
**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**
**CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA**
**UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO**
**CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**TAINAH EMMANUELE SILVA**

# UM PROCESSO PARA ADMINISTRAÇÃO DE SERVIÇOS EM CONTÊINERS DOCKER

**CAMPINA GRANDE - PB**

**2019**

**TAINAH EMMANUELE SILVA**

# UM PROCESSO PARA ADMINISTRAÇÃO DE SERVIÇOS EM CONTÊINERS DOCKER

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharela em Ciência da Computação.**

**Orientador: Professor Dr. Kyller Costa Gorgônio.**

**CAMPINA GRANDE - PB**

**2019**


S586p Silva, Tainah Emmanuele.
Um processo para administração de serviços em contêiners Docker. / Tainah Emmanuele Silva. – 2019.

11 f.

Orientador: Prof. Dr. Kyller Costa Gorgônio.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Monitoramento de contêiners. 2. Contêiners Docker. 3. Monitoramento open source. 4. Prometheus. 5. Administração de sistemas. I. Gorgônio, Kyller Costa. II. Título.

CDU:004(045)

**Elaboração da Ficha Catalográfica:**

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

**TAINAH EMMANUELE SILVA**

# UM PROCESSO PARA ADMINISTRAÇÃO DE SERVIÇOS EM CONTÊINERS DOCKER

**Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharela em Ciência da Computação.**

## BANCA EXAMINADORA:

**Professor Dr. Kyller Costa Gorgônio**
**Orientador – UASC/CEEI/UFCG**

**Professor Me. Pedro Sergio Nicolletti**
**Examinador – UASC/CEEI/UFCG**

**Professor Dr. Tiago Lima Massoni**
**Examinador – UASC/CEEI/UFCG**

**Trabalho aprovado em: 02 de julho de 2019.**

**CAMPINA GRANDE - PB**

# Um processo para administração de serviços em contêiners Docker

Tainah Emmanuele Silva

## RESUMO

Não é muito comum um administrador de sistemas monitorar serviços que são executados em contêiners Docker. A própria cultura de monitoramento dá ênfase a máquinas e muitos problemas que ocorrem nesses serviços, podem estar relacionados a falhas nos contêiners. Essas falhas impactam nos usuários que utilizam esses serviços, fazendo com que as suas tarefas atrasem ou até mesmo não sejam feitas. Uma solução para evitar essa falhas, seria implementar um sistema de monitoramento dedicado a contêiners. Assim, o sistema ao qual esses serviços fazem partem se tornaria mais confiável e evitaria os transtornos ocasionados pelas falhas. Mesmo existindo várias soluções para esse tipo de monitoramento, algumas soluções se sobressaem em relação a outras, seja por serem escaláveis e rápidas, sejam por serem já usadas em monitoramentos de máquinas. O administrador de sistemas deve ponderar as características do sistema que administra para decidir qual solução de monitoramento de contêiners deve usar.

## PALAVRAS-CHAVE

Contêiner. Monitoramento. Serviços.

**Repositório com os artefatos da solução:**

https://github.com/tainahemmanuele/monitoramento_tcc

## 1 Introdução

A administração de sistemas é uma área da computação onde alguém, denominado administrador de sistemas, possui a responsabilidade de gerenciar um ou mais sistemas, sejam eles de software, hardware, servidores ou estações de trabalho. O objetivo principal do administrador de sistemas é garantir que os sistemas estejam funcionando de maneira eficiente e eficaz. Para garantir que o sistema funcione sem problemas, o administrador de sistemas deve possuir algumas responsabilidades, como:

- manter o sistema atualizado;
- utilizar a automação de tarefas, como a instalação/atualização de software;
- manter políticas de segurança: realizar backups do sistema, e definir regras para a segurança de rede do sistema
- solucionar problemas, por exemplo : falha em serviços, problemas/falhas nas máquinas que fazem parte do sistema, problemas de rede (caso o administrador de sistemas também possua a função de administrador de rede).

Considerando as responsabilidades e as tarefas derivadas dessas responsabilidades, é possível perceber que uma organização precisa de um administrador de sistemas para assegurar que os usuários desse sistema possam executar suas tarefas sem que ocorram problemas (em um ambiente ideal). Para garantir que não ocorram problemas, uma das responsabilidades mais importantes de um administrador de sistemas é monitorar sistemas e ter a capacidade de perceber o que está ocorrendo no sistema que ele administra, a qualquer momento.

Dentre as características principais que o administrador de sistemas deve observar no seu monitoramento, estão:

- a porcentagem de recursos usados pelo sistema (CPU, Memória RAM, Disco);
- status das máquinas que compõem o sistema (verificar se estão conectadas à rede, por exemplo);

- status dos serviços que estão sendo executados no sistema;
- status da rede a qual o sistema está conectado

Em geral, a cultura de monitoramento sempre focou as sua soluções para máquinas, sejam elas físicas ou virtuais, e recursos relacionados a rede. Como os serviços usados no sistema eram instalados apenas nas máquinas, os monitoramentos usados nas máquinas conseguiam identificar problemas nesses serviços.

## 1.1 Monitoramento de contêiners Docker

Com a popularização do uso de contêiners, aqui se destaca o uso de contêiners Docker, esse cenário foi modificado. Contêiners são ambientes isolados, ao contrário de VMs, a sua virtualização faz uso do compartilhamento do kernel e os recursos de sistemas do host hospedeiro (que pode ser tanto uma máquina física, quanto uma VM). Enquanto a VM virtualiza o hardware de uma máquina, o contêiner virtualiza o SO. Por conta dessa característica, uma grande parte dos monitoramentos existentes não conseguem monitorar os contêiners, já que eles não possuem uma visão geral do ambiente físico fora do seu espaço de virtualização.

### 1.1.1 Cenário para o uso de monitoramento

O fato de não fazer o monitoramento de contêiners leva a diversos problemas. No cenário a qual sou administradora de sistemas, existem 3 VMs , virtualizadas usando o VirtualBox, com memória RAM entre 6 GB e 15GB e espaço em disco em torno de 200 GB e que executam os serviços necessários para a equipe de desenvolvimento do projeto.

Uma delas, que executa os serviços críticos para a equipe, como o Gitlab e o Sonarqube, muitas vezes trava sem um motivo aparente, pois, segundo o monitoramento da VM, não há nenhum problema que seja detectado. Ao verificar os logs dos contêiners, observa-se que há alterações em relação a comunicação do contêiner com os recursos da máquina. Utilizando um monitoramento que detecte essas falhas, é possível prevenir/corrigir os problemas que causam as indisponibilidades dos serviços automaticamente.

Os principais serviços de monitoramento usados para contêiners Docker são: cAdvisor, Prometheus, DataDog, Scout e Zabbix. No cenário apresentado anteriormente, vamos considerar as características de três dos monitoramentos citados: o Zabbix, o Prometheus e cAdvisor, pelo fato de serem open source e por não haver a possibilidade de adquirir os outros monitoramentos citados.

### 1.1.2 Analisando os Monitoramentos Open Source

Para decidir dentre as três soluções open source usadas para o monitoramento de contêiners, foram analisadas as características de cada uma das soluções open source existentes (Zabbix, Prometheus e cAdvisor). Essas características e particularidades estão na Tabela 1.

**Tabela 1: Comparação Entre Os Monitoramentos Open Source**

| Monitoramento | Desenvolvedor | Banco de Dados | Características Principais | Diferencial em relação a outros monitoramentos |
|---|---|---|---|---|
| Zabbix | Zabbix LLC | Banco de dados relacional , pode usar : Mysql, PostgreSQL, Oracle, Sqlite | Para monitorar contêiners, usa uma solução de terceiros que adiciona recursos ao seu agente de monitoramento e assim é possível visualizar as métricas dos contêiners. Nessa solução, usa templates de terceiros para coletar os dados.<br><br>Os dados armazenados no banco são usados para gerar os gráficos referentes às métricas do monitoramento. Esses gráficos são gerados na própria interface do Zabbix<br><br>O agente de monitoramento do Zabbix no cliente envia dados periodicamente ao servidor cujo tempo é definido pelo administrador de sistemas | Foco de monitoramento mais clássico, em recursos de máquina. |
| Prometheus | SoundCloud | Banco de dados não relacional, usa o que a própria aplicação possui | Usa descoberta de serviços ou configuração estática (no caso de máquinas) para fazer o monitoramento.<br>Isto é definido no seu arquivo de configuração, quais métricas devem ser coletadas dos serviços/contêiners/máquinas que ele irá monitorar<br><br>Exige que o cliente possua sua biblioteca instalada para que as métricas de monitoramento definidas no servidor funcione corretamente | Não possui um foco de monitoramento específico, consegue monitorar bem tanto recursos de máquina como recurso de serviços<br><br>Permite criação de alertas a partir das métricas coletadas. Esses alertas podem enviar notificações para o slack através de um bot, que é acionado quando os valores das métricas passam do considerado aceitável |
| cAdvisor | Google | Banco de dados não relacional, usa o InfluxDB | É um daemon em execução que coleta, agrega , processa e exporta informações sobre a execução de contêiner<br><br>Especificamente, para cada contêiner monitorado , o cAdvisor mantém parâmetros de isolamento de recursos, histórico do uso de recursos e estatísticas de rede | O foco de monitoramento são contêiners<br><br>Sua instalação só é feita através de contêiners Docker |

Analisando as características dos 3 monitoramentos e considerando o cenário dos serviços a serem monitorados, o escolhido para ser usado foi o Prometheus, porque é o único das três soluções analisadas, que permite um escalonamento entre máquinas e contêiners. Além de permitir que o administrador de sistema defina suas próprias métricas, adicione determinadas máquinas ou serviços para monitorar.

Em relação a desempenho, tanto o Prometheus como o cAdvisor possuem desempenhos semelhantes, sendo o Zabbix considerado o mais lento entre os três, uma vez que ele usa um serviço de terceiros para coletar as métricas dos contêineres e em alguns momentos, esse serviço apresenta alguns problemas como: delay entre a coleta de dados e o alerta no Zabbix. O fato do Prometheus possuir o próprio banco de dados e não depender de um banco de terceiros como o cAdvisor, torna a manutenção do serviço mais interessante para o administrador de sistemas, já que não é necessário um sistema muito complexo para garantir a eficácia do monitoramento.

## 2 Arquitetura e Projeto da Solução

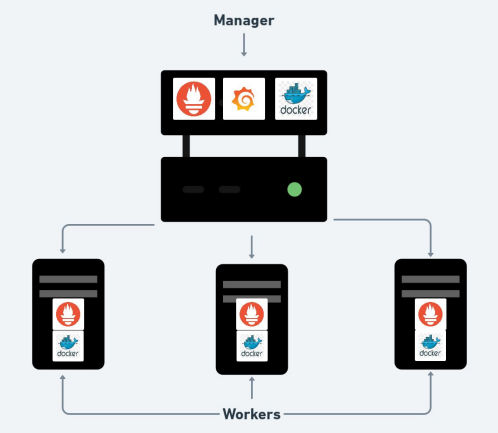


**Figura 1: Solução para instalação e uso do monitoramento de contêiners usando Prometheus**

Para instalar o sistema de monitoramento, foi utilizado a solução do Docker para escalar contêiners em diversas máquinas, o Docker Swarm. Na figura 1 há a representação gráfica da configuração a ser feita: uma máquina é considerada o manager, onde ficam os serviços que não precisam ser replicados (o Grafana e a versão Web do Prometheus). As VMs foram definidas como Workers, que são as máquinas que recebem os serviços que são replicados, nesse caso o agente do Prometheus e o agente para os alertas (bot). Tanto a máquina Manager como as máquinas Workers usam o Docker. Usar o Docker Swarm permite um melhor gerenciamento dos serviços usados para o monitoramento e caso haja falha em algum deles, se torna mais simples de detectar e resolver.

Para fazer com que a estrutura do Docker Swarm funcione e os serviços sejam instalados, usa-se a função docker stack onde a partir de um arquivo .yml com as configurações necessárias para executar os serviços do Prometheus e Grafana. Assim, foi possível instalar e configurar o serviço de monitoramento. Um detalhe importante, é que o docker stack deve ser executado de preferência na máquina manager definida dentro da pilha do Docker Swarm.

No arquivo .yml foram feitas algumas modificações nas configurações em relação ao padrão recomendado pelo Prometheus e Grafana. Houve modificações nas portas externas de ambas as aplicações, pois as utilizadas por padrão já estavam sendo usadas por outros serviços nas VMs. A versão do Grafana usada foi a 5.0.4, porque existe alguns problemas de compatibilidade na versão mais recente do Grafana em relação ao Prometheus.

Para ter uma visualização gráfica dos dados coletados pelo Prometheus, decidi usar o Grafana para plotar os gráficos. O Grafana possui suporte ao Prometheus por padrão. Os gráficos podem ser criados tanto pelo administrador de sistemas,como importado do Grafana Labs, um repositório que possui gráficos para os mais diversos tipos de monitoramento que o Grafana dá suporte. Assim, a melhor escolha é usar os gráficos oficiais do Prometheus.

## 3 Sistema em Uso

Depois de instalado todo o sistema de monitoramento, foram observados: os dados coletados pelo Prometheus; os gráficos que o Grafana ia gerando ao longo do tempo; e configurado o bot para enviar notificações para o slack. Esse bot envia notificações quando ocorre usos muito altos de recursos da máquina a qual o contêiner está sendo executado, sendo que esse uso é causado pelo contêiner.

Nos primeiros dias de uso do monitoramento, houve dois problemas , em dias distintos, na VM que possui serviços como o gitlab, redmine e sonarqube. O primeiro problema fez com que a máquina travasse. Pelos gráficos do monitoramento, foi detectado um alto uso de disco e memória, sendo que segundo o gráfico, o gitlab que estava alocando os recursos. Logo depois a VM travou. Um outro problema também foi detectado : após um alto uso de memória, a VM ficou mais lenta e os serviços demoravam um pouco a mais que o normal. Nesse caso foi o serviço do sonarqube e antes de que a VM viesse a travar e ficar indisponível, o contêiner do sonarqube foi reiniciado para normalizar o uso de recursos.

Depois da ocorrência desses problemas, o bot do Prometheus foi modificado e alguns alertas específicos acerca desses dois contêiners foram criados. Para testar se esses alertas iriam funcionar corretamente, foi escolhido como teste um serviço menos usado de outra VM e algumas operações realizadas para testar a eficácia do bot. Assim, seria possível saber se quando os contêiners do sonarqube e do gitlab desse problema, o alerta ia ser lançado corretamente.

Ocorreram outros travamentos nessa mesma VM durante todo o período de observação de eficiência do monitoramento e todos eles seguiam o mesmo padrão: sempre travavam em torno das 2:00 hs da manhã e sempre havia um alto uso de disco nessas ocasiões. Nesse horário, é executado o backup de todos os serviços da VM. O backup que dava problemas era exatamente a do gitlab, o contêiner precisava de mais recursos do que tinha disponível naquele momento.

Observando os dados coletados pelo monitoramento, detectou-se que algumas das VMs não disponibilizavam os dados corretos do uso de memória, inclusive a que travava com mais frequência. Pesquisando na documentação do Docker, isso ocorre por conta da versão do kernel do Linux da máquina, que era a versão 3.16 e nessa versão, havia alguns problemas de compatibilidade entre o kernel do Linux e o Docker. A solução mais adequada é atualizar o kernel da VM. Após essa atualização, foi possível coletar os dados referentes ao uso de memória.

Depois de atualizar o kernel, a VM do gitlab apresentou dados mais confiáveis acerca do uso de memória, inclusive que o serviço do gitlab consome 60% do total de memória da VM. Na VM haviam alguns contêiners que não estavam mais sendo usados, mas consumiam recursos. Optei por removê-los. Depois de aplicar essas soluções, não houve mais travamentos, nem mesmo durante a execução do backup.

## 4 Experiências e Lições Aprendidas

Durante o processo de instalação e configuração do sistema de monitoramento, percebi que as possibilidades de configuração do monitoramento poderiam ser mais escaláveis do que imaginei a princípio . Em alguns ambientes, pode-se usar uma solução híbrida entre o cAdvisor e Prometheus, a qual permite que o administrador de sistemas não precise definir métricas apenas no prometheus, mas já usar o modelo que o cAdvisor utiliza por padrão. Nessa situação o administrador de sistemas pode usar tanto métricas que ele próprio definiu como utilizar métricas pré-definidas.

O fato do Grafana já possuir em sua base de gráficos, gráficos pré-definidos e oficiais para uma solução de monitoramento com Prometheus, facilitou a configuração do Grafana. Ao contrário do que especifica a documentação, nem sempre se precisa ter o .json referente as configurações dos gráficos para instalá-los. Se o gráfico estiver no repositório do Grafana Labs,

basta importá-lo usando o número dele na plataforma no Grafana que está sendo configurado.

Para escrever o arquivo .yml para a criação do Docker Stack com os serviços de monitoramento, foi necessário estudar a documentação do Docker para escrita de arquivos .yml. Mesmo quem já está habituado a escrita de arquivos .yml para outras funcionalidades do Docker, como o docker-compose, precisa entender alguns detalhes na escrita do arquivo para Docker Stack. Por exemplo,o Docker Stack não permite que se defina o nome do contêiner de serviço no arquivo de criação da Stack.

## 4.1 Principais desafios e suas soluções

O primeiro desafio para instalar e configurar a solução de monitoramento, aconteceu na primeira tentativa de instalação. As documentações do Docker não explicam que se a máquina Manager do Swarm estiver executando uma versão do Docker anterior ao das máquinas Workers, haverá um problema de incompatibilidade. De forma que os serviços funcionam na máquina Manager, mas apresentam problemas nas máquinas Workers.

Para resolver esse problema, foi feita a atualização da versão do Docker no Manager. Depois de atualizar no Manager, as máquinas Workers funcionaram corretamente, mesmo estando nas versões anteriores. Para ficar todas as máquinas envolvidas no sistema de monitoramento seguindo um padrão, atualizei as VMs Workers. Assim, a conclusão é que para não ter problemas durante a instalação/configuração, é necessário certificar se todas as máquinas envolvidas possuem a mesma versão do Docker, caso não estejam, é necessário garantir que pelo menos a máquina Manager esteja atualizada.

O segundo desafio foi que o Grafana não era executado junto com todos os serviços da stack, era o único contêiner que apresentava falha. O problema estava no arquivo de configuração do Docker Stack, que estava com uma configuração relacionada a dependência com o Prometheus configurada incorretamente.

Outro desafio, já citado anteriormente, foi em relação a não identificação do uso de memória que cada contêiner estava utilizando em uma das VMs. Essa falta de dados de uso de memória poderia ser problema na versão do Docker, já que o Prometheus usa a funcionalidade “Docker Stats” para coletar os dados de uso dos contêiners.

Porém, ao pesquisar nos fóruns do Debian e na documentação do Docker é citado que o kernel 3.16 do Linux, qualquer distribuição que o usa, apresenta vários problemas e um deles é não conseguir enxergar corretamente a virtualização entre o Docker e o kernel do Linux. Além de que, caso um contêiner precise de algum recurso extra , muitas vezes o kernel não consegue virtualiza-lo e a máquina entra no que se conhece como “Kernel Panic”, travando a mesma. Essa falha foi corrigida a partir das versões de kernel 4.x do Linux. O kernel das VMs foi atualizado e após a atualização, o uso de memória passou a ser identificado pelo Prometheus.

## 4.2 Limitações

As principais limitações para que o sistema de monitoramento funcionasse corretamente esteve relacionada a versões dos serviços, a versão do Docker e o problema do kernel do Linux. Mesmo já existindo uma versão que já possui a correção do problema entre o Docker e o kernel do Linux, não há garantias que em uma versão posterior não volte a ocorrer. Da mesma forma, existem versões do Docker em que a Docker Stack não funciona, aparece a seguinte mensagem de erro: “*error response from daemon network not found*” porque não consegue configurar a rede interna de comunicação entre os contêiners do Swarm. Várias versões, como a 1.13 do Docker apresentaram bugs ao tentar conectar contêiners, seja não criando a rede, seja criando, mas não conseguindo conectá-la.

Outra limitação existente está entre a comunicação do Prometheus com o Grafana.

Algumas atualizações recentes do Grafana apresentaram alguns problemas na conexão com o Prometheus, não conseguindo fazer com que os serviços se comunicassem entre si. Essa questão de atualização também limita o uso de gráficos já existentes na base de dados do Grafana. Alguns gráficos exigem versões específicas tanto do Prometheus como do Grafana.

## 5 Conclusões e Trabalhos Futuros

Usar monitoramentos em serviços que são executados em contêiners Docker, permite que o administrador de sistemas possa entender e solucionar problemas de forma eficaz. Nem sempre os problemas nos serviços estão relacionados diretamente com o fato dele ser executado em um contêiner, mas com a comunicação entre o Docker e o kernel da máquina a qual o serviço está hospedado.

Há também a questão do serviço/aplicação não usar os recursos de máquina de forma satisfatória. Como o uso de contêineres isola os dados referentes ao uso desses recursos por serviço, o uso de monitoramento permite que seja possível ver os problemas da aplicação. Dessa forma, se torna possível relatar aos desenvolvedores das aplicações esses problemas e desenvolver soluções para que as aplicações tenham melhor desempenho.

O uso de um monitoramento escalável, caso do Prometheus, permite que a solução de monitoramento possa se adaptar nos mais variados tamanhos de sistemas: de um sistema simples até um sistema mais complexo. Permitir que o administrador de sistemas possa criar suas próprias métricas de monitoramento torna o Prometheus a melhor solução open source existente atualmente.

Instalar e usar o monitoramento no ambiente a qual sou administradora de sistemas, permitiu detectar falhas nos serviços e as causas dessas falhas. A partir dos problemas apontados no monitoramento e a aplicação de soluções, eles foram resolvidos. Não houve mais impactos aos usuários do ambiente, que muitas vezes tiveram que atrasar suas tarefas, por conta das falhas dos serviços.

Posteriormente, a ideia é expandir o uso do monitoramento as outras máquinas existentes no ambiente usado para a instalação do monitoramento e que também executam seus serviços em contêiners Docker. Aumentar os serviços usados: usar uma solução com Prometheus e cAdvisor com uma solução para armazenamento de logs : o elasticsearch. Assim, além de métricas relacionadas a uso de recursos, pode-se ter os logs que mostram quando esses recursos estão sendo usados de forma excedente, estão apresentando problemas etc. Para melhorar ainda mais a eficácia do monitoramento, seria viável aumentar a quantidade de máquinas Manager, para garantir que caso haja alguma falha, haverá outra máquina que continuará coletando os dados de monitoramento e enviando os alertas ao administrador de sistemas.

## AGRADECIMENTOS

Agradeço ao meus pais, Assis e Lígia e a minha irmã Gabriela pelo apoio durante toda a graduação e por estarem do meu lado nos momentos mais difíceis dessa jornada. Agradeço a Adailton Cavalcante, amigo da minha família e uma das pessoas que mais me incentivou a seguir na carreira de computação, desde quando fiz o ensino médio no IFPB até hoje, sempre me incentivando a continuar no curso. Agradeço a Graça Teófilo, minha madrinha , que sempre me apoiou durante toda a graduação.

Agradeço ao professor Kyller Costa Gorgônio por ter sido a primeira pessoa a ter me dado uma chance para trabalhar na área de administração de sistemas no Embedded, em 2013 , e posteriormente, ter aceitado ser meu orientador neste TCC.

Agradeço ao professor Tiago Massoni por em 2015 ter me dado a oportunidade de trabalhar como administradora de sistemas no Splab e a fazer parte do projeto ePol. Agradeço a todos os integrantes do projeto ePol, e aos professores coordenadores : João Arthur, Dalton

Serey , Franklin Ramalho , Jorge Abrantes e a Tiago Massoni pela oportunidade de ter participado de um projeto como o ePol, que me fez aprender muito sobre administração de sistemas.

Agradeço ao professor Matheus Gaudencio pelos ensinamentos da área de administração de sistemas durante todo esse tempo que estou no Splab e durante toda a graduação. Por fim, agradeço a todos os meus amigos e aqueles que estando próximo a mim ou até mesmo distante, sempre me apoiaram nessa jornada.

## REFERÊNCIAS

[1] JESHEEN , Hanzel Monitoring Docker Swarm with cAdvisor, InfluxDB and Grafana. 2017. Disponível em : <https://botleg.com/stories/monitoring-docker-swarm-with-cadvisor-influxdb-and-grafana/> Acesso em:103 jun 2019

[2] KOUTOUPIS, Petros.Taking System Monitoring to the Next Level: an Interview with Scalyr CEO Steve Newman. Linux Journal. Issue 292, p. 130 -136, nov. 2018

[3] MOUAT, Adrian. Usando Docker - Desenvolvendo e Implantando Software com Contêiners. 1.ed. Novatec. p. 19- 22 ; 230 - 238; 294 - 301, abril.2016

[4] TRABELSI , Emma. Top Docker Monitoring Tools. 2018. Disponivel em: <https://code-maze.com/top-docker-monitoring-tools/> Acesso em: 03 jun 2019

[5] O que é Zabbix ?. Disponivel em: <https://www.4linux.com.br/o-que-e-zabbix> Acesso em: 03 jun 2019

[6] Monitoring Docker Container Metrics Using Cadvisor. Disponivel em: <https://prometheus.io/docs/guides/cadvisor> Acesso em:03 jun 2019

[7] System administration. Disponivel em: <https://en.wikiversity.org/wiki/System_administration> Acesso em: 03 jun 2019