Ivan de Moura Miranda

Projeto de Automação de Operações e Entrega Contínua para Empresa de Desenvolvimento de Software.

Conselheiro Lafaiete

2016

Ivan de Moura Miranda

Projeto de Automação de Operações e Entrega Contínua para Empresa de Desenvolvimento de Software.

Relatório técnico apresentado como trabalho de conclusão de curso para obtenção do diploma do curso de Engenharia de Computação, Faculdade Presidente Antônio Carlos de Conselheiro Lafaiete.

Faculdade Presidente Antônio Carlos – FUPAC

Engenharia de Computação

Prof. Me. Jean Carlo Mendes

Conselheiro Lafaiete

2016

Resumo

Este trabalho descreve um projeto de otimização do processo de entrega de uma empresa de desenvolvimento de softwares. O objetivo é identificar as etapas de baixa produtividade para implementar ferramentas e aplicar metodologias que melhoram a qualidade dos softwares desenvolvidos. Durante o processo de desenvolvimento diversas tarefas repetitivas são executadas manualmente, com a automatização destas conseguimos reduzir o tempo investido no projeto, e com a aplicação de boas práticas de desenvolvimento, também garantimos melhor qualidade dos produtos desenvolvidos.

Palavras-chaves: entrega contínua, integração contínua, *devops*, metodologias ágeis.

Lista de tabelas

Lista de abreviaturas e siglas

AWS	*Amazon Web Services*
BaaS	*Back-end as a Service*
bug	*Defeito, falha ou erro no código de um programa que provoca seu mau funcionamento.*
deploy	*Implantação de uma versão do software em ambiente de produção*
devops	*Development and Operations*
IaaS	*Infrastructure as a Service*
PaaS	*Platform as a Service*
SaaS	*Software as a Service*
SSH	*Secure Shell*

Sumário

1 Introdução

Segundo uma pesquisa publicada pelo Stack Overflow (2016), dentre os maiores desafios encontrados por desenvolvedores em seu ambiente de trabalho vemos como mais influentes a documentação ruim, os requisitos mal especificados e o processo de desenvolvimento ineficiente.

Diante dessa realidade percebemos que o ambiente de desenvolvimento de softwares é negativamente afetado pela ineficiência do processo que muitas das vezes é excessivamente burocrático e pouco produtivo.

> Mas ser burocrático não é o problema, pois as regras são necessárias para nortear qualquer tipo de trabalho. O problema reside, é claro, em ser excessivamente burocrático, gerando atividades/produtos que não agregam valor e consomem desnecessariamente tempo da equipe. (E-BUSINESS, 2015)

Em busca de resolver estes e outros problemas, várias metodologias foram criadas, como os métodos ágeis, a cultura *devops*, e as ferramentas que objetivam a automação dos processos de operação, permitindo a entrega contínua de software com a mínima intervenção humana, em busca de agilidade e qualidade do sistema entregue. O problema é que essa automação exige conhecimento de muitas tecnologias e ferramentas diferentes, o que é visto na maioria das vezes como barreira por desenvolvedores que não tem familiaridade com especificações de infraestrutura.

De outro lado, várias empresas viram uma oportunidade de negócio atrelada a resolução destes problemas, e diversas plataformas começaram a surgir nos últimos anos, como o *Developer Cloud Service* (ORACLE, 2016), *Open Shift* (RED HAT, 2016) e o *AWS Lambda* (AMAZON, 2016). Estas soluções buscam facilitar a vida dos desenvolvedores oferecendo infraestrutura, plataforma ou *back-end* como serviço para as empresas de desenvolvimento, e algumas ainda oferecem ferramentas para gestão dos projetos do time de desenvolvimento.

Apesar destas soluções prontas, o fato curioso é que, se não levarmos em conta o hardware e considerarmos apenas as ferramentas de automação fornecidas, boa parte delas são gratuitas e de código fonte aberto como o Git, Jenkins e Kubernetes. O próprio Open Shift por exemplo possui uma versão de código aberta que pode ser instalado na sua própria empresa com acesso a todos os recursos. Então a vantagem em contratar esse tipo de serviço se deve apenas a hospedagem na nuvem e ao baixo custo de manutenção dessas ferramentas que dispensam a necessidade de um profissional especializado contratado pela sua empresa, porém este custo ainda não é totalmente viável para micro e pequenas

empresas, que de acordo com uma pesquisa realizada pela ABES (2016) representam 95,1% das empresas de desenvolvimento e produção de tecnologia no Brasil.

Diante desse cenário, boa parte do mercado nacional pode se sentir desmotivado a contratar um serviço desse tipo, pois é muito difícil prever o benefício real proporcionado, afinal, o custo do investimento é fácil calcular, mas o lucro estimado é complicado, uma vez que eficiência e eficácia precisam ser convertidas em dinheiro baseadas na confiança de que a automação irá realmente resolver os problemas da empresa.

Uma alternativa a este investimento ainda existe, que é a capacitação profissional do time de desenvolvimento para a utilização das ferramentas gratuitas em servidores locais, que não precisam ser computadores muito potentes devido a baixa necessidade de escalabilidade e performance, uma vez que os usuários desses sistemas serão os próprios colaboradores da empresa.

Para implantação desse ambiente automatizado que utiliza apenas ferramentas gratuitas, é necessário o estudo de uma gama de soluções a fim de identificar a que mais atende a real necessidade da empresa.

1.1 Objetivos

Em uma tentativa de validar os benefícios da automação e entrega contínua num cenário real, graças ao apoio da DevelOP, uma empresa de desenvolvimento de software situada em Ouro Preto, este trabalho se propõe a implementar um ambiente de desenvolvimento integrado baseado no estudo de boas práticas e ferramentas gratuitas existentes no mercado buscando a automação das tarefas de operações e o aumento da qualidade dos softwares desenvolvidos. Objetiva-se também avaliar a eficácia dos métodos propostos graças a uma analise da performance da empresa antes e depois da implementação do projeto.

1.2 Organização dos capítulos

Parte I

Preparação do relatório

2 Metodologia

A fim de avaliar os benefícios das soluções propostas, este trabalho foi fundamentado na investigação a respeito do tema. As situações referentes ao objeto de estudo, que no caso se trata do processo de desenvolvimento de softwares da empresa DevelOP, foram examinadas com olhar investigativo buscando atingir a maior veracidade possível.

O projeto aborda conhecimentos a respeito da gerência de projetos, desenvolvimento e entrega de softwares e para isso se fez necessário direcionar a abordagem com base na utilização de material teórico, estabelecendo uma linha de investigação pela qual foi conduzido o trabalho para levantar todo o material necessário com o intuito de estabelecer uma avaliação prática dos resultados obtidos.

O projeto foi divido em 3 etapas. A primeira se baseou no estudo do cenário atual da empresa coletando e analisando dados em busca de identificar pontos problemáticos e levantar propostas de soluções. A segunda fase teve por objetivo implementar as soluções propostas e analisar seus efeitos no ambiente da empresa. A terceira fase teve a responsabilidade de mostrar os resultados e avaliar as soluções oferecidas e as vantagens que propõe.

Este trabalho atuou utilizando o método experimental de forma a estudar os fatores que influenciam o tempo e a qualidade do processo de desenvolvimento, desta forma avaliando a veracidade das hipóteses levantadas.

2.1 Procedimentos para coleta e análise dos dados

A coleta dos dados foi feita através do acompanhamento das rotinas da empresa para que o conhecimento da área no qual está focado o tema seja ampliado e permitindo uma análise mais enfática quando se necessita recolher informações de variados aspectos relacionados, assim preenchendo melhor as lacunas no projeto.

A análise dos dados foi feita através do comparativo entre o antes e depois da implementação das soluções propostas. Comparando o tempo necessário para entregar uma nova versão do software, a relação entre a quantidade *builds* quebradas e bem sucedidas por dia, a quantidade de *bugs* reportados por semana, a facilidade de manutenção do código e a satisfação dos desenvolvedores com o processo de desenvolvimento.

3 Análise do cenário atual

3.1 Associação de Desenvolvimento de Software, Produtos e Pessoas da Região dos Inconfidentes Mineiros - DevelOP

A DevelOP é uma associação privada fundada em 2016 por um grupo de profissionais em tecnologia da informação que desenvolve softwares, produtos e pessoas na região dos inconfidentes mineiros. (Pegar mais informações sobre a fundação com o Álvaro)

Situada no centro de Ouro Preto, a empresa tem como principal atividade econômica o desenvolvimento de programas de computador sob encomenda, mas também atua em atividades de desenvolvimento e licenciamento de programas de computador customizável, consultoria em tecnologia da informação e atividades de apoio à educação.

(Visão e Missão.)

Apesar de ser uma empresa nova no mercado, a DevelOP já possui alguns projetos em seu portfólio, como o evento Empreenda. Em Ação! realizado em julho de 2016 onde equipes formadas por alunos de disciplinas de Empreendedorismo da Universidade Federal de Ouro Preto (UFOP) competem na idealização, planejamento e apresentação de modelos de negócio reais e o atual projeto em desenvolvimento, onde a empresa presta serviços a um cliente de São Paulo atuando na automação de processos de extração de conhecimento em Diários Oficiais.

3.2 Estrutura organizacional

As atividades econômicas desenvolvidas pela empresa apresentam características que exigem a divisão de times em projetos, mesmo que alguns funcionários algumas vezes participem de mais de um projeto simultaneamente, como é o caso dos analistas de negócio por exemplo, a empresa se organiza para alocar os recursos temporariamente até sua liberação.

Atualmente a empresa trabalha com especialistas em áreas definidas, como desenvolvedores, administradores, designers, analistas de negócio e jornalistas. A maioria desses áreas é preenchida com apenas um profissional responsável por responder pelas demandas relacionadas ao seu domínio. Quando há a alocação de recursos para projetos, o responsável pela gerência do projeto analisa quais áreas serão relacionadas em que momento, e então é feita a alocação dos times.

A comunicação é feita de maneira informal devido a proximidade dos funcionários

e ao pequeno quadro de colaboradores.

3.3 Recursos disponíveis

O time de desenvolvimento é composto por um desenvolvedor pleno e três estagiários incluindo o autor deste relatório. As áreas de design, análise de negócios e jornalismo contam, cada uma, com apenas 1 funcionário.

Devido a parceria que a empresa tem com a Universidade Federal de Ouro Preto, algumas vezes por ano ela recebe inter cambistas de experiência e área de atuação diversificada.

Dos recursos que são relevantes para o objeto de estudo deste trabalho, a empresa terceiriza o serviço de hospedagem, contando com dois servidores dedicados cuja especificações podem ser encontradas na Tabela 1.

Tabela 1 – Especificações técnicas dos servidores da empresa

	Servidor A	**Servidor B**
CPU	Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz	AMD Opteron(tm) Processor 3280
Memória RAM	32 GB	32 GB
Disco rígido	500 GB	500 GB

3.4 O processo de desenvolvimento

As metodologias envolvidas nos projetos da empresa são inspirados em métodos ágeis, alguns utilizam SCRUM e outros apenas Kanban, isso varia de acordo com os requisitos e recursos disponíveis para cada projeto, mas de forma geral, a empresa sempre busca manter um quadro de atividades (Kanban) e realizar reuniões frequentes.

Através do acompanhamento do processo de desenvolvimento durante o período que este trabalho foi realizado, identifica-se um ciclo baseado em definição de requisitos, implementação e testes manuais. Inicialmente o coordenador do projeto realiza uma reunião com o time para especificar quais funcionalidades são prioritárias e detalhar o que precisa ser feito para seu desenvolvimento. Durante essa reunião é feita a divisão de atividades e a estipulação do tempo necessário por cada responsável. Uma pessoa é encarregada pelo registros das atividades no software de gestão, e em seguida cada membro do time começa a trabalhar na tarefa que lhe foi atribuída.

Diariamente o time realiza reuniões de acompanhamento, onde informam seu progresso ao coordenador do projeto e tiram dúvidas com os demais colaboradores. Caso uma atividade seja concluída, uma nova pode ser atribuída.

Durante o desenvolvimento, os programadores trabalham em *branches* individuais com auxílio de um sistema de controle de versões. Eles testam manualmente cada funcionalidade em seu computador, e quando o time chega ao consentimento que existe um conjunto de funcionalidades pronto para ser entregue, algum membro do time compila o projeto em seu computador e realiza o envio do artefato gerado via SSH para o ambiente de produção. Durante essa etapa de entrega, caso a versão do artefato gerado apresente *bugs*, o desenvolvedor responsável pela tarefa o corrige, gera uma nova versão e a envia novamente ao servidor de produção. Esta tarefa pode se repetir enquanto houverem *bugs* que impossibilitem o correto funcionamento da aplicação.

Caso as características do artefato gerado não influenciem negativamente no processo dos usuários do sistema, um teste manual é realizado no ambiente de produção.

Quando surgem demandas imprevistas durante o ciclo de desenvolvimento, uma reunião é convocada para re-ajustarem o cronograma semanal e redistribuírem as tarefas. Caso um *bugs* seja reportado pelo cliente, ele é avaliado quanto a sua criticidade e comparado à fila de tarefas pendentes. Caso seja um bug crítico, é tratado imediatamente, caso não seja critico e existam atividades prioritárias na fila, ele é registrado e levado em pauta na próxima reunião, onde o time reajusta o cronograma.

3.4.1 Dados coletados com a análise

A etapa de coleta de dados analíticos sobre o atual processo de desenvolvimento teve início no dia 8 de agosto de 2016. Durante essa etapa foram identificadas tarefas que fogem ao escopo de desenvolvimento e levantamento de requisitos, como atividades de compilação do código fonte, execução de testes manuais, configuração de ambientes de trabalho, etc. As execuções dessas atividades foram registradas a fim de sumarizar o tempo necessário e a quantidade de retrabalho envolvido. O acompanhamento e registro desses dados ocorreu até o dia 23 de setembro de 2016.

As seguintes tabelas apresentam dados quantitativos referentes as etapas do processo de desenvolvimento durante a fase de coleta. Graças a essas informações podemos identificar pontos de aprimoramento em potencial.

A Tabela 2 apresenta dados analíticos referentes aos repositórios de código fonte da empresa.

A Tabela 3 demonstra o tempo necessário por entrega de uma nova versão do sistema. Consideramos o tempo necessário para construção de um artefato executável e a sua disponibilização em ambiente de produção. Esta etapa engloba tentativas má sucedidas de construção e entrega de uma versão estável e sem *bugs*.

Após a coleta de dados, uma entrevista foi realizada com o time de desenvolvimento onde foi solicitado que cada um citasse problemas marcantes que aconteceram com eles

Tabela 2 – Dados do repositório de código fonte no início do projeto

Métrica	**Valor**
Número de *commits* realizados	125
Número de artefatos gerados	65
Número de mesclagem de código realizados	6
Número de *bugs* reportados pelo cliente	20
Número de atividades desenvolvidas pelo time que foram registradas no sistema de controle de atividades	70

Tabela 3 – Características das entregas no início do projeto

Data da entrega	**Tempo necessário**	**Número de tentativas má sucedidas até a conclusão da entrega**
16/08/2016	2 horas	0
23/08/2016	6 horas	3
30/08/2016	3 horas	1
07/09/2016	6 horas	5
14/09/2016	8 horas	7
19/09/2016	4 horas	2

no decorrer do projeto com o objetivo de compreender a visão que o time tem sobre o processo de desenvolvimento em busca de identificar aspectos impactantes na rotina de trabalho que precisam ser melhorados. As seguintes respostas foram obtidas:

- Algum membro do time esqueceu de apagar um diretório que estava em lugar indevido, o que ocasionou a sobrecarga do disco-rígido do servidor do cliente, pois havia uma unidade de baixa quantidade de armazenamento disponível alocada apenas para o sistema operacional, e esta ficou inoperante, exigindo a investigação e remoção dos arquivos indevidos.

- Em uma tentativa de entrega de uma nova versão, foi identificado a existência de um diretório chamado "teste"no ambiente de produção. Consultando o time sobre a necessidade daquele diretório e na tentativa de apaga-lo do sistema pois não deveria estar ali, foi comunicado que o ambiente de produção estava neste diretório e não no lugar correto onde deveria ser executado.

- Em uma tentativa de entrega foi identificada a configuração indevida de permissões de um certo diretório feita por outro membro do time.

- Por diversas vezes o time identificou problemas relacionados à implantação de uma nova versão devido a arquivos de propriedades configurados incorretamente.

- Vários problemas já aconteceram devido a execução de *scripts* no ambiente de produção sem os devidos testes serem realizados previamente.
- Diversos problemas aconteceram devido a necessidade da configuração manual do ambiente de produção, uma vez que diretórios precisavam ser manipulados por linha de comando.

3.4.2 Características do processo atual

Baseado nos dados coletados durante a primeira fase do projeto identificamos como principais problemas o tempo necessário para entrega de uma nova versão estável do sistema e a inexistência de uma garantia de qualidade sobre os artefatos entregues. A qualidade neste contexto pode ser melhorada aumentando a capacidade do software atender a necessidade de mudanças, o que poderá ser identificado pela redução da quantidade de *bugs*.

Segundo Reisswitz (2013), a qualidade de um software está diretamente relacionada à qualidade do processo de desenvolvimento, dessa forma, é comum que a busca por um software de maior qualidade passe necessariamente por uma melhoria no processo de desenvolvimento.

Relacionando as dificuldades reportadas pelo time com os problemas levantados, identificamos que as atividades que não são totalmente voltadas ao desenvolvimento nem ao levantamento de requisitos são os principais causadores dos problemas, provavelmente pela falta de especialistas em infraestrutura e pela falta de automação das atividades de implantação.

Os problemas citados se mostram relevantes uma vez que já foram identificados por especialistas, como Silveira et al. (2011, p. 136) que descrevem um ambiente com características similares ao ambiente da empresa alvo de estudo deste trabalho como um ambiente problemático e sujeito a diversas falhas.

A partir de todos esses dados levantados, nossa linha investigativa foi elaborada em busca de reduzir o tempo necessário para implantação de novas versões do código, estabelecer métricas de qualidade, garantir que a versão de produção é estável e reduzir o número de versões defeituosas entregues pelo time.

4 Procedimentos Experimentais

4.1 Levantamento de metodologias e boas práticas

4.2 Adaptando a rotina do time para atender as demandas da empresa

4.3 Levantamento comparativo de ferramentas

Para elaboração de um ambiente de desenvolvimento com tarefas de operações automatizadas, visando aumento da produtividade, economia de tempo e garantia de qualidade, foi realizado um levantamento comparativo entre ferramentas gratuitas ou de código fonte aberto que possibilitem a aplicação de todos os métodos necessários especificados pela nova metodologia de trabalho.

É importante que as ferramentas sejam integradas para agilizar a comunicação do time e que tenham interfaces de utilização que estejam de acordo com o novo processo de desenvolvimento estipulado.

Para resolver os problemas do cenário atual da empresa os seguintes tipos de ferramentas se fazem necessários:

- Sistema de controle de versão com suporte a ramificações e pedidos de mesclagem de código submetidos a revisões preliminares.
- Sistema de execução de *build pipelines* com suporte a ambientes de testes.
- Sistema de automação de implantações.
- Sistema de monitoramento de recursos.

Além dessas ferramentas citadas, para implantação de um ambiente de entrega contínua completo, tolerante a falhas e altamente escalável, um série de ferramentas adicionais se fazem necessárias, como centralizadores de logs, provisionadores de ambientes, orquestradores de máquinas virtuais, dentre outros. Devido a limitações de tempo, nem todas as ferramentas foram avaliadas e aplicadas no ambiente da empresa. Como o objetivo deste trabalho é resolver os principais problemas relacionados a produtividade e qualidade do processo de desenvolvimento, a maior quantidade de esforço foi concentrada na implantação das principais ferramentas e na adaptação da metodologia de trabalho do time.

Uma pesquisa foi realizada para selecionar as ferramentas disponíveis no mercado que atendem os requisitos levantados, pois como foi apontado pela XebiaLabs (2016), existem centenas de ferramentas criadas para estes propósitos, e testar todas tornaria este trabalho completamente inviável por limitações de tempo e pelo surgimento constante de novas ferramentas similares. Devido a estes fatores, para cada tipo de ferramenta foram levantadas entre 4 e 5 opções mais populares baseadas em análises de sites especialistas em cada assunto.

Diante do conjunto de ferramentas identificadas, as consideradas não cruciais para a boa conclusão deste trabalho não foram implantadas neste projeto. Por isso algumas das ferramentas descritas nos capítulos seguintes não apresentam um detalhamento tão aprofundado.

4.3.1 Sistema de controle de versão

Para escolher qual sistema de controle de versão usar levamos em conta que o time já possuía experiência prévia com *git*, o que facilitaria sua utilização, e graças a análises feitas por DevMedia (2016), InfoQ (2008), SitePoint (2014), percebemos que o *git* apresenta características que justificam ainda mais a sua escolha, mas ainda sim precisamos de uma ferramenta de controle de código fonte que tenha suporte a revisão e de preferência com uma interface gráfica para simplificar a gerência dos repositórios.

A Tabela 4 mostra os resultados da pesquisa para as ferramentas mais populares de controle de versão baseadas em git. A escolha dessas ferramentas foi feita baseando-se em recomendações de outros profissionais e recorrendo a análises feitas por sites como Slant (2016) e StackShare (2016a).

Baseando-se neste comparativo, chegamos a conclusão de que o GitHub e o BitBucket não representam a melhor escolha pelas limitações das versões gratuitas. Entre o Gogs e GitLab optamos pelo GitLab pela maior quantidade de recursos disponíveis.

4.3.2 Ferramentas de execução de *build pipelines*

As ferramentas de execução de *build pipelines* e de automação de rotinas de implantações podem ser encontradas na Tabela 5. A seleção dessas ferramentas também foi feita levando-se em conta as análises feitas pelo StackShare (2016b), porém também foram inclusos o BitBucket e o GitLab devido ao fato de serem ferramentas que, além de darem suporte a estes requisitos, também realizam o controle de código fonte. Uma vez que é vantajoso ter menos ferramentas para administrar, elas tem preferência sobre as demais.

Após a análise das funcionalidades disponíveis, o BitBucket e o Travis CI foram descartados pelas limitações das versões gratuitas. Ao experimentarmos o Jenkins e o

Tabela 4 – Comparativo entre ferramentas de controle de versão de código baseadas em git disponíveis no mercado.

	GitHub	**BitBucket**	**Gogs**	**GitLab**
Hospedagem remota	✓	✓	-	✓
Hospedagem local	-	-	✓	✓
Código aberto	-	-	✓	✓
Repositórios privados	Pago	Limite de 5 usuários por repositório	✓	✓
Gerência de organizações e times	✓	✓	✓	✓
Gerenciador de incidentes	✓	✓	✓	✓
Wiki	✓	✓	✓	✓
Pull requests	✓	✓	Apenas entre um *fork* e seu original	✓
Code Review	✓	✓	Não permite comentários em trechos de código	✓
Web Hooks	✓	✓	✓	✓
Build pipeline	-	✓	-	✓
Pesquisa por trecho de código	✓	-	-	-
Estatísticas de uso do repositório	✓	-	-	✓
Visualização gráfica das ramificações	✓	✓	-	✓
Snippets	✓	✓	-	✓

GitLab na prática, percebemos que a quantidade de *plugins* disponíveis para o Jenkins fazem com que ele se torne extremamente flexível em relação a diversidade de projetos e permitem análises profundas de quesitos como cobertura de testes, documentação e relatórios de compilação. Por outro lado o GitLab se mostra simples e prático para projetos menores e com menor diversidade.

Devido a estes fatores optamos pela utilização das duas ferramentas, resolvemos executar a *pipeline* no GitLab em projetos menores e menos complexos e deixamos o Jenkis para lidar com o processo de compilação de projetos maiores e mais complexos.

Tabela 5 – Comparativo entre ferramentas de execução de *build pipelines* disponíveis no mercado.

	Jenkins	**GitLab**	**BitBucket**	**Travis CI**
Hospedagem remota	-	✓	✓	✓
Hospedagem local	✓	✓	-	-
Código aberto	✓	✓	-	-
Construção de *branches*	✓	✓	✓	✓
Construção de *pull requestes*	✓	✓	✓	✓
Nível de complexidade	Mediano	Simples	Simples	Simples
Suporte a múltiplos ambientes	✓	✓	Apenas variáveis de ambiente	Apenas variáveis de ambiente
Integração com *Docker* ou contêineres	✓	✓	-	-
Suporte a plugins	✓	-	-	-
Suporte a rotinas de *deploy*	✓	✓	✓	✓
Forma de definição da *pipeline*	Arquivo de configuração dentro do repositório ou através de uma interface gráfica	Arquivo de configuração dentro do repositório	Arquivo de configuração dentro do repositório	Arquivo de configuração dentro do repositório

4.3.3 Ferramentas de monitoramento

As ferramentas de monitoramento de recursos, também identificadas com ajuda de análises de sites como DevOps.com (2015) e StackShare (2016c) e de indicações de profissionais. Elas podem ser encontradas na Tabela 6.

Devido a pequena diferença em relação a quantidade de funcionalidades entre as ferramentas de monitoramento, utilizamos como critério a experiência do time com as ferramentas em busca de reduzir o tempo de aprendizado e optamos pelo Zabbix.

4.3.4 Centralização de logs

As ferramentas de centralização de logs foram deixadas de lado durante a execução deste projeto devido a limitações de tempo e pelo fato do time já realizar um bom trabalho administrando os logs gerados pela aplicação graças aos *frameworks* de desenvolvimento utilizados. Porém uma breve análise de algumas das principais ferramentas de centralização de logs foi realizada para que um estudo de suas vantagens pudesse ser feito e essa decisão

Tabela 6 – Comparativo entre ferramentas de monitoramento de recursos disponíveis no mercado.

	Zabbix	**Cacti**	**Prometheus**	**Nagios**	**Icinga**
Hospedagem remota	-	-	-	-	-
Hospedagem local	✓	✓	✓	✓	✓
Código aberto	✓	✓	✓	✓	✓
Geração de gráficos	✓	✓	✓	✓	✓
Suporte a protocolo SNMP	✓	✓	✓	✓	✓
Suporte a protocolo JMX	✓	✓	✓	✓	✓
Suporte a protocolo IPMI	✓	✓	✓	✓	✓
Cliente para coleta customizada	✓	✓	✓	✓	✓
Suporte a alertas e notificações de situações críticas	✓	-	✓	✓	✓

pudesse ser tomada.

Uma das ferramentas mais descomplicada e robusta para centralização de logs se chama Kibana (KIBANA, 2016). O Kibana é uma aplicação *web* com interface simples e amigável que permite a visualização de arquivos de log e estatísticas. Basicamente ele funciona como uma ferramenta de visualização em conjunto do Elastic Search, que é o responsável pelo armazenamento dos arquivos de log. Para utilização do Kibana é necessário configurar sua aplicação para enviar e armazenar os logs no Elastic Search, e com algumas configurações, preparar o Kiabana para receber consultas e exibir painéis informativos sobre as informações registradas.

Uma boa alternativa ao Kibana é o GrayLogs (GRAYLOG, 2016), que também funciona em conjunto do Elastic Search, porém uma das principais diferenças entre eles é que o GrayLog utiliza também o Apache Kafka como sistema intermediário de controle de mensagens que permite o particionamento do fluxo de dados entre *clusters* de máquinas, o que é ideal para aplicações de analise de *big data*. Diversas arquiteturas distribuídas podem se beneficiar desse modelo expandindo seu nível de controle e monitoramento do estado da aplicação.

4.3.5 Conteinerização e provisionamento

De acordo com os requisitos atuais dos projetos desenvolvidos pela empresa, apenas uma *build pipeline* com *deploy* automático configurado no Jenkins e no GitLab já se mostraram suficiente para atender as necessidades atuais, porem não podemos deixar de comentar sobre o uso de contêineres e provisionamento de máquinas virtuais, uma vez que

essas tecnologias se mostram cada vez mais presentes em ambientes de desenvolvimento integrado, como pode ser observado em uma pesquisa publicada em 2016 pela RightScale (2016).

Os contêineres são resultado uma nova tecnologia de máquinas virtuais capazes de serem executadas sem a necessidade de carregamento de supervisores, o que as tornam mais leves que as máquinas virtuais convencionais. A utilização de contêineres para implantar um software em produção traz vantagens como controle total do ambiente de execução da aplicação, mais organização para configurações, versionamento de infra-estrutura, maior controle de dependências a nível de sistema operacional, maior aproveitamento do hardware, dentre outras (TURNBULL, 2016).

Com a utilização de contêineres, o time de operações de uma empresa pode trabalhar com provisionamento de máquinas virtuais onde não há a necessidade de configuração manual de servidores, basta criar os *scripts* de instanciação do contêiner, embutir a aplicação e inicia-lo no servidor de produção.

O provisionamento e implantação dos contêineres feito puramente com Docker muitas vezes não é tão simples quanto poderia ser, é aqui que entram as ferramentas de orquestração. As ferramentas de orquestração mais populares segundo a pesquisa da RightScale (2016) são o Puppet e o Chef, que muitas das vezes são utilizadas em conjunto. Além dessas duas, outras bem populares são o Kubernetes (KUBERNETES, 2016) e o Open Shift (OPENSHIFT, 2016).

Essas ferramentas permitem a construção de arquivos de descrição de infraestrutura, onde o desenvolvedor pode especificar quais imagens de contêineres serão instanciadas, quais serão os volumes de dados persistentes que ficarão salvos em disco, quais são as dependências entre contêineres para que o serviço funcione corretamente, quantas réplicas de cada serviço devem ser instanciadas, qual política de reinicialização deve ser aplicada a cada serviço em caso de falha, quais os requisitos de consumo para o levantamento de mais réplicas, dentre várias outras especificações.

4.4 Implantação do ambiente de desenvolvimento integrado

Parte II

Resultados

5 Análise do novo cenário

5.1 Coleta de dados

5.2 Comparativos com o cenário anterior

6 Conclusão

7 Sugestões para outras empresas

8 Trabalhos futuros

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

9 Conclusão

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.

Sed eleifend, eros sit amet faucibus elementum, urna sapien consectetuer mauris, quis egestas leo justo non risus. Morbi non felis ac libero vulputate fringilla. Mauris libero eros, lacinia non, sodales quis, dapibus porttitor, pede. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi dapibus mauris condimentum nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam sit amet erat. Nulla varius. Etiam tincidunt dui vitae turpis. Donec leo. Morbi vulputate convallis est. Integer aliquet. Pellentesque aliquet sodales urna.

Referências

ABES. Mercado brasileiro de software: panorama e tendências. 2016. Disponível em: <http://central.abessoftware.com.br/Content/UploadedFiles/Arquivos/Dados%202011/ABES-Publicacao-Mercado-2016.pdf>. Acesso em: 17 out. 2016. Citado na página 10.

AMAZON. Aws lambda, detalhes do produto. 2016. Disponível em: <https://aws.amazon.com/pt/lambda/details/>. Acesso em: 17 out. 2016. Citado na página 9.

DEVMEDIA. Controles de versão para projetos java. 2016. Disponível em: <http://www.devmedia.com.br/controles-de-versao-para-projetos-java/26056>. Acesso em: 10 out. 2016. Citado na página 22.

DEVOPS.COM. 9 open source devops tools we love. 2015. Disponível em: <https://devops.com/9-open-source-devops-tools-love/>. Acesso em: 10 out. 2016. Citado na página 24.

E-BUSINESS. Entendendo o manifesto Ágil. 2015. Disponível em: <http://www.ebusinessconsultoria.com.br/infonews/entendendo-o-manifesto-agil>. Acesso em: 10 out. 2016. Citado na página 9.

GRAYLOG. Graylog: Open source log management. 2016. Disponível em: <https://www.graylog.org/>. Acesso em: 10 out. 2016. Citado na página 25.

INFOQ. Sistemas de controle de versão distribuído: Um guia não tão rápido. 2008. Disponível em: <https://www.infoq.com/br/articles/dvcs-guide>. Acesso em: 10 out. 2016. Citado na página 22.

KIBANA. Kibana: Explore, visualize, discover data. 2016. Disponível em: <https://www.elastic.co/products/kibana>. Acesso em: 10 out. 2016. Citado na página 25.

KUBERNETES. Kubernetes: Production-grade container orchestration. 2016. Disponível em: <http://kubernetes.io/>. Acesso em: 10 out. 2016. Citado na página 26.

OPENSHIFT. Openshift: Paas by red hat, built on docker and kubernetes. 2016. Disponível em: <https://www.openshift.com/>. Acesso em: 10 out. 2016. Citado na página 26.

ORACLE. Oracle developer cloud service. 2016. Disponível em: <https://cloud.oracle.com/en_US/opc/developer-service/features>. Acesso em: 17 out. 2016. Citado na página 9.

RED HAT. Open shift features. 2016. Disponível em: <https://www.openshift.com/features/>. Acesso em: 17 out. 2016. Citado na página 9.

REISSWITZ, F. *Análise de Sistemas*: Qualidade de software. [S.l.: s.n.], 2013. v. 7. Citado na página 19.

RIGHTSCALE. State of the cloud report: Devops trends. 2016. Disponível em: <http://assets.rightscale.com/uploads/pdfs/rightscale-2016-state-of-the-cloud-report-devops-trends.pdf>. Acesso em: 10 out. 2016. Citado na página 26.

SILVEIRA, P. et al. *Introdução à Arquitetura e Design de Software*: Uma visão sobre a plataforma java. [S.l.]: Elsevier, 2011. Citado na página 19.

SITEPOINT. Version control software in 2014: What are your options? 2014. Disponível em: <https://www.sitepoint.com/version-control-software-2014-what-options/>. Acesso em: 10 out. 2016. Citado na página 22.

SLANT. What are the best alternatives to github for open source projects? 2016. Disponível em: <https://www.slant.co/topics/5335/~alternatives-to-github-for-open-source-projects>. Acesso em: 10 out. 2016. Citado na página 22.

STACK OVERFLOW. Stack overflow developer survey 2016 results. 2016. Disponível em: <https://stackoverflow.com/research/developer-survey-2016>. Acesso em: 10 out. 2016. Citado na página 9.

STACKSHARE. Code collaboration & version control. 2016. Disponível em: <http://stackshare.io/code-collaboration-version-control>. Acesso em: 10 out. 2016. Citado na página 22.

STACKSHARE. Continuous integration. 2016. Disponível em: <http://stackshare.io/continuous-integration>. Acesso em: 10 out. 2016. Citado na página 22.

STACKSHARE. Continuous integration. 2016. Disponível em: <http://stackshare.io/monitoring-tools>. Acesso em: 10 out. 2016. Citado na página 24.

TURNBULL, J. *THE DOCKER BOOK*: Containerization is the new virtualization. [S.l.: s.n.], 2016. Citado na página 26.

XEBIALABS. Periodic table of devops tools. 2016. Disponível em: <https://xebialabs.com/periodic-table-of-devops-tools/>. Acesso em: 10 out. 2016. Citado na página 22.

Apêndices

APÊNDICE A – Quisque libero justo

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

APÊNDICE B – Nullam elementum urna vel imperdiet sodales elit ipsum pharetra ligula ac pretium ante justo a nulla curabitur tristique arcu eu metus

Nunc velit. Nullam elit sapien, eleifend eu, commodo nec, semper sit amet, elit. Nulla lectus risus, condimentum ut, laoreet eget, viverra nec, odio. Proin lobortis. Curabitur dictum arcu vel wisi. Cras id nulla venenatis tortor congue ultrices. Pellentesque eget pede. Sed eleifend sagittis elit. Nam sed tellus sit amet lectus ullamcorper tristique. Mauris enim sem, tristique eu, accumsan at, scelerisque vulputate, neque. Quisque lacus. Donec et ipsum sit amet elit nonummy aliquet. Sed viverra nisl at sem. Nam diam. Mauris ut dolor. Curabitur ornare tortor cursus velit.

Morbi tincidunt posuere arcu. Cras venenatis est vitae dolor. Vivamus scelerisque semper mi. Donec ipsum arcu, consequat scelerisque, viverra id, dictum at, metus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut pede sem, tempus ut, porttitor bibendum, molestie eu, elit. Suspendisse potenti. Sed id lectus sit amet purus faucibus vehicula. Praesent sed sem non dui pharetra interdum. Nam viverra ultrices magna.

Aenean laoreet aliquam orci. Nunc interdum elementum urna. Quisque erat. Nullam tempor neque. Maecenas velit nibh, scelerisque a, consequat ut, viverra in, enim. Duis magna. Donec odio neque, tristique et, tincidunt eu, rhoncus ac, nunc. Mauris malesuada malesuada elit. Etiam lacus mauris, pretium vel, blandit in, ultricies id, libero. Phasellus bibendum erat ut diam. In congue imperdiet lectus.

Anexos

ANEXO A – Morbi ultrices rutrum lorem.

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

ANEXO B – Cras non urna sed feugiat cum sociis natoque penatibus et magnis dis parturient montes nascetur ridiculus mus

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

ANEXO C – Fusce facilisis lacinia dui

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.

Formulário de Identificação

Exemplo de Formulário de Identificação, compatível com o Anexo A (informativo) da ABNT NBR 10719:2015. Este formulário não é um anexo. Conforme definido na norma, ele é o último elemento pós-textual e opcional do relatório.

<table>
<tr><th colspan="4">Dados do Relatório Técnico e/ou científico</th></tr>
<tr><td colspan="3" rowspan="2">Título e subtítulo</td><td>Classificação de segurança</td></tr>
<tr><td>No.</td></tr>
<tr><td colspan="3">Tipo de relatório</td><td>Data</td></tr>
<tr><td colspan="3">Título do projeto/programa/plano</td><td>No.</td></tr>
<tr><td colspan="4">Autor(es)</td></tr>
<tr><td colspan="4">Instituição executora e endereço completo</td></tr>
<tr><td colspan="4">Instituição patrocinadora e endereço completo</td></tr>
<tr><td colspan="4">Resumo</td></tr>
<tr><td colspan="4">Palavras-chave/descritores</td></tr>
<tr><td>Edição</td><td>No. de páginas</td><td>No. do volume</td><td>Nº de classificação</td></tr>
<tr><td>ISSN</td><td></td><td>Tiragem</td><td>Preço</td></tr>
<tr><td colspan="4">Distribuidor</td></tr>
<tr><td colspan="4">Observações/notas</td></tr>
</table>