Engineer

Prometheus

base de datos que consulta fuentes y guarda sus métricas

Qué es

Prometheus es un sistema de monitorización de código abierto cuyo núcleo es una base de datos propia de series temporales: cada pocos segundos pide por HTTP los valores actuales a los programas que observa, guarda cada valor con una marca de tiempo y permite consultar, representar en gráficos y generar alertas a partir de lo almacenado. El nombre abarca tanto el servidor por sí solo como, con la misma frecuencia, la pequeña familia de componentes que ha crecido a su alrededor.

Se creó en SoundCloud, donde el trabajo comenzó en 2012, y ahora pertenece a la Cloud Native Computing Foundation. Se incorporó en 2016 como segundo proyecto acogido por la fundación, después de Kubernetes, y alcanzó la categoría de proyecto maduro («graduated») en 2018. Prometheus 3.0, anunciado en noviembre de 2024, fue su primera versión principal en siete años. El código está escrito principalmente en Go y se publica bajo la licencia Apache 2.0.

Todo lo que almacena es una serie temporal, identificada por una métrica y un conjunto de etiquetas: pares de clave y valor que indican qué instancia de algo se ha medido, como en `api_http_requests_total{method="POST", handler="/messages"}`. Cada muestra de esa serie es un valor de coma flotante de 64 bits, o un histograma nativo, con una marca de tiempo en milisegundos. Un valor distinto en cualquier etiqueta da lugar a una serie distinta. Eso permite segmentar los datos fácilmente por host, ruta o código de estado, y también facilita sobrecargar el sistema.

Los programas exponen sus valores en un endpoint HTTP, normalmente `/metrics`, mediante una biblioteca cliente compilada dentro del código o mediante un exportador: un pequeño adaptador que traduce un sistema que Prometheus no puede instrumentar directamente, como un host Linux o una

base de datos. El servidor encuentra sus objetivos mediante descubrimiento de servicios o una lista estática y recoge sus métricas según una programación. Un trabajo por lotes que termina antes de la siguiente recogida puede, en cambio, enviar sus datos a una pasarela de recepción independiente. Las reglas de alerta se ejecutan en el servidor, y las alertas activadas llegan a Alertmanager, que las deduplica y agrupa, aplica silenciamientos y las dirige al correo electrónico, PagerDuty, OpsGenie y receptores similares.

Suele compararse con recopiladores que reciben datos enviados por los sistemas y siguen la tradición de Graphite y StatsD, con bases de datos generales de series temporales como InfluxDB y con servicios de métricas alojados. Grafana suele ser la interfaz que se pone delante. Almacena números en lugar de eventos, por lo que una línea de registro o una traza corresponde a otro lugar, y el proyecto deja claro que prioriza la fiabilidad sobre la exhaustividad: la facturación por petición, donde cada recuento tiene que ser exacto, exige otro sistema.

Utilice el concepto en una decisión

Aplique «¿Qué es Prometheus?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Monitorización y alertas que avisan cuando un fallo aún es pequeño, y cifras que muestran con qué frecuencia se mantiene así.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Primer panel de observabilidad de la empresa — la primera visión de la empresa sobre sus propios sistemas

Primer panel de observabilidad de la empresa Póngase en contacto →

Por qué importa

La decisión en la que interviene Prometheus es si operar una base de datos propia de métricas o contratarla como servicio y, si se opera una propia, qué tamaño debe tener.

Operarlo no conlleva una tarifa de licencia. El coste es una máquina y las horas de quien mantiene el sistema en buen estado. Un servicio alojado elimina ese trabajo y lo sustituye por una tarifa que crece con los datos enviados, lo que convierte cada etiqueta nueva en una línea de la factura. Para unas pocas decenas de máquinas, el servidor propio suele ser la opción más barata y sencilla. Las cuentas cambian cuando el número de series alcanza los millones y el equipo prefiere pagar antes que dedicar personal a operarlo.

Los cálculos de dimensionamiento son más favorables de lo que suele esperarse. La documentación de almacenamiento del proyecto sitúa la media entre uno y dos bytes por muestra. Mil series recogidas cada 15 segundos y conservadas durante los 15 días de retención predeterminada suman 86,4 millones de muestras, unos 86–173 MB en disco. El disco rara vez es lo que se agota. Lo es la memoria: el bloque que se está llenando reside en ella, protegido frente a una caída por un registro de escritura anticipada, y cada serie activa consume algo de memoria aunque nadie llegue a consultarla.

Cada servidor es deliberadamente autónomo. No depende de almacenamiento en red, de modo que sigue respondiendo cuando falla la infraestructura que observa, precisamente cuando más se necesita. El precio es que el almacenamiento local no está agrupado en un clúster ni replicado. La alta disponibilidad requiere dos servidores idénticos que recojan datos de los mismos objetivos, y un historial de más de unas pocas semanas exige enviar una copia mediante `remote_write` a un almacén diseñado para conservarlo.

Solicitar los datos en lugar de esperar a recibirlos también tiene una ventaja práctica. Como es el servidor quien pregunta, detecta cuándo un objetivo deja de responder y lo registra haciendo que `up` baje a cero. Además, un ingeniero puede abrir la misma página `/metrics` en un navegador para ver exactamente lo que ve el servidor, lo que permite resolver enseguida una discrepancia entre el gráfico y la aplicación.

Cómo se aplica en la práctica

En el trabajo de la fundadora, Prometheus aparece como el recopilador que alimenta un panel: en un caso, para un conjunto de servidores de bases de datos; en otro, como segunda versión de una vista que consultaba toda la oficina.

El caso de las bases de datos abarcaba un conjunto de diez servidores virtuales con Ubuntu 20.04 LTS, PostgreSQL 14 y MS SQL Server 2019. Prometheus y Grafana seguían en tiempo real la CPU, la memoria, la E/S de disco y el rendimiento de las consultas, junto con copias de seguridad automatizadas, reglas de cortafuegos y scripts específicos para analizar registros. En el conjunto de los diez servidores, la disponibilidad se mantuvo en el 99,9 %. Los hosts de bases de datos se prestan bien a la recogida periódica, porque las cifras que anticipan problemas, como un disco que se llena o una consulta que se ralentiza, evolucionan a lo largo de horas, y una muestra cada pocos segundos las detecta mucho antes de que lo haga un usuario.

La vista de la oficina muestra el paso de un recopilador artesanal a Prometheus. En un puesto anterior, la fundadora creó el primer panel de observabilidad de una organización. Empezó recogiendo por SSH medidas de CPU, RAM, disco y temperatura, entre otras métricas de servidores Linux remotos, y observaba del mismo modo los contenedores Docker. Una segunda versión se reconstruyó sobre Prometheus y Grafana en torno a las métricas elegidas con los equipos de DevOps e ingeniería: utilización de CPU, uso de memoria, disponibilidad de servicios y latencia de API, con alertas al superar los umbrales. Con el panel en funcionamiento, los equipos detectaban y resolvían los incidentes un 40 % más rápido. Consultar por SSH es un buen comienzo para unas pocas máquinas; un recopilador con etiquetas y un lenguaje de consulta es lo que permite que la misma vista siga creciendo.

Dónde surgen los problemas

La memoria aumenta después de una versión que no cambió nada importante. Alguien añadió una etiqueta cuyos valores no tienen límite: un identificador de usuario, una dirección de correo, una ruta URL completa o un identificador de petición. Cada valor nuevo se convirtió en una serie nueva, y el servidor ahora mantiene millones donde antes tenía miles. La guía de nombres advierte precisamente contra eso. Limite los valores de las etiquetas a conjuntos pequeños y conocidos, envíe los detalles de cada petición a registros o trazas y encuentre las métricas que más series generan con una consulta como esta:

```
topk(10, count by (__name__) ({__name__=~".+"}))
```

Un gráfico muestra una caída o un pico que nunca ocurrió. Un contador solo aumenta y vuelve a cero cuando el proceso se reinicia. Representar el valor bruto o restar a mano convierte cada reinicio en un desplome o una subida repentina. `rate()` reconoce los reinicios, así que calcule primero la tasa y sume después, sobre una ventana que abarque varios intervalos de recogida:

```
sum by (job) (rate(http_requests_total{status=~"5.."}[5m]))
```

Una regla de registro que suma un contador antes de calcular la tasa oculta los reinicios dentro de la suma y hace que reaparezcan las caídas falsas.

Un servicio se cae y su alerta de errores sigue en verde. Una regla como la de una tasa de errores superior a un umbral se evalúa sobre las series que existen. Cuando desaparece el objetivo, también desaparecen sus series, la expresión devuelve un resultado vacío y la alerta se resuelve sola en el peor momento posible. Genere una alerta sobre `up == 0` para cada objetivo de recogida, use `absent()` para cualquier serie que deba estar siempre presente y haga que algo externo al servidor compruebe que el propio servidor está en funcionamiento.

Las cifras del trimestre pasado ya no están cuando alguien las pide. El almacenamiento local conserva 15 días de forma predeterminada y es una única copia en un único disco. Sirve como ventana al comportamiento reciente, pero falla como archivo histórico. Aumente la retención con cuidado, envíe mediante `remote_write` a almacenamiento de larga duración lo que deba perdurar y conserve las cifras de las que depende el dinero en una base de datos que actúe como registro de referencia.

Lecturas adicionales

- Descripción general de Prometheus — La explicación del propio proyecto sobre qué es, sus componentes y los casos en los que no encaja.
- Modelo de datos — Nombres de métricas, etiquetas y muestras, y por qué cada valor de etiqueta inicia una serie nueva.
- Tipos de métricas — Contador, indicador, histograma y resumen, y las preguntas que puede responder cada uno.
- Fundamentos de las consultas — Selectores de PromQL, vectores de rango y tipos que puede devolver una expresión.
- Almacenamiento — Bloques de dos horas, registro de escritura anticipada, retención e interfaz de escritura remota.
- Nombres de métricas y etiquetas — Unidades básicas, sufijo `_total` y advertencia sobre las etiquetas de alta cardinalidad.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Prometheus?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/prometheus/