Discovery de Mapas de Aplicações

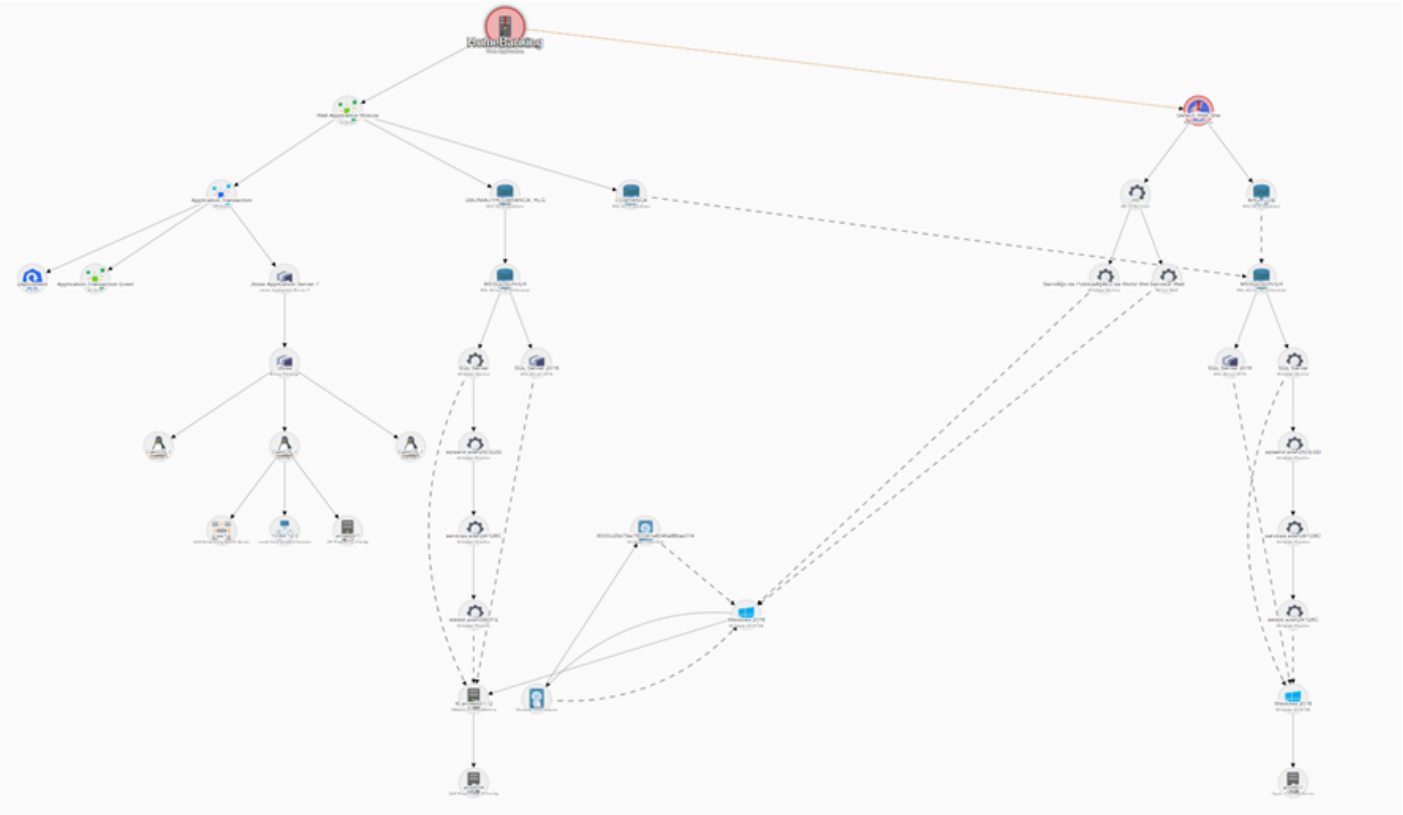
O Priax Application Mapping

O Priax Application Mapping tem em sua base o conceito de rastreamento distribuído (Distributed Tracing) que é usado por diversas ferramentas de Application Performance Monitoring (APM) para solucionar o problema de entender o fluxo de execução de uma solicitação em sistemas distribuídos, nos quais uma única solicitação é processada por vários componentes ou serviços.
A ideia central do rastreamento distribuído é que uma solicitação pode ser dividida em várias operações menores, conceitualmente chamadas de "spans", que representam unidades de trabalho individuais dentro de um sistema distribuído. Cada span registra informações sobre a operação, como seu início, duração, identificador exclusivo e quaisquer metadados relevantes. Os spans são conectados em uma árvore hierárquica, na qual o span raiz representa a solicitação inicial e os spans filhos representam operações dependentes ou subprocessos.
Essa estrutura hierárquica de spans permite visualizar o fluxo de execução completo de uma solicitação, identificar gargalos de desempenho, analisar o tempo gasto em cada operação e depurar problemas em sistemas distribuídos complexos. Além disso, cada span pode conter anotações adicionais, como logs e tags, que fornecem informações contextuais para facilitar a compreensão do comportamento do sistema.
A teoria relacionada às spans no rastreamento distribuído é fortemente influenciada por conceitos de observabilidade, como causalidade, transparência e correlação de eventos. Ao coletar e correlacionar spans de diferentes componentes, é possível reconstruir a trajetória completa de uma solicitação e obter insights valiosos sobre o desempenho e a eficiência do sistema distribuído.
O rastreamento distribuído e o uso de spans tornaram-se fundamentais para a observabilidade e o monitoramento de sistemas distribuídos modernos, permitindo aos desenvolvedores e operadores uma compreensão mais profunda do comportamento e do desempenho de suas aplicações.
Com base nessas tecnologias, o Application Mapping, que é um módulo do Priax, visa o mapeamento automático de dependências de Aplicações, correlacionando os Itens de Configuração da CMDB em função do uso que as Aplicações, ao longo de seus funcionamentos, fazem desses recursos. Para tal, o Application Mapping analisa as spans geradas pela aplicação, buscando informações sobre o uso ou consumo de recursos externos à aplicação (Itens de Configuração), para gerar o correlacionamento de dependência. Desta forma o Priax é capaz de identificar, entre outros tipos de Itens de Configuração os seguintes tipos de recursos:

- Bases (schemas ou databases) de dados em Bancos de Dados Relacionais;
- Bancos de dados não-relacionais;
- WebServices e outras aplicações web consultadas;

- Sockets de rede consultados;
- Processos (executáveis em execução) locais e remotos com os quais a aplicação troca informação;
- Tópicos ou filas de serviços de mensagens ou fluxo de dados como Kafka e RabbitMQ.

O resultado dessas análises são mapas de dependências, que correlacionam os Itens de Configuração diretamente e indiretamente consumidos pelas transações executadas pela Aplicação, formando uma árvore de dependências que explica exatamente o que cada aplicação precisa para funcionar, podendo ser usadas posteriormente para simulações de impactos e de dependências, na análise em tempo real de impactos no caso de uma falha de algum IC ou para o planejamento de continuidade de negócios e recuperação de desastres. Abaixo um exemplo de mapa gerado pelo Application Mapping.



O Application Mapping realiza também a remoção de vínculos de dependência que não mais estão presentes na aplicação, removendo as dependências que não mais se apresentaram por tempo determinado e configurável.

Funcionamento do Application Mapping

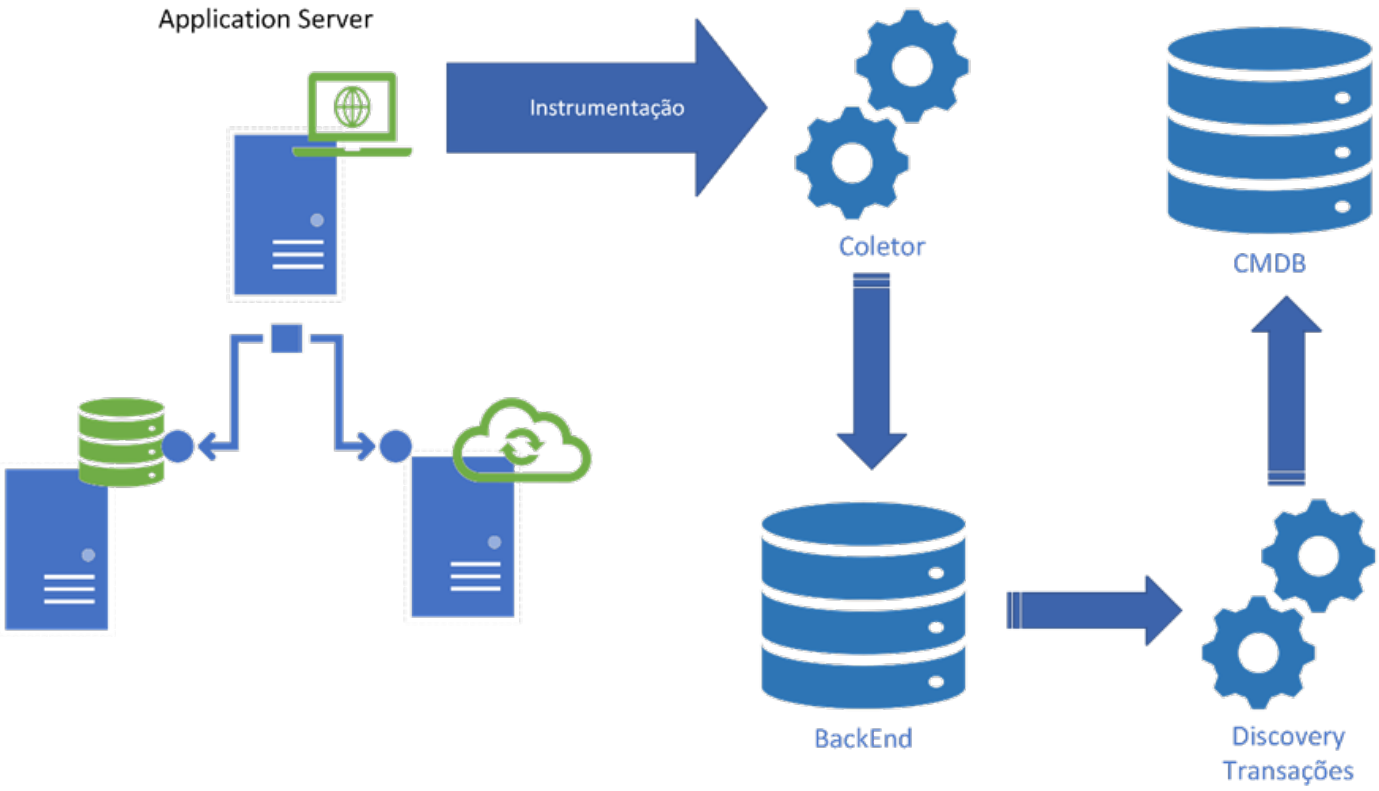
O Priax se beneficia-se do Rastreamento Distribuído de aplicações para realizar o mapeamento de Aplicações, tendo como base as spans que podem ser capturadas de diversas formas utilizando quatro componentes básicos:

1. Instrumentação: A instrumentação é o ato de adicionar bibliotecas de telemetria ao código-fonte ou em camadas inferiores de sua infraestrutura (servidores de aplicação,

proxies máquinas virtuais Java, etc.). Essas bibliotecas são projetadas para diferentes linguagens de programação e fornecem APIs para instrumentar pontos específicos do seu código. Por exemplo, você pode instrumentar chamadas de função, solicitações HTTP ou consultas de banco de dados. A instrumentação gera dados de telemetria, como spans (para rastreamento) e métricas. Para instrumentar uma aplicação podemos fazer uso de bibliotecas prontas ou alterar o código fonte das aplicações para realizar uma instrumentação própria.

2. Coletores: Os coletores são responsáveis por coletar os dados de telemetria gerados pela instrumentação. Existem coletores específicos para diferentes tipos de dados, como rastreamento, métricas e logs. Os coletores podem ser executados como agentes no mesmo host da sua aplicação ou podem ser configurados como serviços independentes.
3. BackEnds: Os dados coletados pelos coletores são enviados para backends de armazenamento, onde podem ser processados e analisados posteriormente. Os backends podem ser serviços de terceiros, como provedores de nuvem ou sistemas de armazenamento de dados internos. Eles oferecem recursos para consultar, visualizar e analisar os dados de telemetria, fornecendo insights sobre o desempenho e o comportamento do seu aplicativo. Os dados fornecidos nessa camada são utilizados para descobrir as relações entre as Aplicações e os itens de configuração presentes na CMDB Priax.
4. Discovery de Transações: O sistema de Discovery de transações é um componente do Priax que consulta e analisa os dados das spans presentes nos BackEnds utilizando inteligência artificial e análise de dados, relaciona os Itens de Configuração na CMDB, criando as transações das aplicações na CMDB como Itens de Configuração que interligam os ICs.

No desenho abaixo pode-se observar como estes componentes se relacionam.



O Priax é compatível com os seguintes mecanismos de Instrumentação:

Instrumentação	Coletores	BackEnds
		
		
		
		

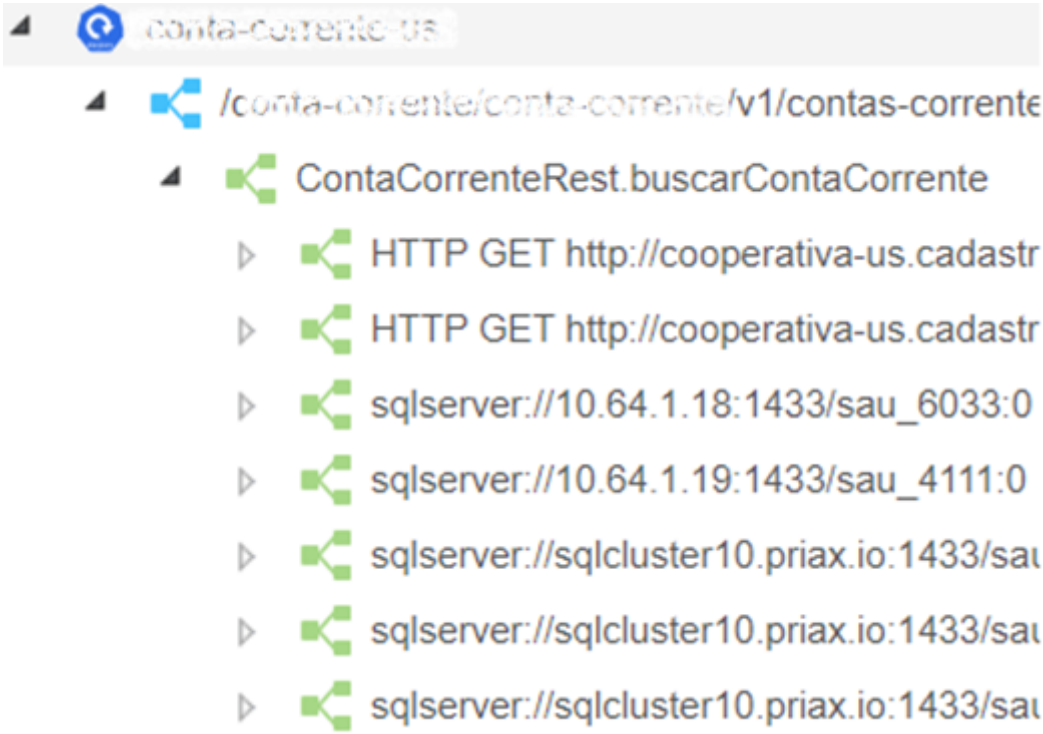
Devido aos melhores resultados obtidos, é recomendado a pilha OpenTelemetry, visto que a quantidade de informações coletadas é mais abundante e devido à melhor anonimização dos dados obtidas por essas bibliotecas de instrumentação.
Na camada de Instrumentação o Priax também é compatível com instrumentações proprietárias, desde que utilizem Coletores e BackEnds compatíveis. Desta forma é possível instrumentar uma aplicação para funcionar com o Priax sem depender diretamente de bibliotecas de terceiros em seu código-fonte. As tecnologias compatíveis com o Priax possuem especificações e API aberta que permitem implementações personalizadas para integrar a telemetria em sua aplicação.
Aqui estão algumas abordagens que você pode seguir para instrumentar sua aplicação sem utilizar bibliotecas de terceiros:

1. Implementação manual: Você pode criar sua própria implementação personalizada das APIs do OpenTelemetry em seu código-fonte. Isso envolve a criação de classes, métodos e estruturas de dados necessários para capturar e enviar os dados de telemetria, como spans, métricas e logs. No entanto, essa abordagem requer um esforço significativo de desenvolvimento e manutenção, pois você precisará lidar com aspectos como a geração de IDs de rastreamento exclusivos, a propagação do contexto de rastreamento e a integração com os coletores compatíveis.
2. Adaptação de código-fonte aberto: Se você preferir não desenvolver uma implementação personalizada do zero, pode considerar adaptar e modificar um código-fonte aberto existente que seja compatível com o Priax. Existem várias implementações de referência e bibliotecas de código aberto que você pode usar como base e personalizar de acordo com as necessidades da sua aplicação. Recomendamos neste caso as bibliotecas OpenTelemetry.
3. Integração proxies: Uma alternativa é utilizar proxies que atuam como intermediários entre sua aplicação e os coletores. Esses agentes podem interceptar as chamadas da sua aplicação, gerar spans automaticamente e enviá-los para os coletores. Essa abordagem pode ser útil se você deseja evitar a modificação direta do código-fonte da sua aplicação.

> Independentemente da abordagem escolhida, é importante garantir que você esteja seguindo as especificações compatíveis com uma das pilhas compatíveis com o Priax e fornecendo as informações de telemetria necessárias, como spans, metadados e contexto de rastreamento, para obter uma visão completa do comportamento e desempenho da sua aplicação distribuída.

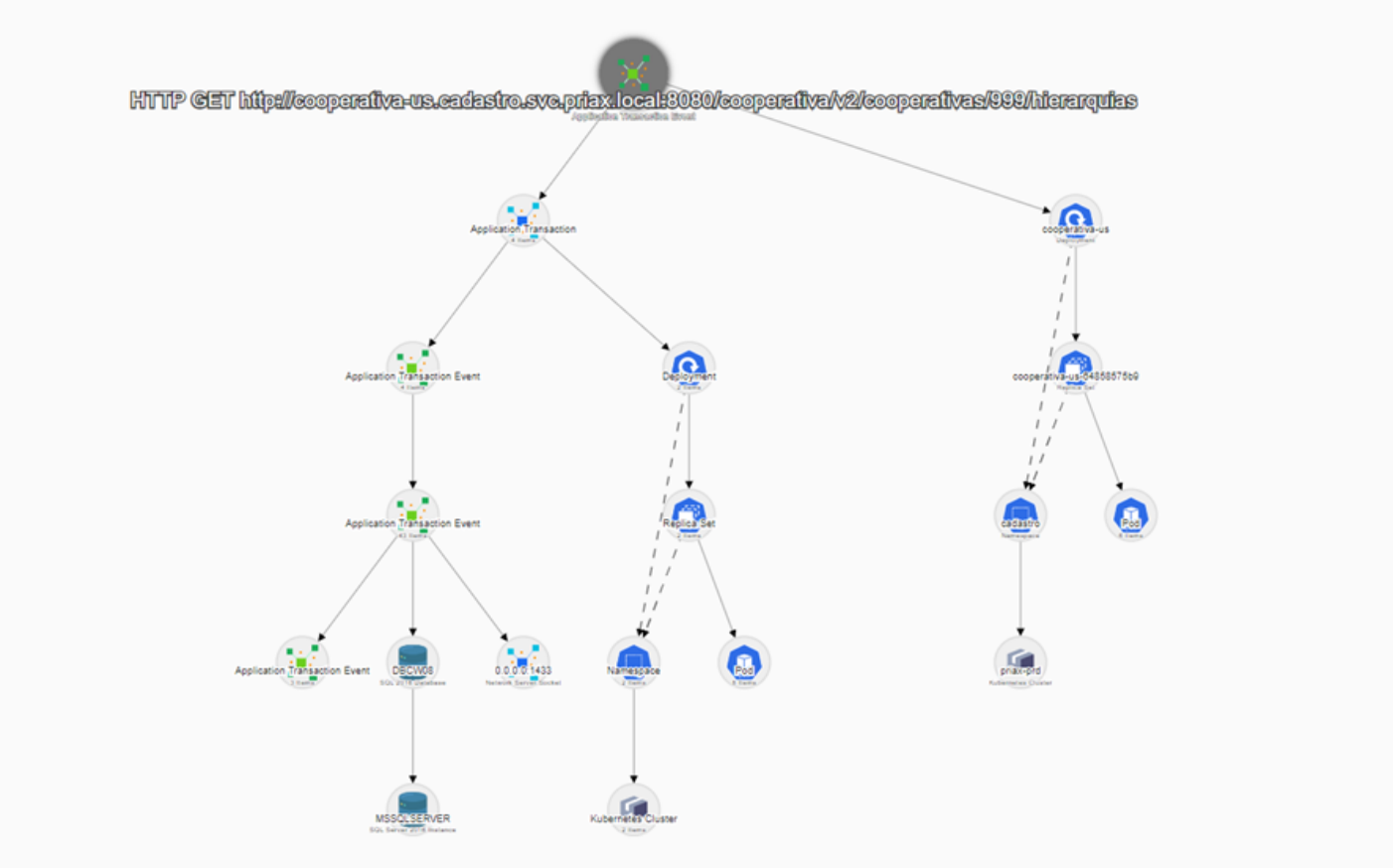
Aplicações, Serviços e Transações

O Priax Application Mapping, através do discovery de transações cria no Priax as transações das aplicações. As transações são os Itens de Configuração que representam as spans encontradas no BackEnd que apresentam funcionamento similar. Cada diferente perfil de spans gera diferentes transações no Priax. Consultas em bancos de dados com mesmo endereço de servidor, database e query e consultas à webservices com mesmo endereço e estrutura são exemplos de perfis de spans que são transformados em transações na CMDB Priax pelo Discovery de Transações. Uma transação representa centenas e milhares de repetições de execução do mesmo trecho de código fonte dentro da aplicação.
Além de detectar as diferentes transações nos BackEnds, o discovery de transações detecta quais os Itens de Configuração que foram utilizados na execução de cada transação. Desta forma se uma transação consumir uma base de dados, um tópico de um sistema Kafka, uma fila de um sistema RabbitMQ, um socket de rede com um serviço específico ou ainda um webservice, estes componentes serão relacionados com as transações.
Abaixo um exemplo de transações detectada em um cluster Kubernetes.



No topo do desenho, é representado um Deployment do Kubernetes, o qual foi instrumentado. Os elementos abaixo representam hierarquicamente as transações executadas nestes containers. Cada uma das transações consomem ICs que estão na CMDB, portanto cada transação é

relacionada com tais ICs, como pode-se ver no exemplo abaixo para uma transação de consulta a um webservice, hospedado por outro container instrumentado.



Como a transação consome um Deployment também instrumentado, veja que este Deployment também irá apresentar suas transações e dependências, chegando em uma base de dados MS SQL e nas suas respectivas dependências. A sucessão desta lógica, cria o mapa completo de dependência da aplicação.

Revision #1
Created 17 August 2024 14:31:30 by Wagner B. Simonato
Updated 14 May 2025 15:11:56 by Wagner B. Simonato