Universidade Federal de Pernambuco
Centro de Informática

Gabriel Henrique Daniel da Silva
Fagner Fernandes

Relatório do projeto final desenvolvido desenvolvido para a disciplina de Microsserviços.

Recife - PE

No nosso projeto, buscamos desenvolver uma ferramenta capaz de prover observabilidade de forma agnóstica a uma aplicação qualquer baseada em microsserviços.

Primeira Tentativa

Inicialmente, optamos por tentar implementar uma solução robusta o suficiente para ser capaz de realizar o processo de monitoramento de forma multi camadas, permitindo que tanto os nós, quanto clusters, quanto os servidores que hospedam a aplicação fossem devidamente, e até em certo ponto, redundantemente monitorados, resultando em uma ferramenta bastante resistente a erros ou problemas em algum dos níveis.

Para compor essa ferramenta optamos por utilizar uma série de outras ferramentas já existentes para simplificar o nosso trabalho, podemos dividir a seguinte stack de ferramentas utilizadas em basicamente dois tipos: Ferramentas voltadas para infraestrutura da aplicação e ferramentas voltadas para a aplicação em sí a ser desenvolvida.

Ferramentas Estruturais:

Virtualbox: Criação das máquinas virtuais que iriam hospedar tanto a aplicação, quanto a nossa ferramenta.

Vagrant: Uma ferramenta capaz de facilitar todo o processo de gerência de configuração para a criação dos ambientes virtuais.

Ansible: Uma ferramenta para automação e provisionamento, simplificando o processo de instalação de dependências e padronização no versionamento das ferramentas que compõem a aplicação.

Ferramentas Voltadas para Aplicação:

Kubernetes: Ferramenta voltada para a automatização da implantação, dimensionamento e gerenciamento de aplicativos em containers [5].

Istio: Um service mesh para microsserviços que resolve uma série de problemas como por exemplo: Load balance, monitoramento de tráfego entre serviços, controle de falhas entre outros [6].

Kiali: Um console de gerenciamento para service mesh. Capaz de inferir topologia de tráfego e prover novas métricas. Facilmente integrável com Istio, Grafana e Jaeger [7].

Jaeger: Uma ferramenta capaz de realizar o monitoramento de transações de ponta a ponta em sistemas distribuídos [8].

Além de todas essas ferramentas ainda utilizaríamos o Prometheus e o Grafana para obtenção de métricas e criação de dashboards.

Porém ao aprofundarmos mais no assunto durante o processo de desenvolvimento, nos deparamos com 2 principais problemas:

1. A nossa ferramenta, apesar de robusta seria bastante custosa no que tange a recursos computacionais, afinal teríamos uma stack com um número bastante elevado de ferramentas.
2. Encontramos enormes dificuldades referentes à integração das ferramentas dentro dessa stack de ferramentas que desejávamos utilizar.

Por conta dessas dificuldades, optamos por dar um passo para trás e revisitarmos toda a proposta da nossa aplicação, resultando em alterações na arquitetura e nas ferramentas utilizadas.

A Nossa Ferramenta

Continuamos em parte com a nossa proposta de desenvolver uma ferramenta capaz de prover observabilidade de forma agnóstica a uma aplicação qualquer baseada em microsserviços, porém optamos por focar nossos esforços apenas nos containers que irão empacotar os microsserviços da aplicação a ser observada.

A nossa ferramenta irá permitir monitorar os containers em tempo real se utilizando de algumas outras ferramentas já bastante utilizadas atualmente juntamente com um outro módulo totalmente desenvolvido por nós capaz de capturar e analisar logs dos containers da aplicação.

Ferramentas utilizadas à nivel de infraestrutura:

Docker: Uma ferramenta que permite empacotar softwares em containers, simplificando e agilizando todo o processo de deploy de uma aplicação [1]. No

nosso caso, foi de vital importância para simplificar o processo de uso das outras ferramenta utilizadas, assim como o deploy da aplicação de teste escolhida por nós (robot-shop).

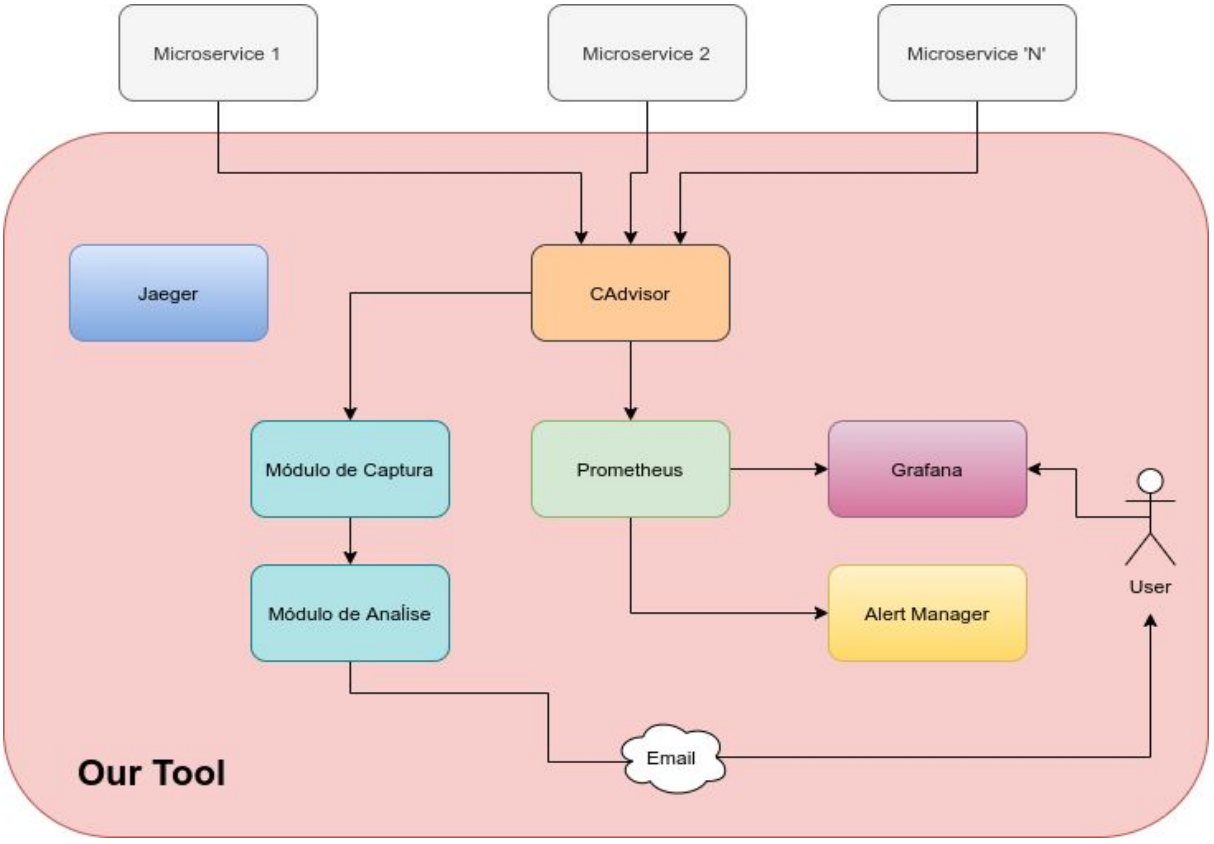
Ferramentas utilizadas à nível de aplicação:

cAdvisor: Capaz de monitorar a utilização de recursos a nível de containers, assim como outras características como por exemplo performance e estatísticas relacionadas. Possui suporte nativo à containers do Docker [2].

Prometheus: Um sistema de monitoramento open source capaz de lidar com dados em forma de séries temporais [3].

Grafana: Uma ferramenta bastante popular que é capaz de gerar gráficos e dashboards através de métricas coletadas por outras aplicações como o Prometheus e o Graphite [4].

Com a utilização das ferramentas citadas anteriormente somadas a um módulo de captura e análise desenvolvido por nós (e que detalharemos posteriormente), chegamos a seguinte arquitetura:



Módulo de Captura e Análise

O desenvolvimento desse módulo foi realizado utilizando a linguagem Python, com auxílio de uma ferramenta de desenvolvimento chamada Jupyter Notebook.

A ferramenta é capaz de realizar requisições HTTP para uma API para coletar dados que serão analisados posteriormente. A nível de teste, inicialmente estamos utilizando dados advindos da API do cAdvisor, uma ferramenta capaz de captar dados dos containers do Docker.

Uma vez que temos esses dados na nossa aplicação, criamos um dataframe e salvamos em um arquivo do tipo planilha (.csv) para serem processados pela parte do módulo responsável pela análise dos dados.

Dentro do módulo de análise, fazemos um processamento dos dados, aplicando sobre eles técnicas de aprendizagem de máquina para encontrar alguns padrões que possam ser úteis para o gerenciamento de um determinado container que está hospedando um serviço da aplicação em questão.

Inicialmente, aplicamos um algoritmo de clusterização chamado K-means que funciona da seguinte forma: Primeiramente, é passado um parâmetro com a quantidade de clusters que serão criados. A partir dessa quantidade de clusters previamente atribuída, são definidos os centróides para cada um dos clusters criados. Posteriormente atribuímos cada um dos elementos ao cluster cujo centróide for mais próximo.

Após essa etapa realizada, aplicamos um segundo algoritmo chamado Isolation Forest, que possui o objetivo de identificar anomalias nos padrões encontrados nos dados. Essas anomalias podem ser indicativos de erros ou algum tipo de problema no container em questão.

Por fim, ao detectarmos uma anomalia, imediatamente e automaticamente disparamos um email informando o usuário sobre o que foi detectado.

Conclusões Obtidas

Mesmo com as dificuldades obtidas no processo de desenvolvimento desse projeto, fomos capazes de desenvolver uma ferramenta ainda que a nível de protótipo, capaz de prover observabilidade de forma passiva e dinâmica para o usuário. Isso ocorre pois nossa ferramenta é capaz de notificar o usuário via email no momento em que ocorre a detecção de um outlier a partir do processo de captura e análise feito nos dados obtidos a partir dos containers que desejamos observar, sendo esse processo realizado sobre uma determinada métrica ou conjunto de métricas de interesse. Dessa forma, estamos simplificando a vida do usuário que antes estava limitado a um parâmetro limítrofe ou regras como no Alert Manager.

Além disso, ainda montamos uma stack com algumas das principais ferramentas de observabilidade “ativa” existentes na atualidade, contendo cAdvisor, Prometheus e Grafana. Essa stack é facilmente posta no ar utilizando apenas um comando do docker-compose.

Para efeitos de testes, utilizamos uma aplicação de teste chamada Robot Shop que é composta por vários microsserviços criados com uma grande diversidade de tecnologias de desenvolvimento [9]. Fomos capazes de subir a nossa de stack de observabilidade ativa e realizar o monitoramento nos microsserviços do Robot Shop sem maiores dificuldades. Assim como testes de utilização do nosso módulo de captura e análise, foram realizados sobre containers do Robot Shop, utilizando os dados disponíveis para consulta na API do cAdvisor e que são obtidos diretamente dos containers da aplicação de teste Robot Shop. É válido mencionar, que poderíamos mudar a aplicação alvo sem maiores dificuldades ou problemas de integração.

Possíveis Melhorias e Trabalhos Futuros

Em relação ao módulo de captura e análise, acredito que a principal melhoria a ser feita, seria uma troca de plataforma. Poderíamos sair de uma ferramenta com caráter mais “experimental” ou prova de conceito, criada com o jupyter notebook, para uma plataforma mais comercial e voltada para o usuário final, seja na forma de um App, website ou até mesmo sendo integrado a outras soluções, como por exemplo, sendo um recurso de uma plataforma de cloud.

Outra melhoria possível seria na forma de armazenamento dos dados, que atualmente é feita através de simples planilhas (.csv), e que poderiam passar a ser armazenados em bancos dados baseados em séries de dados temporais como por exemplo, no InfluxDB.

Referências

[1] https://www.docker.com/why-docker

[2] https://github.com/google/cadvisor

[3] https://prometheus.io

[4] https://grafana.com

[5] https://kubernetes.io/pt/

[6] https://www.itwonderlab.com/en/istio-in-local-kubernetes/

[7] https://kiali.io

[8] https://www.jaegertracing.io

[9] https://github.com/instana/robot-shop