar a tareas como la generación de contenido, resúmenes, búsqueda semántica y traducción de lenguaje natural a código. Fine-tuning también está disponible mediante el aprendizaje con pocos datos o ejemplos (few-shot learning) y la personalización de hiper parámetros. En comparación con la propia API de OpenAI, Azure OpenAI Service se beneficia de las características de seguridad y cumplimiento de nivel empresarial de Azure, está disponible para más regiones (aunque la disponibilidad es limitada fpara cada una de las regiones geográficas más grandes) y admite redes privadas, filtrado de contenido y control manual de versiones del modelo. Por estos motivos y nuestra experiencia positiva con él, recomendamos que las empresas que ya utilizan Azure consideren utilizar Azure OpenAI Service en lugar de la API de OpenAI.

23. DataHub

Probar

Al construir productos de datos usando un enfoque en el producto, s necesario considerar el linaje, la visibilidad y la gobernanza de los datos. Nuestros equipos han encontrado que **DataHub** puede ser de gran ayuda en estos casos. A pesar de que las primeras versiones de DataHub necesitaban que se genere una rama y se gestione la sincronización desde el producto principal a mano (si existiese la necesidad de actualizar el modelo de metadatos), las mejoras recientes han introducido características que permiten a nuestros equipos implementar modelos personalizados de metadatos con una arquitectura basada en complementos. Otra característica útil de DataHub es el robusto linaje de datos de extremo a extremo, desde el origen, pasando por el procesamiento, hasta el consumo. DataHub soporta integraciones de tipo push y pull para la extracción de linajes que automáticamente examinan los metadatos de orígenes, schedulers, orquestadores (scanning the Airflow DAG), tareas de procesamiento de pipelines y tableros (dashboards), por nombrar algunos. Como opción de código abierto para un catálogo de datos holístico, DataHub se está convirtiendo en la opción por defecto de nuestros equipos.

24. Plataformas de orquestación de infraestructura

Probar

La base de código para la orquestación de infraestructura interna con frecuencia se convierte en una pérdida de tiempo para mantener y solucionar problemas. La aparición de plataformas de orquestación de infraestructuras promete estandarizar y producir varios aspectos de los flujos de trabajo de despliegue y entrega de código de infraestructura. Estos incluyen herramientas de compilación como Terragrunt y Terraspace, servicios de los proveedores de herramientas IaC como Terraform Cloud y Pulumi Cloud así como herramientas y servicios de plataformas independientes como env0 y Spacelift. Existe un rico ecosistema de herramientas y servicios de orquestación de Terraform, frecuentemente llamado TACOS (Terraform Automation and Collaboration Software), incluyendo Atlantis, Digger, Scalr, Terramate y Terrateam. Cada una de estas herramientas permite diferentes flujos de trabajo, incluyendo GitOps, Entrega Continua y cumplimiento como código. Damos la bienvenida al crecimiento de soluciones en este espacio. Recomendamos que los equipos de

plataforma e ingeniería exploren cómo usarlos para reducir la cantidad de código diferenciador que necesitan para desarrollar y mantener sus infraestructuras. La estandarización de cómo se estructura, comparte, entrega y despliega el código de infraestructura, también debería crear oportunidades para el surgimiento de un ecosistema de herramientas compatibles para probar, medir y monitorear la infraestructura.

25. Pulumi

Probar

Las herramientas en el espacio de la infraestructura como código siguen evolucionando, y nos complace ver que **Pulumi** no es la excepción a esta tendencia. La plataforma recientemente añadio soporte para Java y YAML, para gestionar la infraestructura a escala así como para multitud de configuraciones e integraciones en la nube, haciendo a la plataforma aún más convincente. Para nuestros equipos, sigue siendo la principal alternativa a Terraform para desarrollar código para múltiples plataformas en la nube.

26. Rancher Desktop

Probar

Los cambios en las licencias de Docker Desktop nos han dejado buscando desesperadamente alternativas para ejecutar una flota de contenedores en el entorno local del portátil de las personas desarrolladoras. Hemos tenido éxito recientemente con **Rancher Desktop**. Esta aplicación gratuita y de código abierto es relativamente fácil de descargar e instalar en máquinas Apple, Windows o Linux y proporciona un práctico clúster local de Kubernetes con una interfaz gráfica de usuario para la configuración y monitorización. Aunque Colima se ha convertido en nuestra alternativa predilecta a Docker Desktop, es fundamentalmente una herramienta de línea de comandos. En contraposición, Rancher Desktop atraerá a aquellas personas que no quieran renunciar a la interfaz gráfica que Docker Desktop proporciona. Al igual que Colima, Rancher Desktop te permite elegir entre dockerd o containerd como el entorno de ejecución de contenedores subyacente. La elección de containerd directo te libera de DockerCLI, pero la opción de dockerd proporciona compatibilidad con otras herramientas que dependen de él para comunicarse con el demonio de ejecución.

27. Weights & Biases

Probar

Weights & Biases es una plataforma de aprendizaje automático (ML por sus siglas en inglés) para construir modelos rápidamente mediante el seguimiento de experimentos, versionado de conjuntos de datos, visualización del rendimiento del modelo y gestión de modelos. Puede integrarse con el código de ML existente para obtener métricas en vivo, registros de terminal y estadísticas del sistema transmitidas al tablero para un análisis más profundo. Recientemente, Weights & Biases ha expandido su alcance a la observabilidad de modelos de lenguaje de gran escala (LLM) con Traces. Traces visualiza el flujo de ejecución de cadenas de prompts, así como las entradas/salidas intermedias, y proporciona metadatos sobre la ejecución de la cadena (como los tokens utilizados y el tiempo de inicio y finalización). Nuestros equipos lo encuentran útil para depurar y obtener un mayor entendimiento de la arquitectura de secuencias de prompts.

28. Bun

Evaluar

Bun es un nuevo motor de tiempo de ejecución (run time) para JavaScript, similar a Node.js o Deno. A diferencia de éstos, Bun está creado a partir del componente JavaScriptCore de WebKit en vez del motor V8 de Chrome. Diseñado como un sustituto directo para Node.js, Bun es un archivo ejecutable (escrito en Zig) que actúa como un empaquetador (bundler), transpilador (transpiler), y administrador de paquetes para aplicaciones JavaScript y TypeScript Desde nuestra última edición, Bun ha dejado la fase beta con la versión estable 1.0. Su implementación se ha hecho contemplando varias optimizaciones, como un arranque rápido, renderización mejorada en el servidor y un gestor de paquetes alternativo mucho más veloz, por lo que te animamos a probarlo.

29. Chronosphere

Evaluar

Cuando se gestionan arquitecturas distribuidas, tener en cuenta el coste de ordenar, indexar y acceder a los datos es tan importante como la observabilidad. **Chronosphere** Adopta un enfoque único a la gestión de costes, haciendo un seguimiento del uso de los datos de observabilidad para que las organizaciones puedan considerar las compensaciones coste-valor de distintas métricas. Con la ayuda del Analizador de uso de métricas, que forma parte del Panel de control de Chronosphere, los equipos pueden identificar y excluir las métricas que rara vez (o nunca) utilizan, lo que significa un importante ahorro de costes al reducir la cantidad de datos que las organizaciones tienen que revisar. Dadas estas ventajas, así como la capacidad de Chronosphere para igualar la funcionalidad de otras herramientas de observabilidad para soluciones alojadas en la nube, creemos que es una opción convincente que las organizaciones deben considerar.

30. DataOS

Evaluar

Con el incremento de la adopción de data mesh nuestros equipos han estado en la búsqueda de plataformas de datos que traten a los productos de datos como entidades de primera clase. **DataOS** es uno de esos productos. Proporciona una gestión del ciclo de vida de extremo a extremo para diseñar, construir, desplegar y evolucionar productos de datos. Ofrece especificaciones declarativas estandarizadas escritas en YAML que abstraen la complejidad de bajo nivel de la instanciación de la infraestructura y permiten a las personas desarrolladoras definir fácilmente los productos de datos a través de un CLI/API. Este soporta políticas de control de acceso con ABAC y políticas de datos para filtrar y enmascarar los datos. Otra funcionalidad a resaltar es la habilidad de federar datos por toda una gran variedad de fuentes de datos, lo que reduce la duplicación y el movimiento de datos a un lugar centralizado. DataOS funciona mejor para escenarios nuevos donde hace el trabajo pesado ya que proporciona una solución lista para usar para la gobernanza y descubrimiento de datos, para el manejo de los recursos de infraestructura y para la observabilidad. En proyectos con antigüedad, la habilidad de orquestar recursos fuera de DataOS (por ejemplo, pilas de datos como Databricks) está en estado emergente y aún en evolución. Si tu ecosistema no tiene mucha opinión sobre las herramientas de datos, DataOS es una buena manera de acelerar tu camino de construir, desplegar y consumir productos de datos de extremo a extremo.

31. Dify

Evaluar

Dify es una plataforma visual para el desarrollo de aplicaciones basadas en grandes modelos de lenguaje (LLM, large language model) que hace que la creación de prototipos sea aún más accesible. Soporta el desarrollo de aplicaciones de chat y de generación de texto con plantillas de prompts. Además, Dify soporta generación aumentada por recuperación (RAG, retrieval-augmented generation) con conjuntos de datos importados y puede trabajar con múltiples modelos. Estamos entusiasmados con esta categoría de aplicaciones. Sin embargo, basados en nuestra experiencia, Dify aún no está totalmente lista, ya que algunas de sus características presentan errores o no parecen estar completamente desarrolladas. No obstante, por el momento, no conocemos de ningún competidor que sea mejor.

32. Elasticsearch Relevance Engine

Evaluar

A pesar de que las bases de datos vectoriales han ganado popularidad para los casos de uso de generación mejorada por recuperación (RAG), Investigaciones y reportes de experiencia sugieren que combinar la busqueda de textos completos tradicional con la busqueda por vectores (a una busqueda hibrida) genera mejores resultados. A través de **Elasticsearch Relevance Engine (ESRE)**, la bien establecida plataforma de búsqueda de texto completo Elasticsearch soporta modelos embebidos tanto incorporados como adaptados, búsqueda vectorial y busqueda hibrida con mecanismos de clasificación como la Fusion Reciproca Categorica. A pesar de que este espacio está aún madurando, en nuestra experiencia, utilizar estas funcionalidades ESRE en conjunto con las capacidades de filtrado, clasificación y ordenamiento tradicionales que vienen con Elasticsearch han otorgado resultados prometedores, sugiriendo que las plataformas de búsqueda establecidas que soportan búsqueda semántica no deben ser pasadas por alto.

33. FOCUS

Evaluar

Los datos sobre facturación en la nube y SaaS pueden ser complejos, inconsistentes entre proveedores y difíciles de entender. **FOCUS** (FinOps Open Cost & Usage Specification) busca reducir la fricción con una especificación que contiene un conjunto de terminologías (alineadas con el framework de FinOps), un esquema y un conjunto mínimo de requerimientos para los datos de facturación. La especificación está destinada a soportar casos de uso común a una variedad de profesionales de FinOps. A pesar de que sigue en una fase temprana de desarrollo y adopción, vale la pena echarle un vistazo porque, con la creciente adopción de la industria, FOCUS hará más fácil para las plataformas y los usuarios finales obtener una visión holística de sus gastos en la nube a través de una larga lista de proveedores de la nube y SaaS.

34. Gemini Nano

Evaluar

Gemini es una familia de LLMs fundamentales diseñados para ser ejecutados en una gran variedad de hardware, desde data centers hasta teléfonos celulares. **Gemini Nano** ha sido específicamente optimizado y reducido para ejecutarse en aceleradores de silicio móviles. Esto habilita capacidades como resúmen de texto de alta calidad, respuestas de contexto inteligentes y correcciones

gramaticales avanzadas. Por ejemplo, el entendimiento de lenguaje de Gemini Nano le permite al Pixel 8 Pro resumir contenido en la app de Grabación. La ejecución en el dispositivo elimina muchas de las preocupaciones de latencia y privacidad relacionadas con sistemas basados en la nube y además permite que algunas características sigan funcionando sin conexión de red. Android AICore simplifica la integración del modelo en aplicaciones Android, pero solo unos pocos dispositivos son soportados al momento de escribir este volumen del radar.

35. HyperDX

Evaluar

HyperDX es una plataforma de observabilidad de código abierto, que unifica los tres pilares de la observabilidad: registros, métricas y seguimiento. Con esto, puedes correlacionar un extremo a otro y pasar desde la reproducción de sesiones hasta los registros y seguimiento en solo unos pocos clics. La plataforma aprovecha a ClickHouse como un almacén central de datos para toda la información de telemetría, y escala para agregar patrones de registro y condensar miles de millones de eventos en distintos clusters. Aunque puedes elegir entre varias plataformas de observabilidad, queremos destacar a HyperDX por su experiencia de desarrollo unificada.

36. IcePanel

Evaluar

IcePanel facilita el modelado de arquitectura y la creación de diagramas utilizando el modelo C4, de forma colaborativa, lo cual permite a los stakeholders técnicos y comerciales acercarse al nivel de detalle técnico que necesitan. Admite el modelado de componentes de arquitectura cuyos metadatos y conexiones se pueden reutilizar entre diagramas, junto con la visualización de flujos entre esos componentes. El control de versiones y el etiquetado permiten a los colaboradores modelar diferentes estados de la arquitectura (por ejemplo, tal como está versus lo que será) y realizar un seguimiento de las clasificaciones definidas por el usuario de varias partes de la misma. Estamos atentos a IcePanel por su potencial para mejorar la colaboración en arquitectura, particularmente para organizaciones con arquitecturas complejas. Para encontrar una alternativa con mejor soporte para diagramas como código, consulta Structurizr.

37. Langfuse

Evaluar

Langfuse es una plataforma de ingeniería para observabilidad, testabilidad y monitorización de aplicaciones que utilizan Modelos de Lenguaje de Gran Tamaño (LLM). Su SDK permite utilizar Python, JavaScript y TypeScript, OpenAI, LangChain y LiteLLM entre otros lenguajes y frameworks. Puedes auto-hospedar la versión abierta o utilizarla como un servicio de pago en la nube. Nuestros equipos han tenido una experiencia positiva, particularmente depurando cadenas LLM complejas, analizando la completitud y monitorizando métricas clave como coste y latencia entre usuarios, sesiones, geografías, prestaciones y versiones del modelo. Si estás considerando construir aplicaciones LLM dirigidas por datos, Langfuse es una buena opción a tener en cuenta.

38. Qdrant

Evaluar

Qdrant es una base de datos de vectores escrita en Rust de código abierto. En la edición del Radar de septiembre 2023, hablamos de pgvector, una extensión de PostgreSQL para buscar vectores. Sin embargo, si tienes que escalar la base de datos de vectores horizontalmente entre nodos, te recomendamos evaluar Qdrant. Tiene integrado soporte de aceleración single instruction, multiple data (SIMD) para mejorar el rendimiento de la búsqueda y ayudarte a asociar los payloads de JSON con vectores.

39. RISC-V para sistemas embebidos

Evaluar

Mientras la arquitectura Arm continúa expandiendo su impacto — hemos actualizado nuestra evaluación de Arm en la nube en abril de 2024 — el interés en la nueva y menos establecida arquitectura RISC-V ambién crece. RISC-V no trae avances en rendimiento o eficiencia — de hecho, su rendimiento por vatio es similar al de Arm y no puede competir en rendimiento absoluto — pero es de código abierto, modular y no está atado a una única compañía. Esto lo convierte en una propuesta atractiva para sistemas embebidos, donde el costo de las licencias de arquitecturas propietarias es una preocupación significativa. Esta también es la razón por la cual el campo de **RISC-V para sistemas embebidos** está madurando y algunas compañías, incluyendo SiFive y espressif, están ofreciendo tableros de desarrollo y SoCs para una amplia gama de aplicaciones. Microcontroladores y microprocesadores capaces de ejecutar el kernel de Linux están disponibles actualmente, junto con la pila de software y el conjunto de herramientas correspondientes. Seguimos atentos a este espacio y esperamos ver una mayor adopción en los próximos años.

40. Tigerbeetle

Evaluar

Tigerbeetle es una base de datos distribuida de código abierto para contabilidad financiera. A diferencia de otras bases de datos, está diseñada para ser una máquina de estado de dominio específico para mayor seguridad y rendimiento. El estado de un nodo en el clúster se replica de forma determinista en otros nodos mediante el protocolo de consenso Viewstamped Replication Nos gustan bastante las decisiones de diseño detrás de Tigerbeetle para implementar la contabilidad de doble entrada con estrictas garantías de serialización. Es una base de datos relativamente nueva y en evolución activa, aunque aún no está lista para producción.

41. WebTransport

Evaluar

WebTransport es un protocolo basado en HTTP/3 que ofrece comunicación bidireccional entre servidores y aplicaciones. WebTransport ofrece varios beneficios sobre su predecesor, WebSockets, incluyendo conexiones más rápidas, menor latencia y la capacidad para manejar flujos de datos, tanto ordenados y confiables, como no ordenados (como UDP). Puede manejar múltiples flujos en la misma conexión sin bloqueos de línea de cabecera, permitiendo una comunicación más eficiente en aplicaciones complejas. En general, WebTransport es adecuado para un amplio rango de casos de uso, entre ellos aplicaciones web en tiempo real, streaming de medios y comunicaciones de datos para Internet de las Cosas (IoT). Aunque WebTransport aún está en una etapa inicial de su desarrollo (el soporte entre los navegadores va madurando gradualmente, con librerías populares como socket. io que ya lo admite) nuestros equipos están actualmente evaluando su potencial para aplicaciones de IoT en tiempo real.

42. Zarf

Evaluar

Zarf es un gestor de paquetes declarativo para ambientes offline y semiconectados de Kubernetes Con Zarf, puedes crear y configurar aplicaciones mientras está conectado a Internet; una vez creados, puedes empacar y enviar a un ambiente desconectado para ser desplegado. Como herramienta independiente, Zarf trae consigo varias características útiles, incluyendo generación de Lista de Materiales de Software (SBOM en inglés) registro de Docker incorporado, Gitea y tableros de K9s para administrar clústers desde la terminal. Entrega de software Air-gap para aplicaciones nativas en la nube tiene algunos desafíos; Zarf aborda la mayoría de ellos.

43. ZITADEL

Evaluar

ZITADEL es una herramienta de gestión de identidades y usuarios de código abierto, y una alternativa a Keycloak. Es ligera (escrita en Golang), tiene opciones de despliegue flexibles y es fácil de configurar y gestionar. También es multi-tenant, ofrece características completas para la construcción de sistemas de autenticación seguros y escalables, en particular para aplicaciones B2B, tiene características de seguridad integradas como la autenticación multifactor y pistas de auditoría. Con ZITADEL, los desarrolladores pueden reducir el tiempo de desarrollo, mejorar la seguridad de las aplicaciones y conseguir escalabilidad para bases de usuarios en crecimiento. Si buscas una herramienta de código abierto, segura y fácil de usar para la gestión de usuarios, ZITADEL es un firme candidato.

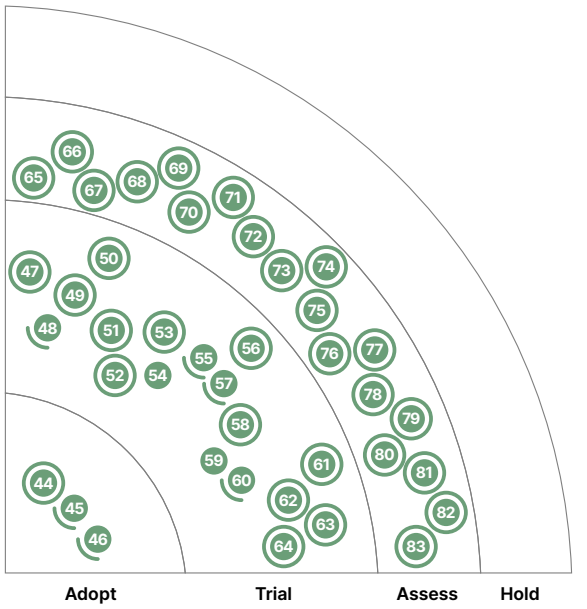
Herramientas

Adoptar
44. Conan
45. Kaniko
46. Karpenter

Probar
47. API Conformance Scan de 42Crunch
48. actions-runner-controller
49. Android Emulator Container
50. AWS CUDOS
51. aws-nuke
52. Bruno
53. Develocity
54. GitHub Copilot
55. Gradio
56. Catálogos de versiones de Gradle
57. Maestro
58. Herramienta SBOM de Microsoft
59. Open Policy Agent (OPA)
60. Ejecutores de GitHub auto-alojados de Philips
61. Pop
62. Renovate
63. Terrascan
64. Velero

Evaluar
65. aider
66. Akvorado
67. Baichuan 2
68. Cargo Lambda
69. Codium AI
70. Continue
71. Fern Docs
72. Granted
73. LinearB
74. LLaVA
75. Marimo
76. Mixtral
77. NeMo Guardrails
78. Ollana
79. OpenTofu
80. QAnything
81. System Initiative
82. Tetragon
83. Winglang

Resistir
—



Nuevo
Desplazado dentro/afuera
Ningún cambio

44. Conan

Adoptar

Conan es un gestor de dependencias de código abierto para aplicaciones en C/C++. Provee una interfaz intuitiva para definición, adquisición y administración de dependencias, lo cual simplifica el trabajo de los desarrolladores al integrar librerías provistas por terceros dentro de sus proyectos. Conan trabaja en la mayoría de sistemas operativos y una gran variedad de plataformas, incluyendo equipos de escritorio, servidores, dispositivos móviles y embebidos. Puede también ser utilizado para compilar y publicar librerías y paquetes de C/C++. Los paquetes pueden ser compartidos con equipos de trabajo vía servidores JFrog Artifactory. Al tomar ventaja sobre binarios pre-compilados, reduce significativamente tiempos de compilación, especialmente para casos de dependencias complejas y de gran tamaño. Conan se integra con sistemas de compilación populares como CMake y posee además un SDK para Python, extendiendo los sistemas de compilación para tareas como firmado. En nuestra experiencia, Conan se traduce en mejoras de replicación de compilación a lo largo de diferentes entornos y una mayor velocidad en ciclos de desarrollo. Las bases de código resultantes son más limpias y fáciles de mantener, un gran valor agregado para proyectos de gran tamaño en C y C++. Si te encuentras luchando con la administración de dependencias en tus proyectos, Conan es una herramienta que debes considerar para mejorar tu eficiencia de desarrollo.

45. Kaniko

Adoptar

Añadimos Kaniko al Radar en octubre de 2022, poco después de que Kubernetes dejara de dar soporte a Docker, exponiendo en ese momento la tendencia a apartarse de Docker como el constructor de contenedores de imagen por defecto en pipelines basadas en contenedores. Desde entonces hemos experimentado exhaustivamente con Kaniko utilizando varias herramientas y configuraciones de pipelines. Nuestros equipos aprecian la flexibilidad y el rendimiento de Kaniko y es por esto que lo hemos movido al área de Adopción, convirtiéndola en la herramienta por defecto en este espacio.

46. Karpenter

Adoptar

Una de las capacidades fundamentales de Kubernetes es el escalamiento automático horizontal: su habilidad para iniciar nuevos pods cuando se necesita capacidad adicional y apagarlos cuando la carga disminuye. Sin embargo, esto solo funciona si los nodos necesarios para alojar los pods ya existen. Cluster Autoscaler puede expandir de forma rudimentaria el clúster cuando suceden fallos de pods, pero no es muy flexible; En cambio, **Karpenter**, es un escalador automático de nodos, de tipo Operador Kubernetes más inteligente y de código abierto: analiza las cargas de trabajo actuales y las limitaciones de planificación de los pods, selecciona un tipo apropiado de instancia y la inicia o detiene según sea necesario. Karpenter es un operador al estilo de herramientas como Crossplane que puede provisionar recursos de la nube por fuera del clúster. A pesar de que Karpenter fue originalmente desarrollado por AWS para EKS, se está convirtiendo en la herramienta de aprovisionamiento automático de nodos predeterminada para todos los proveedores de servicios de Kubernetes en la nube; y Azure recientemente empezó a soportar a Karpenter mediante el AKS Karpenter Provider.

47. API Conformance Scan de 42Crunch

Probar

El API Conformance Scan _** de 42Crunch** es una herramienta de pruebas dinámicas diseñada para identificar discrepancias entre el comportamiento documentado de una API y su implementación real. Esta herramienta toma la definición de especificaciones de la API en formato OpenAPI, que describe las funcionalidades y respuestas esperadas, y la compara con su comportamiento real. Al generar tráfico real e interactuar con los endpoints vivos, la herramienta puede identificar cualquier discrepancia entre lo que la API promete y lo que entrega. Esto se traduce en varias ventajas para los equipos de desarrollo. Por ejemplo, detecta incoherencias en fases tempranas del desarrollo, lo que ahorra tiempo y previene que los problemas lleguen a producción. La herramienta también ayuda a mejorar la calidad y la seguridad de la API al identificar posibles vulnerabilidades que surgen de desviaciones del comportamiento documentado. En general, API Scan ayuda a evaluar la postura de seguridad de las APIs identificando problemas como protocolos de autenticación débiles, prácticas de manejo de datos inseguras e insuficientes validaciones de datos de entrada. Proporciona informes detallados que resaltan los problemas encontrados y recomendaciones para su solución.

48. actions-runner-controller

Probar

actions-runner-controller es un controlador de Kubernetes que opera con ejecutores auto-hospedados para GitHub Actions. Los ejecutores auto-hospedados pueden ayudar en escenarios donde el trabajo que Github Actions ejecuta necesita acceder a recursos que o no son accesibles para los ejecutores en la nube de GitHub o tienen un sistema operativo específico y requisitos de entorno que son diferentes a los que GitHub provee. En aquellos escenarios donde el equipo usa clusters de Kubernetes, actions-runner-controller orquesta y escala estos ejecutores. A nuestros equipos les gusta la habilidad para escalar ejecutores basados en el número de flujos de trabajo ejecutándose en un repositorio, organización, empresa o cluster de Kubernetes, así como su habilidad para manejar tanto ejecutores de Linux y Windows.

49. Android Emulator Container

Probar

Contenedores de Emuladores de Android agilizan las pruebas de aplicaciones Android al eliminar las complejidades derivadas de los problemas de compatibilidad con el sistema operativo y las dependencias del sistema, así como de la configuración de emuladores para varias versiones de Android. Tradicionalmente, esta complejidad suponía un esfuerzo adicional o que los equipos renunciaran por completo a las pruebas automatizadas, lo que, a su vez, ralentizaba los ciclos de desarrollo y pruebas. Los contenedores de emuladores de Android simplifican este proceso, permitiendo una integración perfecta en las pipelines CI para pruebas automatizadas. Nuestros equipos utilizan estos contenedores principalmente para pruebas instrumentadas, que se ejecutan automáticamente con cada commit para proporcionar retroalimentación instantánea a los desarrolladores. Además, aprovechamos los contenedores de emuladores de Android para ejecutar pruebas nocturnas de extremo a extremo.

50. AWS CUDOS

Probar

Nuestra recomendación ha sido siempre monitorizar costes como una función de aptitud. Los proveedores de la nube ofertan una variedad de servicios para monitorizar costes como por ejemplo AWS Cost Explorer o Google Cloud FinOps Hub. En el ecosistema AWS, nuestros equipos utilizan los paneles de control CUDOS (Cost and Usage Dashboards Operations Solution) para monitorizar el gasto en AWS Marketplace desglosado por departamentos de negocio o entidades legales en organizaciones matriciales grandes. Este panel de control provee de detalles integrales de costes y uso, con granularidad a nivel de recurso que ayuda a optimizar costes, hacer seguimiento de los objetivos de uso y alcanzar la excelencia operacional.

51. aws-nuke

Probar

aws-nuke es una herramienta de código abierto que aborda el habitual desafío de acumulación de recursos no utilizados en cuentas AWS de desarrollo y pruebas que pueden llevar a ineficiencias de costes. La herramienta identifica y borra todos los recursos eliminables en una cuenta AWS o región a excepción de los recursos por defecto o gestionados por AWS, esencialmente restableciendo el entorno a estado Día Uno. También ofrece políticas de exclusión personalizables para asegurar que recursos críticos permanecen protegidos. Hemos utilizado esta herramienta para el caso de uso por defecto de optimización de costes así como en contextos de recuperación ante desastres (DR por sus siglas en inglés) con buenos resultados. Automatizando la limpieza en entornos de desarrollo y pruebas, aws-nuke ayuda a minimizar el gasto innecesario de recursos. También facilita el desmontaje de infraestructura temporal para DR después de simulacros o ejercicios. Aunque estable, aws-nuke es una herramienta muy destructiva y no está orientada a ser utilizada en entornos de producción. Realice siempre una simulación para confirmar que los recursos esenciales no serán borrados.

52. Bruno

Probar

Bruno es una alternativa de escritorio a Postman e Insomnia de código abierto para pruebas, desarrollo y depuración de APIs. Almacena tus colecciones localmente en el sistema de archivos para que puedas usar Git o un sistema de control de versiones de tu elección para colaborar. Varios equipos de Thoughtworks están utilizando Bruno y les gusta su diseño simple y solo offline.

53. Develocity

Probar

Develocity (anteriormente Gradle Enterprise) aborda el problema de los largos ciclos de compilación y prueba en proyectos de software a gran escala. Emplea mejoras de rendimiento, como el almacenamiento en caché de compilaciones y la selección de pruebas predictivas para acortar los ciclos de feedback de los desarrolladores tanto en entornos locales como de CI/CD. Nuestros equipos de plataforma lo han encontrado útil para acelerar compilaciones y pruebas, analizar comandos para determinar qué parte del flujo de trabajo aún necesita optimizarse, identificar y solucionar problemas de pruebas inestables y realizar análisis en el hardware utilizado para ejecutarlas.

54. GitHub Copilot

Probar

Mientras que el mercado de asistencia de codificación con IA está cada vez más ocupado, GitHub Copilot sigue siendo nuestra opción por defecto y es utilizada por muchos de nuestros equipos. Desde la última vez que escribimos sobre GitHub Copilot, las mejoras más interesantes se han producido en la función de chat. Por ejemplo, ya no es necesario saturar el código con comentarios como indicaciones; en su lugar, un chat en línea te ayuda a proporcionar indicaciones sin tener que escribir un comentario. El chat en línea también puede modificar el código, no sólo escribir nuevas líneas. Ahora también puedes ampliar significativamente el contexto del chat cuando hagas preguntas sobre tu código, utilizando la etiqueta @workspace. Esto le permite hacer preguntas sobre todo el código base, no sólo los archivos abiertos. Puedes ampliar este contexto aún más con la versión Copilot Enterprise que extrae el contexto de todos los repositorios que alojas en GitHub. Por último, GitHub ha empezado a enrutar algunas peticiones de chat a un modelo más potente basado en GPT-4, y la disponibilidad del chat en los populares IDEs de Jetbrains es inminente (aunque todavía en beta privada en el momento de escribir esto). Estos lanzamientos demuestran que el ritmo de las mejoras en este ámbito no se ha ralentizado. Si probaste un asistente de codificación el año pasado y lo descartaste, te recomendamos que sigas atento a las funciones que se van lanzando y le des otra oportunidad.

55. Gradio

Probar

Gradio es una librería Python de código abierto que facilita la creación de interfaces interactivas basadas en la web para modelos de aprendizaje automático (ML). Una interfaz de usuario gráfica sobre los modelos de ML proporciona un mejor entendimiento de las entradas, restricciones y salidas por parte de audiencias no técnicas. Gradio ha ganado mucha acogida en el espacio de la IA generativa, ya que es una de las herramientas que hace que los modelos generativos sean tan accesibles como para experimentar con ellos. Normalmente, ponemos tecnologías en el anillo Probar cuando las hemos visto usadas en producción al menos una vez. El propósito y fortaleza de Gradio es la experimentación y la creación de prototipos, y la hemos utilizado con ese fin muchas veces. Recientemente, uno de nuestros equipos incluso la utilizó para ayudar a un cliente con demostraciones en vivo en grandes eventos. Estamos muy felices con las capacidades de Gradio para esos casos de uso y, por lo tanto, lo pasamos al anillo Probar.

56. Catálogos de versiones de Gradle

Probar

Los Catálogos de versiones de Gradle ison una característica útil de esta herramienta de construcción que permiten gestionar de una manera más centralizada las dependencias en el archivo de construcción (build file). Nuestros equipos han encontrado que es especialmente útil para proyectos con módulos múltiples de Android. En lugar de definir nombres y versiones fijas de las dependencias en archivos de construcción individuales y gestionar las actualizaciones por separado, es mejor crear un catálogo central de versiones de estas dependencias y luego referenciarlo de manera tipada con la asistencia de Android Studio.

57. Maestro

Probar

Maestro es extremadamente útil cuando se prueban flujos complejos en aplicaciones móviles. Nuestros equipos lo han encontrado fácil de aprender, simple de comprender y sencillo de integrar en nuestro flujo de trabajo de desarrollo. Maestro soporta una variedad de plataformas móviles incluyendo iOS, Android, React Native y aplicaciones en Flutter Su sintaxis declarativa YAML simplifica la automatización de interacciones complejas en la IU móvil. Basados en la evolución de la herramienta, marcada por mejoras en su funcionalidad como soporte comprensivo a iOS, y la introducción de herramientas como Maestro Studio y Maestro Cloud, invitamos a aquellos que buscan optimizar sus procesos de pruebas en aplicaciones móviles a probar esta herramienta.

58. Herramienta SBOM de Microsoft

Probar

La Herramienta SBOM de Microsoft de código abierto, permite generar una lista de materiales de software (SBOM, Software Bill of Materials) compatible con SPDX-Habíamos ya hablado sobre la necesidad de la SBOM y este aplicativo hace que sea más fácil comenzar. La Herramienta SBOM tiene soporte para una variedad de administradores de paquetes populares (incluidos npm, pip y Gradle), haciéndola compatible con una amplia gama de proyectos. Es muy fácil de usar y se puede integrar en los flujos de trabajo de desarrollo existentes, incluida la integración con pipelines de CI/CD. Con la SBOM generada con esta herramienta las personas desarrolladoras obtienen múltiples ventajas. La mejora de la seguridad del software es un beneficio clave, ya que una visión clara de los componentes permite una identificación de vulnerabilidades y una gestión de riesgos más sencilla. El cumplimiento de la licencia también mejora, ya que se puede garantizar el cumplimiento de todos los acuerdos relevantes. Además, la SBOM promueve la transparencia dentro de la cadena de suministro de software, ayudando a rastrear dependencias y a mitigar riesgos potenciales. Si se busca optimizar la generación de la SBOM, mejorar la seguridad del software y obtener control sobre la cadena de suministro de software, dale una oportunidad a la Herramienta SBOM de Microsoft.

59. Open Policy Agent (OPA)

Probar

Open Policy Agent (OPA) es un marco de trabajo uniforme y un lenguaje para declarar, aplicar y controlar políticas. Para nuestros equipos, se ha convertido en una de las formas preferidas para definir políticas para sistemas distribuidos, especialmente cuando necesitamos implementar revisiones de cumplimiento en puntos de cambio. OPA permite a los equipos implementar varios patrones de ingeniería de plataformas, como controlar lo que se despliega a los clústeres de Kubernetes, hacer cumplir los controles de acceso entre los servicios de una malla e implementar políticas de seguridad como código precisas para acceder a los recursos de la aplicación. Aunque hay cierta complejidad asociada a las implementaciones de OPA, ha demostrado ser una herramienta muy valiosa para garantizar el cumplimiento en una cultura DevOps. Seguimos con atención el crecimiento y la madurez de OPA en escenarios diferentes a los de sistemas operacionales, como en las (grandes) soluciones centradas en los datos.

60. Ejecutores de GitHub auto-alojados de Philips

Probar

Aunque los ejecutores de GitHub Actions cubren un rango amplio de los entornos de ejecución más comunes y son los más rápidos para empezar, a veces los equipos necesitan administrar ejecutores autoalojados, como cuando la política de la organización solo permite despliegues a infraestructura alojada de forma privada desde dentro del perímetro de seguridad de la propia organización. En tales casos, los equipos pueden utilizar ejecutores de GitHub autoalojados, de Philips, un módulo de Terraform que inicia ejecutores personalizados en instancias de spot de EC2 de AWS. El módulo también crea un conjunto de funciones Lambda que maneja la gestión del ciclo de vida (escalabilidad hacia arriba y hacia abajo) para estos ejecutores. Según nuestra experiencia, esta herramienta simplifica enormemente el aprovisionamiento y gestión de ejecutores de GitHub Actions auto-alojados. Una alternativa para los equipos que usan Kubenentes es actions-runner-controller.

61. Pop

Probar

La programación en pares se mantiene como una técnica esencial para nosotros, debido a que ayuda a mejorar la calidad del código y difunde el conocimiento dentro del equipo. Aunque es mejor hacerlo en persona, nuestros equipos distribuidos han explorado herramientas para hacer que la programación en pares de forma remota sea lo más agradable y eficaz posible, tales como Tuple, Visual Studio Live Share, Code With Me y herramientas de chat y conferencia de propósito general. La última herramienta que ha llamado nuestra atención es **Pop** (anteriormente conocido como Screen). De los fundadores de Screenhero, permite a múltiples personas compartir pantalla, anotaciones y audio/video de alta calidad. Algunos de nuestros equipos lo han usado extensamente para la programación en pares y para sesiones de trabajo remotas y han valorado su experiencia como positiva.

62. Renovate

Probar

Monitorear y actualizar dependencias automáticamente como parte del proceso de creación de software se ha convertido en una práctica estándar en toda la industria. Remueve la incertidumbre de mantenerse al día con las actualizaciones de seguridad de los paquetes de código abierto a medida que son liberados. Por muchos años, Dependabot ha sido la herramienta estándar para esta práctica, pero Renovate se ha convertido en la herramienta predilecta para muchos de nuestros equipos. Ellos encuentran que Renovate es más afín a los ambientes de desarrollo de software modernos donde un sistema desplegable no solo depende del código y librerías, sino que abarcan herramientas de tiempos de ejecución, infraestructura y servicios de terceros. Renovate cubre las dependencias en estos artefactos auxiliares en conjunto con el código. Nuestros equipos también encuentran que Renovate ofrece mayor flexibilidad a través de las opciones de configuración y personalización. A pesar que Dependabot es una respuesta confiable y segura que está convenientemente integrada con GitHub, recomendamos evaluar Renovate en afán de reducir en mayor medida la carga manual sobre los desarrolladores para mantener los ecosistemas de sus aplicaciones seguros y protegidos.

63. Terrascan

Probar

Terrascan es un analizador de código estático para infraestructura como código (IaC) diseñado para detectar vulnerabilidades de seguridad y problemas de cumplimiento antes de aprovisionar la infraestructura nativa de la nube. Admite el escaneo de Terraform, Kubernetes (JSON/YAML), Helm, AWS CloudFormation, Azure Resource Manager, Dockerfiles y GitHub. El paquete con las políticas por defecto soporta a los proveedores más populares de la nube, GitHub, Docker y Kubernetes. Nuestros equipos usan Terrascan localmente como un pre-commit hook y lo integran en el pipeline de CI para detectar vulnerabilidades y violaciones de IaC.

64. Velero

Probar

Velero es una herramienta de código abierto para respaldar y restaurar recursos y volúmenes persistentes de Kubernetes. Simplifica la recuperación de desastres y las migraciones de clústeres al permitir la realización de respaldos programados y bajo demanda. Velero también permite controles más precisos sobre los recursos a respaldar, así como sobre el flujo de trabajo de respaldo y restauración (con hooks). Nuestros equipos aprecian su facilidad de uso y que opera con las API de Kubernetes, en vez de las capas de nivel inferior como etcd.

65. aider

Evaluar

aider es un asistente de codificación de IA de código abierto. Como muchas herramientas de código abierto en este espacio, aider no tiene integración directa con el IDE, pero se inicia como una interfaz de línea de comandos en la terminal. aider es interesante porque proporciona una interfaz de chat con acceso de escritura al código base en múltiples archivos, mientras que muchos de los productos de asistente de codificación actuales solo leen el código o pueden cambiar un solo archivo a la vez. Esto permite que aider te ayude a implementar conceptos que abarcan múltiples archivos (por ejemplo,agregar localizadores a mi HTML y también usarlos en mi prueba funcional) y crear nuevos archivos y estructuras de carpetas en el código base (por ejemplo,crear un nuevo componente similar al de la carpeta X). Como aider es de código abierto y no un producto alojado, debes traer tu propia clave API de OpenAI o Azure OpenAI para usarlo. Por un lado, esto puede ser genial para un uso ocasional porque solo hay que pagar por uso. Por otro lado, aider parece ser bastante hablador en sus interacciones con la API de IA, así que mantente atento a los costos de las request y los límites de tarifas cuando lo uses.

66. Akvorado

Evaluar

Akvorado es una herramienta de código abierto para análisis y monitoreo de redes. Captura los flujos de red, Netflow/IPFIX y sFlow, los enriquece con nombres de interfaz e información geográfica y luego guarda los flujos actualizados en ClickHouse para análisis futuros. Aunque OpenTelemetry está ganando adopción para observar el tráfico a nivel de aplicaciones, a menudo nos encontramos con desafíos en la capa de red que pueden ser difíciles de detectar y solucionar. Herramientas como Akvorado son muy útiles en tales situaciones, ya que le ayudan a analizar los flujos de red en varios dispositivos en la topología de la red.

67. Baichuan 2

Evaluar

Baichuan 2 es parte de una nueva generación de modelos de lenguaje de gran tamaño de código abierto. Fue entrenado en un corpus de alta calidad con 2,6 billones de tokens, logrando un rendimiento bastante bueno para su tamaño tanto en chino como inglés y en comparativas con varios idiomas. Baichuan ha sido entrenado en varios corpus de dominios específicos, incluidos conjuntos de datos legales y de atención médica, por lo que preferimos usarlo en estos campos y sus relacionados.

68. Cargo Lambda

Evaluar

La eficiencia y desempeño de Rust lo convierte en una buena alternativa para la computación serverless. Otra ventaja es que las funciones de Rust no requieren de un tiempo de ejecución, lo que se traduce en tiempos de arranque más rápidos. Sin embargo, la experiencia de desarrollo al escribir las funciones en Rust no ha sido la mejor. Esto cambió con el uso de **Cargo Lambda**. Como un subcomando de Cargo, se integra con el flujo típico de Rust, permitiendo correr y probar funciones AWS Lambda directamente en el ordenador de desarrollo sin la necesidad de Docker, máquinas virtuales (VMs) u otras herramientas. Usando un toolchain de Zig Cargo Lambda puede ejecutar compilación cruzada de las funciones en diversos sistemas operativos para los sandboxes de Linux utilizados por AWS Lambda, siendo Arm e Intel soportadas como arquitecturas objetivo.

69. Codium AI

Evaluar

En el ocupado y emergente espacio de los asistentes de código basados en IA, algunos productos, en lugar de competir con los ya fuertemente establecidos, toman un enfoque más especializado. Codium AI se especializa en la generación de pruebas con IA. Funciona en todos los lenguajes pero tiene soporte avanzado para stacks comunes, como JavaScript y Python. Particularmente nos gusta que la herramienta, en lugar de llevar a los desarrolladores directamente al código de pruebas, ofrezca una lista de descripciones de escenarios de pruebas en lenguaje natural para ser revisados. Esto facilita a los desarrolladores razonar sobre los escenarios de pruebas y decidir cuáles convertir en código. Con el fin de mejorar aún más la generación de las pruebas para una base de código y caso de uso particulares, los usuarios pueden proporcionar ejemplos de pruebas e instrucciones generales para mejorar la información utilizada por la IA en la generación de las pruebas.

70. Continue

Evaluar

Continue es un piloto automático de código abierto para los entornos de desarrollo integrado VS Code y JetBrains. Nos gusta bastante porque elimina la molestia del copiar/pegar desde una interfaz basada en chat a un modelo de lenguaje grande (LLMs) con integración directa en el entorno de desarrollo integrado. Soporta varios modelos tanto comerciales como de código abierto y vuelve sencillo probar diferentes proveedores, incluyendo LLMs autohospedados. Inclusive puedes correr Continue sin conexión a internet.

71. Fern Docs

Evaluar

Una característica distintiva de las API REST ampliamente utilizadas es que sus contratos están minuciosamente documentados. Es más probable que los desarrolladores adopten y utilicen API cuyo comportamiento y sintaxis se describen con precisión de forma estructurada y organizada. Mantener esta documentación actualizada a medida que evoluciona el contrato puede llevar mucho tiempo y es una tarea que fácilmente se pasa por alto. Fern Docs ayuda con esto al reducir el trabajo involucrado en escribir y mantener la documentación de la API. Fern genera automáticamente un sitio web con documentación atractiva y utilizable a partir de un archivo de especificación que se puede versionar junto con el código de la API. Si bien nuestras impresiones iniciales de este producto son positivas, Fern requiere que mantengas la información de la API en un archivo de configuración propietario. Si bien proporciona una forma de convertir las especificaciones de OpenAPI a su propio formato de configuración, preferiríamos una herramienta que genere documentos directamente a partir del código fuente anotado.

72. Granted

Evaluar

Dado lo común que es la estrategia multicuentas en los ambientes AWS en las organizaciones, los ingenieros frecuentemente tienen que cambiar entre múltiples cuentas en un período de tiempo corto. Granted, auna herramienta de línea de comandos que simplifica la apertura de múltiples cuentas en el navegador simultáneamente, agiliza el cambio de cuenta. Aprovecha las funcionalidades nativas del navegador para aislar múltiples identidades, Los Contenedores Multicuentas de Firefox y los Perfiles de Chromium. Si un servicio específico (como S3) es usado como un argumento, Granted abrirá la landing page del servicio. Granted actualmente sólo soporta AWS. Destacadamente, almacena las credenciales temporales de inicio de sesión único de AWS de manera segura en el llavero (keychain) en lugar de un archivo de texto plano en el disco.

73. LinearB

Evaluar

LinearB es una plataforma diseñada para capacitar a líderes de ingeniería con conocimiento guiado por datos para mejora continua. Aborda tres áreas clave: evaluaciones comparativas, automatización de flujos de trabajo e inversión. Nuestra experiencia con las herramientas para métricas de LinearB resalta su potencial para respaldar una cultura de mejora continua. Uno de nuestros equipos aprovechó la plataforma para realizar un seguimiento de las métricas de ingeniería, identificar y discutir oportunidades de mejora, y definir pasos accionables basados en datos, conduciendo a un progreso cuantificable. Nos complace ver que esto se alinea con la proposición de valor fundamental de LinearB: evaluar, automatizar y mejorar. LinearB se integra con GitHub, GitLab, Bitbucket y Jira. Ofrece una suite integral de métricas de ingeniería preconfiguradas, con un fuerte enfoque en métricas DORA (frecuencia de despliegue, tiempo de espera, tasa de fallo de cambio y tiempo de restauración). Como fervientes defensores de las cuatro métricas clave definidas por el estudio DORA, apreciamos el énfasis de LinearB en medir lo que realmente importa para el rendimiento en la entrega de software. Históricamente, obtener métricas DORA específicas ha sido un desafío. Los equipos han recurrido a compleja instrumentación en “pipelines CD”, paneles de control personalizados o depender de procesos manuales. Aunque nuestra experiencia está limitada a un equipo, LinearB parece ser una alternativa convincente para la recolección y seguimiento de métricas de ingeniería, así como para promover una aproximación a la mejora continua guiada por datos.

74. LLaVA

Evaluar

LLaVA (Asistente Visual y de Lenguaje grande, del inglés: Large Language and Vision Assistant) es un modelo multimodal en código abierto que conecta un codificador visual y un modelo de lenguaje grande (o LLM en inglés) para el entendimiento visual y lingüístico con propósito general. La gran capacidad de LLaVA en el seguimiento de instrucciones lo posiciona como un oponente altamente competitivo entre los modelos de IA multimodal. La última versión, LLaVA-NeXT, proporciona una mejor respuesta. Entre los modelos de código abierto para asistencia lingüística y visual, LLaVA es una opción prometedora cuando es comparado con GPT-4 Vision. Nuestros equipos han estado experimentando con él para responder visualmente a preguntas.

75. Marimo

Evaluar

Marimo ofrece una perspectiva totalmente diferente sobre Python "notebooks" al priorizar la reproducibilidad e interactividad. Aborda el desafío de manejar los estados ocultos en notebooks tradicionales (como Jupyter) el cual puede llevar a comportamientos inesperados e impedir la reproducibilidad. Esto lo logra, almacenando "notebooks" como archivos simples de Python sin estados ocultos y utilizando una ejecución determinista basada en el orden de las dependencias (cuando una variable cambia, todas las celdas afectadas son automáticamente ejecutadas). Además, Marimo incluye elementos interactivos de UI que propagan de una manera similar, los cambios de valores a las celdas que dependen de ellas. Al poder ser desplegada como una aplicación web, se convierte en una herramienta útil para demostraciones y creación de prototipos. Aunque, nos emociona el potencial de Marimo, particularmente en términos de reproducibilidad para exploración y análisis de datos, seguimos advirtiendo contra la producción de notebooks.

76. Mixtral

Evaluar

Mixtral es parte de la familia de grandes modelos de lenguaje de pesos abiertos que Mistral ha liberado, y que utiliza la arquitectura dispersa de mezcla de expertos (Mixture of Experts). Estos modelos se ofrecen tanto en formas puras pre-entrenadas así como afinadas, con tamaños de parámetros 7B y 8×7B. Sus tamaños, naturaleza de pesos abiertos, desempeño en evaluaciones de rendimiento y una longitud de contexto de 32,000 tokens los convierten en una opción atractiva para LLMs auto hospedados. Es importante notar que estos modelos de pesos abiertos no están afinados para ser seguros por defecto, por lo que los usuarios deben refinar la moderación según sus propios casos de uso. Tenemos experiencia con esta familia de modelos en el desarrollo de Aalap, un modelo Mistral 7B afinado y entrenado con datos relacionados a tareas legales específicas de la India, el cual ha mostrado un rendimiento satisfactorio a un costo accesible.

77. NeMo Guardrails

Evaluar

NeMo Guardrails es un conjunto de herramientas de código abierto de NVIDIA de fácil uso que permite a los desarrolladores implementar restricciones en los modelos de lenguaje de gran tamaño (LLMs) utilizados en aplicaciones conversacionales. A pesar de que las LLMs tienen un inmenso potencial para crear experiencias interactivas, sus limitaciones inherentes en torno a la exactitud

en los hechos, sesgos y su posible uso indebido requieren protecciones. Guardrails nos ofrece un enfoque prometedor para garantizar unos LLMs responsables y confiables. Aunque tiene donde elegir cuando se trata de restricciones LLM, nuestros equipos han encontrado que NeMo Guardrails es particularmente útil porque soporta reglas programables, integración en tiempo de ejecución y puede ser aplicado a aplicaciones LLM existentes sin grandes modificaciones de código.

78. Ollama

Assess

Ollama Es una herramienta de código abierto para ejecutar y gestionar modelos grandes de lenguaje (LLMs) en tu entorno local de desarrollo. Anteriormente, hemos hablado de los beneficios de LLMs auto-hospedados, y nos complace ver que el ecosistema ha madurado con herramientas como Ollama. Soporta distintos modelos populares — incluyendo LLaMA-2, CodeLLaMA, Falcon y Mistral — que puede descargarse y ejecutarse localmente. Una vez descargados, puedes usar la línea de comando, API o SDK para interactuar con el modelo y ejecutar tus tareas. Estamos evaluando Ollama y viendo los primeros éxitos a medida que mejora la experiencia de los desarrolladores al trabajar LLMs en entornos locales.

79. OpenTofu

Evaluar

OpenTofu es una rama de Terraform construida en respuesta a un cambio reciente ambiguo en la licencia emitida por HashiCorp. Es un producto de código abierto y ha sido aceptado por la Fundación Linux. Está avalado por varias organizaciones, incluyendo proveedores externos. La versión actual es compatible con la última versión de código abierto de Terraform. La versión 1.7 añade encriptación en el lado cliente. No está claro si en el futuro el proyecto OpenTofu soportará la compatibilidad con nuevas versiones de Terraform. También hay dudas respecto al soporte a largo plazo por parte de sus actuales promotores. Recomendamos permanecer atentos al proyecto pero ser cautelosos respecto al uso, excepto por equipos que tengan la capacidad de manejar riesgos que pueden incluir contribuir al código.

80. QAnything

Evaluar

Los grandes modelos de lenguaje (LLM, large language models) y las técnicas de generación aumentada por recuperación (RAG, retrieval-augmented generation) han mejorado enormemente nuestra capacidad de sintetizar y extraer información. Estamos viendo a herramientas emergentes aprovechar esto y QAnything es una de ellas. QAnything es un motor de gestión del conocimiento con una interfaz de preguntas y respuestas que puede resumir y extraer información de una amplia gama de formatos de archivos, como PDF, DOCX, PPTX, XLSX y MD, entre otros. Por motivos de seguridad de datos, QAnything también admite la instalación sin conexión. Algunos de nuestros equipos la utilizan para desarrollar la base de conocimientos de su equipo. Para escenarios de IA generativa de mayor profundidad industrial (como generar resúmenes para informes de inversión), también intentamos utilizar esta herramienta en pruebas de concepto antes de crear productos reales y mostrar el potencial de los LLMs y la RAG.

81. System Initiative

Evaluar

En los últimos años han aparecido pocas herramientas de infraestructura como código capaces de desafiar el dominio de Terraform. A pesar de que han surgido alternativas como Pulumi, CDK y más recientemente Wing, el paradigma modular y declarativo de Terraform ha probado ser el más duradero. **System Initiative** es una herramienta nueva y experimental que representa una dirección radicalmente nueva para el trabajo de DevOps. Una forma de ver a System Initiative es como un gemelo digital de tu infraestructura. Los cambios interactivos en el estado de System Initiative producen una serie de conjuntos de cambios que pueden ser aplicados en la infraestructura real. De la misma forma, los cambios en la infraestructura son reflejados en el estado de System Initiative. Una de las mayores ventajas de este enfoque es el ambiente colaborativo que crea para cosas como el despliegue de aplicaciones y la observabilidad. Se interactúa con System Initiative a través de una interfaz de usuario que tiene una representación gráfica de todo el entorno. Además de administrar la infraestructura en la nube, también se puede usar la herramienta para administrar contenedores, scripts, herramientas y más. Aunque por lo general somos escépticos de este tipo de herramientas visuales, System Initiative se puede ampliar, mediante código TypeScript, para manejar nuevos recursos o hacer cumplir políticas. Nos gusta mucho el pensamiento creativo puesto en esta herramienta y esperamos que impulse a otros a romper el status quo de los enfoques de infraestructura como código. System Initiative es gratis y de código abierto bajo una licencia Apache 2.0 y actualmente se encuentra en beta pública. Sus desarrolladores todavía no recomiendan el uso de la herramienta para producción, pero creemos que vale la pena probarla en su estado actual para experimentar un enfoque diferente en las herramientas de DevOps.

82. Tetragon

Evaluar

Tetragon es una herramienta de cumplimiento de tiempo de ejecución y observabilidad de seguridad basada en eBPF de código abierto. Mencionamos a Falco por detectar amenazas a la seguridad hace un tiempo en el Radar. Tetragon va más allá de la detección de amenazas al aprovechar eBPF para aplicar políticas de seguridad en tiempo de ejecución en el kernel de Linux. Puedes utilizar Tetragon como herramienta independiente en bare metal o dentro del entorno de Kubernetes.

83. Winglang

Evaluar

Estamos viendo mucho movimiento en el espacio de infrastructure-as-code (IaC) con herramientas como **Winglang** emergiendo. Winglang toma un enfoque diferente para definir la infraestructura y el comportamiento en tiempo de ejecución. Provee abstracciones de alto nivel sobre detalles de la plataforma expuestos por herramientas como CloudFormation, Terraform, Pulumi y Kubernetes. Con Winglang, escribes código que se ejecuta en tiempo de compilación para generar la configuración de la infraestructura, y luego, código que corre en tiempo de ejecución para el comportamiento de la aplicación. Provee un modo de simulación para ejecutar localmente y tiene un framework de pruebas integrado. Estamos vigilando esta interesante herramienta; es una vista previa potencial de la dirección futura de IaC.

Lenguajes y Frameworks

Adoptar

—

Probar

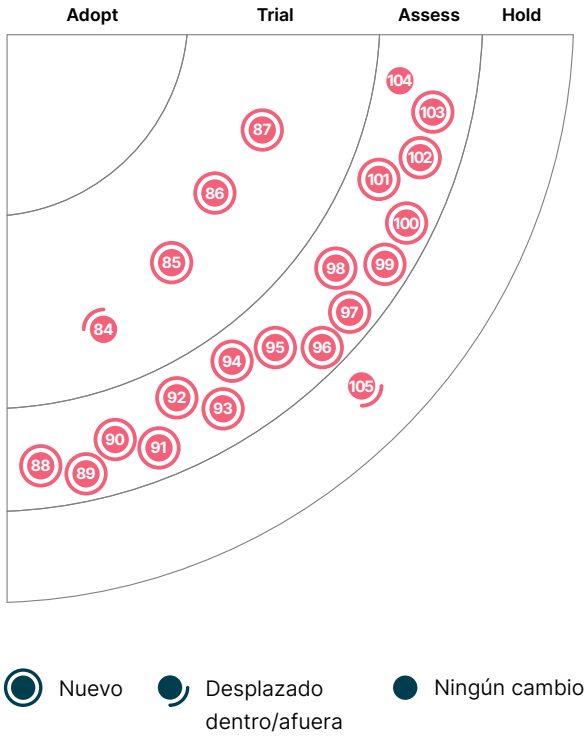
84. Astro
85. DataComPy
86. Pinia
87. Ray

Evaluar

88. Android Adaptability
89. Concrete ML
90. Crabviz
91. Crux
92. Databricks Asset Bundles
93. Electric
94. LiteLLM
95. LLaMA-Factory
96. MLX
97. Mojo
98. Otter
99. Pkl
100. Rust para el desarrollo de UI
101. vLLM
102. Voyager
103. WGPU
104. Zig

Resistir

105. LangChain



84. Astro

Probar

El marco de trabajo **Astro** está ganando más popularidad en la comunidad. Uno de nuestros equipos ha utilizado Astro para construir sitios web orientados al contenido como blogs y sitios web de mercadeo. Astro es un framework para crear aplicaciones de múltiples páginas que renderiza HTML en el servidor y minimiza la cantidad de JavaScript enviado por la red. Nos gusta que Astro soporta, cuando es apropiado, a componentes activos selectos escritos en cualquier framework de front-end de JavaScript, aunque recomienda enviar solo HTML. Esto lo hace a través de su arquitectura de islas. Las islas son regiones de interactividad dentro de una sola página donde el JavaScript requerido se descarga solo cuando se necesita. Así, la mayoría de las áreas del sitio se convierten en HTML estático, veloz, y las partes de JavaScript están optimizadas para la carga en paralelo. A nuestro equipo le gusta tanto su rendimiento para el renderizado de páginas como la velocidad en el proceso de compilación. La sintaxis para componentes de Astro es una simple extensión de HTML y la curva de aprendizaje es bastante suave.

85. DataComPy

Probar

La comparación de DataFrames es una tarea común en la ingeniería de datos, frecuentemente realizada para comparar la salida de dos enfoques de transformación de datos para asegurarse que no hayan ocurrido desviaciones o inconsistencias significativas. DataComPy es una librería de Python que facilita la comparación de dos DataFrames en pandas, Spark y más. La librería va más allá de realizar comparaciones básicas de igualdad al ofrecer información detallada sobre discrepancias a nivel de filas y columnas. DataComPy tiene también la capacidad para especificar tolerancias absolutas o relativas en comparaciones de columnas numéricas, así como diferencias ya conocidas que no hace falta resaltar en su reporte. Algunos de nuestros equipos la utilizan como parte de su suite de smoke testing; la encuentran eficiente al comparar DataFrames extensos y consideran que sus reportes son fáciles de entender y sobre los que actuar.

86. Pinia

Probar

Pinia es una librería y framework de gestión de estados para Vue.js. Utiliza sintaxis declarativa y ofrece su propia API para gestión de estados. Comparada con Vuex, Pinia provee una API simple con menos formalidades, ofrece el Composition-style APIs, y lo más importante, posee soporte de inferencia tipo sólida cuando es utilizada con TypeScript. Pinia es promovido por el equipo de Vue.js como una alternativa verificada para Vuex, y es actualmente la librería oficial de Vue.js para gestión de estados. Nuestros equipos están aprovechando Pinia por su simplicidad y fácil uso de implementación.

87. Ray

Probar

Las cargas de trabajo actuales de aprendizaje automático (ML, por sus siglas en inglés) requieren cada vez de más recursos de computación. Por muy convenientes que sean, los entornos de desarrollo compuestos de un único nodo como puede ser nuestro portátil no pueden escalar para cumplir con estas demandas. Ray es un framework unificado para escalar la IA y el código Python desde el portátil hasta el clúster. Ray es esencialmente un framework de computación distribuida bien encapsulado con una serie de librerías de IA para simplificar el trabajo de aprendizaje automático (ML). Mediante la integración con otros frameworks (por ejemplo, PyTorch y TensorFlow), puede usarse para construir plataformas de aprendizaje automático a gran escala. Compañías como OpenAI y Bytedance usan Ray intensivamente para el entrenamiento e inferencia de modelos. Nosotros también usamos sus librerías de IA como soporte para el entrenamiento distribuido y para el ajuste de hiperparámetros en nuestros proyectos. Te recomendamos que pruebes Ray cuando tengas que construir proyectos de aprendizaje automático escalables.

88. Android Adaptability

Evaluar

Algunas aplicaciones y juegos móviles pueden consumir tantos recursos que causan el estrangulamiento térmico (thermal throttling) del dispositivo en pocos minutos. En este estado, la frecuencia de la CPU y la GPU se reducen para ayudar su enfriamiento, aunque provoca la reducción de la frecuencia de imagen (frame rate) en los juegos. Cuando la condición térmica mejora, la frecuencia de imagen aumenta nuevamente y el ciclo se repite, provocando una pobre experiencia en el uso del software. **Android Adaptability**, un nuevo conjunto de librerías, permite que las personas desarrolladoras de aplicaciones implementen mecanismos de adaptación a los cambios de rendimiento y de condiciones térmicas de los dispositivos. El Android Dynamic Performance Framework (ADPF) incluye a la Thermal API para suministrar información sobre el estado térmico del dispositivo y a la Hint API para ayudar a Android a escoger de manera óptima el punto de operación y ubicación de núcleos en la CPU. Los equipos que usan Unity encontrarán útil al paquete Unity Adaptive Performance ya que funciona con ambas APIs.

89. Concrete ML

Evaluar

Anteriormente, hablamos de la técnica Encriptación homomórfica que permite ejecutar análisis y cálculos directamente sobre los datos encriptados. **Concrete ML** es una de esas herramientas de código abierto que permite aprendizaje automático con preservación de la privacidad. Construido sobre Concrete, esto simplifica el uso del encriptado totalmente homomórfico (FHE) para científicos de datos, para ayudar a convertir automáticamente los modelos de aprendizaje automático en su equivalente homomórfico. Los modelos integrados de los aprendizajes automáticos de Concrete ML tienen APIs que son casi idénticas a sus contrapartes scikit-learn. También se puede convertir redes de PyTorch a FHE con APIs de conversión de Concrete ML. Tomar en cuenta, sin embargo, que FHE con Concrete ML puede ser lento sin hardware optimizado.

90. Crabviz

Evaluar

Crabviz es un complemento para Visual Studio Code para crear gráficos de llamadas. Los gráficos son interactivos, algo esencial al trabajar con bases de código moderadamente grandes como las de microservicios, y muestran tipos, métodos, funciones e interfaces agrupadas por archivo y también presentan las relaciones de llamadas a funciones y las de implementación de interfaces. Ya que Crabviz está basado en el Language Server Protocol, soporta un sin número de lenguajes, siempre que el servidor correspondiente al lenguaje esté instalado. Al mismo tiempo esto significa que Crabviz está limitado al análisis de código estático, lo cual puede ser insuficiente en algunos casos. El complemento está escrito en Rust y está disponible en la tienda de extensiones de Visual Studio Code.

91. Crux

Evaluar

Crux es un marco de desarrollo de aplicaciones multiplataforma de código abierto escrito en Rust. Inspirado en la arquitectura Elm, Crux organiza el código de lógica empresarial en el núcleo y en la capa de UI en marcos de desarrollo nativos como SwiftUI, Jetpack Compose, React/Vue o marcos de desarrollo basados en WebAssembly (como Yew). Con Crux, puedes escribir código de comportamiento, libre de efectos secundarios, en Rust y compartirlo entre iOS, Android y la web.

92. Databricks Asset Bundles

Evaluar

La reciente versión pública preliminar de Databricks Asset Bundles (DABs), incluida con Databricks CLI versión 0.205 y superiores, se está convirtiendo en la forma recomendada oficialmente de empaquetar activos de Databricks para control de código fuente, pruebas y despliegue. Ha comenzado a reemplazar dbx entre nuestros equipos. DABs soporta el empaquetado de la configuración de flujos de trabajo, trabajos y tareas, así como el código a ser ejecutado en esas tareas, como un paquete que puede ser desplegado en múltiples entornos. Cuenta con plantillas para tipos comunes de activos y soporta plantillas personalizadas. Si bien DABs incluye plantillas para blocs de notas y soporta su despliegue en producción, seguimos recomendando no producir notebooks y en su lugar animamos a escribir intencionadamente código de producción con prácticas de ingeniería que respalden las necesidades de mantenibilidad, resiliencia y escalabilidad de dichas cargas de trabajo.

93. Electric

Evaluar

Electric es un framework local-first de sincronización para aplicaciones web y móviles. Local-first es un paradigma de desarrollo donde el código de nuestra aplicación conversa directamente con una base de datos local embebida, mientras sincroniza los datos en segundo plano a través de una replicación activa-activa con la base de datos central. Con Electric, tenemos SQLite como la opción local embebida y PostgreSQL para el almacenamiento central. Si bien el paradigma local-first mejora ampliamente la experiencia del usuario, no está exento de desafíos, y los inventores de CRDT han trabajado en el framework Electric para reducir estos dolores.

94. LiteLLM

Evaluar

LiteLLM es una librería para la perfecta integración con diversas APIs de proveedores de large language model(LLM) que estandarizan interacciones a través de un formato de API de OpenAI. ISoporta una amplia gama de proveedores y modelos y ofrece una interfaz única para funciones de acabado, incrustación y generación de imágenes funcionales. LiteLLM simplifica la integración al traducir las variables de entrada para coincidir con los requisitos específicos de la puerta de enlace de cada proveedor. Esto es particularmente valioso en el panorama actual, donde la falta de especificaciones de API para proveedores LLM complica la inclusión de múltiples LLMs en proyectos. Nuestros equipos han aprovechado LiteLLM para intercambiar modelos subyacentes en aplicaciones LLM, abordando retos de integración significativos. Sin embargo, es crucial reconocer que modelos de respuesta a prompts idénticos varían, demostrando que un método de invocación único podría no optimizar completamente el rendimiento del completado. Denotar que LiteLLM tiene distintas características, como un servidor proxy, que no se encuentran dentro del alcance de este resumen.

95. LLaMA-Factory

Evaluar

Seguimos siendo precavidos respecto a apresurarnos a afinar grandes modelos linguisticos (LLMs) a menos que sea absolutamente crítico — ya que conlleva una sobrecarga significativa en terminos de coste y experiencia. Sin embargo, creemos que LLaMA-Factory puede ser útil cuando se requiera un ajuste preciso. Se trata de un marco de entrenamiento y ajuste fácil de usar, de código abierto, para los LLM. Con soporte para LLaMA, BLOOM, Mistral, Baichuan, Qwen y ChatGLM, hace que un concepto tan complejo como el ajuste preciso sea relativamente accesible. Nuestros equipos utilizaron con éxito LLaMA-Factory's LoRA tuning para un modelo de LLaMA 7B, así que si necesitas hacer ajustes a tus modelos, vale la pena evaluar este marco.

96. MLX

Evaluar

MLX es un array-framework de código abierto, diseñado para aprendizaje automático eficiente y flexible en procesadores Apple silicon. Permite que los científicos de datos e ingenieros de aprendizaje automático (ML) puedan acceder a la GPU integrada, permitiéndoles elegir el hardware que mejor se adapte a sus necesidades. El diseño de MLX está inspirado por frameworks tales como NumPy, PyTorch y Jax, por nombrar algunos. Una de las diferencias claves de MLX, es su modelo de memoria unificada, que elimina el exceso de transferencias de datos entre CPU y GPU, resultando en una ejecución más rápida. Esta funcionalidad hace posible correr los modelos en dispositivos tales como los iPhones, abriendo una tremenda oportunidad para aplicaciones de inteligencia artificial en dispositivos. Aunque de nicho, la comunidad de desarrolladores ML encontrará que vale la pena seguirlo.

97. Mojo

Evaluar

Mojo es un nuevo lenguaje de programación de tipo AI-first. Propone cerrar la brecha entre la investigación y la producción al combinar la sintaxis y el ecosistema de Python con programación de sistemas y características de metaprogramación. Es el primer lenguaje en tomar ventaja del nuevo compilador de backend MLIR y contiene características geniales como abstracciones sin costo, ajuste automático, destrucción ambiciosa, optimización de llamadas de cola y mejor ergonomía de "una instrucción, múltiples datos" (SIMD). Nos gusta bastante Mojo y te animamos a probarlo. El SDK de Mojo está actualmente disponible para sistemas operativos de Ubuntu y macOS.

98. Otter

Evaluar

Otter es una librería de caché libre de contención en Go. Aunque Go cuenta con varias librerías de este tipo, queremos destacar a Otter por dos razones: su excelente throughput y su ingeniosa implementación del algoritmo S3-FIFO para obtener una buena tasa de aciertos de caché. Otter también admite tipos genéricos, por lo que puedes usar cualquier tipo comparable como claves y cualquier tipo como valores.

99. Pkl

Evaluar

Pkl es un lenguaje y herramienta de configuración creada para uso interno de Apple y ahora de código abierto. La característica clave de Pkl es su tipo y sistema de validación, lo que permite detectar errores de configuración antes de la implementación. Genera archivos JSON, .plist, YAML y .properties y tiene una amplia integración con lenguajes e IDEs, incluida la generación de código.

100. Rust para el desarrollo de UI

Evaluar

El impacto de Rust continúa creciendo, y muchas de las herramientas de compilación y de línea de comandos que hemos cubierto recientemente están escritas en Rust. Ahora, también estamos viendo un movimiento en el uso de Rust para el desarrollo de UI. La mayoría de los equipos que prefieren usar el mismo lenguaje para el código que se ejecuta en el navegador y en el servidor optan por usar JavaScript o TypeScript. Sin embargo, con WebAssembly puedes usar Rust en el navegador, y esto se está volviendo más común ahora. Frameworks como Leptos y sauron se centran en el desarrollo web, mientras que Dioxus y varios otros admiten el desarrollo de aplicaciones multiplataforma móviles y de escritorio además del desarrollo web.

101. vLLM

Evaluar

vLLM es un motor de servicio e inferencia de alto rendimiento y memoria eficiente para modelos lingüísticos grandes (LLM), que es particularmente eficiente gracias a su implementación de procesamiento por lotes continuos para solicitudes entrantes. Admite varias opciones de despliegue, incluyendo el despliegue de inferencia distribuida con tensores en paralelo usando Ray como servidor en tiempo de ejecución, despliegue en la nube con SkyPilot y despliegue con NVIDIA Triton, Docker y LangChain. Nuestros equipos han tenido una buena experiencia ejecutando servicios de trabajo de vLLM dockerizados en una máquina virtual on-prem, integrando un servidor API de OpenAI compatible, que a su vez se aprovecha de una variedad de aplicaciones, incluyendo complementos de IDE para asistencia en codificación y chatbots. Nuestros equipos utilizan vLLM para ejecutar modelos como CodeLlama 70B, CodeLlama 7B y Mixtral. Además es notable la capacidad de escalamiento del motor: solo son necesarios un par de cambios de configuración para pasar de ejecutar un modelo 7B a uno 70B. Si está buscando generar LLM, vale la pena explorar vLLM.

102. Voyager

Evaluar

Voyager es una librería de navegación hecha para Jetpack Compose. de Android. Soporta diferentes tipos de navegación, incluyendo Linear, BottomSheet, Tab y Nested, y su modelo de pantalla se integra con marcos de trabajo populares como Koin y Hilt. Cuando se usa Jetpack Compose en un

proyecto multiplataforma, Voyager es una buena elección para implementar un patrón común de navegación en todas las plataformas compatibles. El desarrollo en Voyager se ha reanudado y en Diciembre de 2023 la librería alcanzó la versión 1.0.

103. WGPU

Evaluar

wgpu es una librería gráfica para Rust basada en el API WebGPU, notable por su capacidad para manejar gráficos de propósito general y realizar tareas de cálculo en la GPU de manera eficiente. Esta librería pretende llenar el vacío dejado por la eliminación progresiva de estándares gráficos más antiguos, como OpenGL y WebGL. Introduce un enfoque moderno para el desarrollo de gráficos que abarca tanto aplicaciones nativas como proyectos basados en web. Su integración con WebAssembly permite además, que aplicaciones gráficas y de cálculo se ejecuten en el navegador. wgpu representa un paso adelante para hacer que la programación avanzada de gráficos avanzados sea más accesible para los desarrolladores web con una variedad de aplicaciones, desde juegos hasta la creación de animaciones web sofisticadas, posicionando wgpu como una tecnología interesante a evaluar.

104. Zig

Evaluar

Zig es un nuevo lenguaje que comparte muchos atributos con C, pero con tipificación más fuerte, asignación de memoria más fácil y soporte para espacios de nombres, entre un conjunto de otras características. El objetivo de Zig es proporcionar un lenguaje muy sencillo con compilación directa que minimiza efectos secundarios y ofrece una ejecución predecible y fácil de rastrear. Zig también proporciona acceso simplificado a la capacidad de compilación cruzada de LLVM. Algunos de nuestros desarrolladores han encontrado esta característica tan valiosa que están usando Zig como un compilador cruzado, aunque no estén escribiendo código Zig. Vemos equipos en la industria usando Zig para ayudar a construir cadenas de herramientas C/C++. Zig es un lenguaje novedoso y vale la pena investigarlo para aplicaciones donde se esté considerando el uso de C o ya se esté utilizando.

105. LangChain

Resistir

En el anterior Radar mencionamos algunas críticas sobre LangChain Desde entonces, queremos ser aún más precavidos con su uso. Aunque este marco de desarrollo ofrece un poderoso conjunto de funcionalidades para construir aplicaciones con Modelos de Lenguaje de Gran Tamaño (LLMs), creemos que es difícil de utilizar y demasiado complejo. LangChain ganó popularidad y atención muy pronto, lo que lo convirtió en la opción por defecto para muchos desarrolladores. Sin embargo, como LangChain está intentando evolucionar y mantenerse actualizado con el acelerado ritmo del cambio, se está volviendo cada vez más difícil para los desarrolladores navegar esos cambios de conceptos y patrones. También nos parece que el diseño de la API es inconsistente y verboso. Como tal, muchas veces esconde lo qué está ocurriendo realmente debajo de la superficie, haciendo difícil para los desarrolladores comprender y controlar cómo los LLMs y los varios patrones asociados a ellos funcionan realmente. Estamos moviendo LangChain al anillo de Resistir para reflejar esto. En muchos de los casos de uso, hemos encontrado que una implementación con un uso mínimo de frameworks especializados es suficiente. Dependiendo de tu caso de uso, quizás quieras considerar otros frameworks como Semantic Kernel, Haystack o LiteLLM.

Mantente al día de todas las noticias y opiniones relacionadas con Radar

Suscríbete al Radar Tecnológico para recibir correos electrónicos cada dos meses con información sobre tecnología de Thoughtworks y futuras publicaciones del Radar Tecnológico.

Suscríbete ahora

Thoughtworks es una consultora tecnológica global que integra estrategia, diseño e ingeniería para impulsar la innovación digital. Contamos con más de 10.500 empleados en 48 oficinas de 19 países. Durante los últimos 30 años, hemos logrado un impacto extraordinario junto con nuestros clientes, ayudándoles a resolver problemas empresariales complejos con la tecnología como elemento diferenciador.

/thoughtworks

Strategy. Design. Engineering.