**FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS**

**FATEC PROFESSOR JESSEN VIDAL**

**THIAGO LUIS SILVA FORTUNATO**

**PLANEJAMENTO E IMPLANTAÇÃO DO SOFTWARE**

**AVALIADOR DE COMPETÊNCIAS MAPSKILLS**

**São José dos Campos**

**2016**

**THIAGO LUIS SILVA FORTUNATO**

**PLANEJAMENTO E IMPLANTAÇÃO DO SOFTWARE**

**AVALIADOR DE COMPETÊNCIAS MAPSKILLS.**

Trabalho de Graduação apresentado à

Faculdade de Tecnologia São José dos

Campos, como parte dos requisitos

necessários para a obtenção do título de

Tecnólogo em Banco de Dados.

**Orientador: Me. Eduardo Sakaue**

São José dos Campos

2015

**Sumário**

**- CAPITULO I**

**Problema (Tema):**

Planejamento e Implantação do Software Avaliador de Competências Mapskills.

**- Justificativa (Motivação): (Escritório de Carreiras)**

Com o intuito de aproximar os alunos do mercado de trabalho e auxiliá-los no desenvolvimento de competências, o Escritório de Carreiras proporciona esse suporte à Fatec de São José dos Campos.

O Escritório de Carreiras da Fatec foi moldado para ser um mecanismo direcionado a ajudar na preparação dos alunos para o mercado de trabalho. O escritório também atua na aproximação da FATEC com o setor Produtivo de São José dos Campos. Para que atingir esses objetivos ele foi dividido nas seguintes áreas: Setor de Carreira e Setor de Estágio.

Em conjunto com o Escritório de Carreiras foi desenvolvido um Software que é capaz de analisar os perfis dos alunos que estão ingressando na faculdade, especificamente aqueles que estão no 1º Semestre, mostrando aos professores e gestores da instituição, os mais diferentes perfis dos estudantes na faculdade. Dessa forma, a administração da Faculdade poderá criar soluções para combater o crescente número de defasagem de alunos que desistem dos cursos nos primeiros semestres.

Este software é na forma de um jogo baseado em uma história com alguns personagens, em determinados momentos será necessário que o aluno escolha uma opção referente a alguma atitude que o personagem deveria tomar. Com base nessas perguntas e respostas serão acrescentados pontos às competências do aluno e ao final do questionário é montado um gráfico com o percentual referente ao perfil aluno.

**- PROPOSTA E SOLUÇÃO (Metodologia):**

Esse software foi desenvolvido para que tenha acesso simultâneo de aproximadamente 50 usuários, porém, é de interesse do Centro Paula Souza disponibiliza-lo para que todas ETECs e FATECs acessem de forma simultânea, sendo alocado em um único servidor e as demais instituições farão requisições para esta aplicação.

Desta forma, existe a necessidade de garantir a entrega do software com maior qualidade e frequência, ou seja, a aplicação disponível e com versões num menor intervalo de tempo. Esta necessidade hoje em dia é de suma necessidade das empresas, pois agrega valor ao software em questão, pois não é aceitável que uma nova funcionalidade do software seja implementada somente quando houver um grande evento de implementação, ou numa atualização de versão. Hoje em dia o software só tem valor quando ele está em uso, disponível e com qualidade na entrega do que ele foi proposto a fazer, ou seja, com um ciclo de vida operacional menor para obter novos softwares e recursos nas mãos dos usuários e clientes.

Nesse sentido é necessário olhar para todos os setores envolvidos no desenvolvimento de um software, como os responsáveis por: Infraestrutura, Banco de Dados, Servidores, Testes e o grupo de desenvolvedores. Nem sempre são as mesmas pessoas que cuidam dessas tarefas e para que não haja conflito na equipe o *DevOps* veio solucionar todos os problemas elencados acima, pois ele alinha as necessidades individuais e coletivas no desenvolvimento e entrega do software.

Com a alta demanda de requisições, será necessário a implementação de um dos princípios da *Extreme Programming*, o *DevOps*, onde esta pratica tem por definição garantir Agilidade, Qualidade, Estabilidade com Escalabilidade e Elasticidade.

Aplicando seus princípios conseguirei atingir o objetivo de garantir a Qualidade e Disponibilidade de forma que Escalável. Para isso deverei integrar todos esses conceitos nos ambientes em que a aplicação estará hospedada. Utilizando das práticas *DevOps* e Entrega Contínua pregam resolverei o problema de implementação do Software, se tornando um software com valor agregado, arquitetado no melhor e mais atual ambiente de desenvolvimento existente no mercado.

**- CAPITULO II (Levantamento de Requisitos)**

Neste capítulo serão citadas metodologias utilizadas para o planejamento e implementação do MapSkills.

**DevOps**

Conceito inicado no ano de 2007, pelo desenvolvedor Andrew Shafer e o engenheiro de sistemas John Allspaw, que criaram um evento para discutir especificamente os desafios da integração das áreas de desenvolvimento e operações existentes nas empresas. O evento foi chamado de “*DevOps Day”,* esse nome se deu justamente pela junção das palavras desenvolvimento e operações, sendo muito bem aceito pela comunidade envolvida.

Modelo utilizado quando se trata sobre metodologia ágeis, afim de realizar entregas rápidas com qualidade. Tem finalidade de integrar os setores de desenvolvimento e operações, diminuindo a dificuldade que encontravam quando se lançava uma nova funcionalidade do software, pois os setores operacionais criam um ambiente propício para execução de determinadas ferramentas pré-definidas no escopo do projeto, alterando algo, pode acontecer de o software perder pontos no quesito qualidade.

Requisitos funcionais, naturalmente, são tangíveis, pois o usuário pode por si próprio aferir se está de acordo com as suas expectativas ou não. Por outro lado, os requisitos não funcionais não são levados em conta. Requisitos não funcionais estão ligados à performance, disponibilidade, segurança, escalabilidade e, nos dias atuais, a elasticidade é outro requisito não funcional que vem crescendo em nível de importância.

**Disponibilidade**

Garantir que o software estará sempre disponível para que os usuários consigam com qualidade na entrega do serviço em que ele foi proposto a fazer.

Como o software será utilizado por todos alunos do Centro Paula Souza, é fundamental que este requisito seja preenchido, pois de qualquer lugar e a qualquer momento o software deverá estar disponível.

**Segurança**

Atividade realizada pelos setores de desenvolvimento e operação, utilizando de ferramentas e metodologias para garantir a segurança do software. Sua implementação é de suma importância, pois nenhum cliente deseja que suas informações estejam disponíveis nas mãos de terceiros. Outra questão que tem que ser destacada são os possíveis ataques que aplicação pode sofrer, e com um alto nível de segurança alcançado, o software estará seguro quando a possíveis quedas.

**Escalabilidade**

Capacidade de poder aumentar a capacidade de demanda de requisições que o sistema suportará, de maneira fácil e com total controle sobre os recursos utilizados.

**Elasticidade**

A elasticidade é um grande recurso disponibilizado por alguns fornecedores de computação na nuvem (Cloud Computing) onde a infraestrutura se adapta à demanda, alocando mais recursos conforme a demanda por processamento aumenta e liberando estes recursos computacionais na medida em que a demanda diminui.

**Entrega Contínua**

A Entrega Contínua é um conjunto de práticas que tem como objetivo garantir que o novo código pode ser implantado no ambiente de produção a qualquer momento, já a implantação contínua leva um passo mais longe.

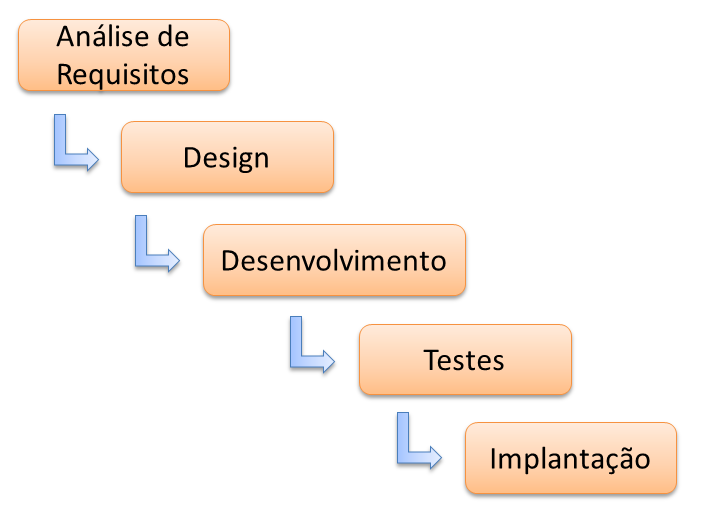
****

Figura 1.0 - Entrega Contínua

**CLOUD COMPUTING**

Cloud computing ou computação em nuvem, basicamente é definida como acesso ao software diretamente através de um navegador web, de qualquer lugar e independente da plataforma, como se estivesse instalado convencionalmente no computador do usuário.

Atualmente, a computação em nuvem é dividida da seguinte forma:

IaaS - Infrastructure as a Service ou Infra-estrutura como Serviço: quando se utiliza uma porcentagem de um servidor, geralmente com configuração que se adeque à sua necessidade.

PaaS - Plataform as a Service ou Plataforma como Serviço: utilizando-se apenas uma plataforma como um banco de dados, um web-service, etc. (p.ex.: Windows Azure).

DaaS - Development as a Service ou Desenvolvimento como Serviço: as ferramentas de desenvolvimento tomam forma no cloud computing como ferramentas compartilhadas.

SaaS - Software as a Service ou Software como Serviço: uso de um software em regime de utilização web (p.ex.: Google Docs, Microsoft SharePoint Online).

CaaS - Communication as a Service ou Comunicação como Serviço: uso de uma solução de Comunicação Unificada hospedada em Data Center do provedor ou fabricante (p.ex.: Microsoft Lync).

EaaS - Everything as a Service ou Tudo como Serviço: quando se utiliza tudo, infraestrurura, plataformas, software, suporte, enfim, o que envolve T.I.C. (Tecnologia da Informação e Comunicação).

**SERVIDOR**

Computador desenvolvido para fornecer serviço de software a outros computadores que estiverem conectados a ele ou a mesma rede.

Foram pensados para aguentar cargas mais pesadas de trabalho e mais aplicativos em seu conteúdo. Buscando diminuir o tempo de espera em inatividade, assim aumentando a produtividade.

**- CAPITULO III (Desenvolvimento)**

**Microsoft Azure**

Como o Centro Paula Souza tem um convênio com a Microsoft, foi definido com o requisito a hospedagem do aplicativo na plataforma integrada Microsoft Azure. Essa plataforma forneces acesso a diversas ferramentas para implantação, monitoramento, escalonamento da aplicação.

O Centro Paula Souza forneceu acesso a um host que está alocado na Plataforma Microsoft Azure.

No Microsoft Azure podemos monitorar, controlar e criar diversos recursos tudo na nuvem. Para isso é necessário o conhecimento do que é necessário para satisfazer a sua necessidade.

Para que tivesse acesso aos aplicativos que citarei em seguida, foi necessário realizar uma configuração na Plataforma Azure, liberando portas, ou seja, criando um canal, onde acontecerão conexões entre dados dos aplicativos e o Ubuntu Server VM.

Para configurar a liberação de portas no Microsoft Azure é necessário acessar o Grupo de Recursos, clicar em Network Security Group e dentro dessa opção realizar as configurações necessárias. Como no meu caso estou utilizando de aplicativos de serviço, foi necessário configurar individualmente as portas de entrada e saída para cada aplicativo instalado.

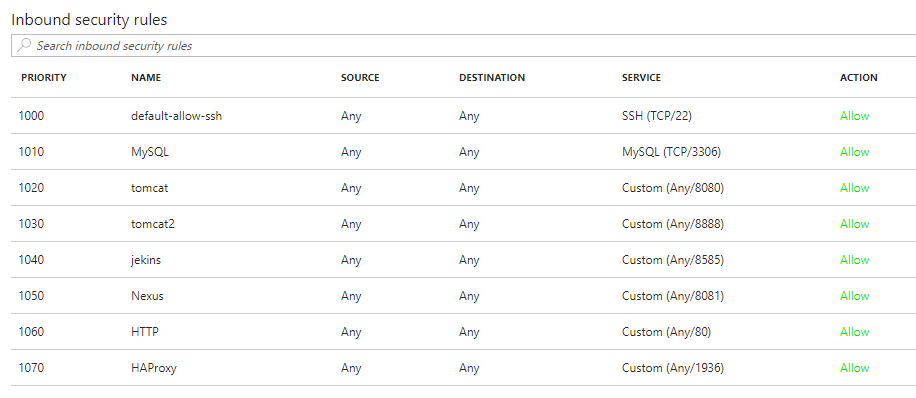


Figura 3. Regras de Segurança Azure Inbound

Configurando os Outbounds e Inbounds é possível acessar por meio externo ao Azure as aplicações instaladas no Docker dentro do Ubuntu Server.

Estas configurações são necessárias para que os recursos disponíveis na Virtual Machine sejam acessados fora do Host, ou seja, que aceite requisições por meio das portas configuradas e liberadas.

Foram adicionado a este Grupo de Recursos as seguintes configurações:

**Porta 8888**

Configurada para monitoramento dos containers.

**Porta 8080**

Configurada para acesso ao Haproxy Load Balancer, ele será responsável de controlar e balancear todas requisições feitas ao Back-End.

**Porta 8585**

Configurada para acesso ao Jenkins, aplicativo este que tem a finalidade de realizar a Entrega Contínua.

**Porta 8081**

Porta destinada ao Nexus Repository, que irá hospedar todos artefatos e arquivo pom.xml gerados no build pelo Jenkins.

**Porta 3306**

Porta configurada para acesso ao Banco de Dados Mysql.

**Virtual Machine**

Para desenvolvimento desta solução foi utilizado uma Virtual Machine, que nada mais é que um computador como qualquer outro, porém, virtualizado, ou seja, não existe um dispositivo físico. Todos seus recursos estão disponíveis na internet por meio da plataforma integrada, e sendo necessário apenas a criação de uma conta na plataforma para a criação da VM.

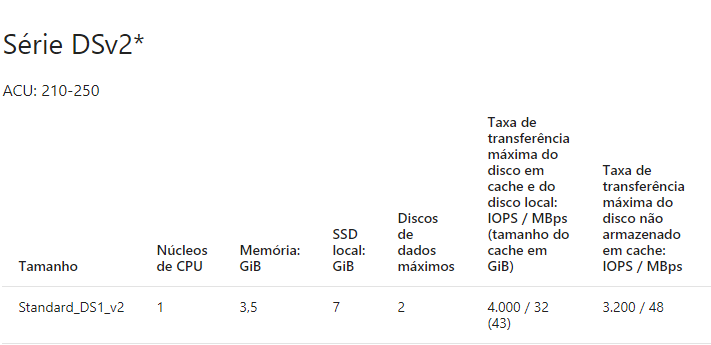
Com a Virtual Machine irei reduzir o custo do projeto, pois não é necessário a aquisição de um computador para realizar o serviço de servidor. Todos recursos estarão disponíveis na forma virtualizada, ou seja, é criado todo um ambiente computacional, sendo ele composto por: Memória Principal (RAM), Memória Secunda (HD) e Processadores.

**Ubuntu Server**

Na Plataforma Microsoft Azure, existem diversas opções de Sistemas Operacionais que podem ser instalados, porém, foi disponibilizado pelo Centro Paula Souza uma Virtual Machine com sistema operacional Ubuntu Server 16.04 LTS (Long Term Suppport), sendo baseado em Software Livre, ele garante total liberdade do usuário de customização do ambiente computacional e a diversidade de programas que são criados por diversos desenvolvedores do mundo inteiro.

A escolha do SO Ubuntu Server se deu por diversos requisitos alguns deles foram: Um sistema operacional voltado para o serviço de servidores, onde o mesmo não compartilha de recurso gráfico o que impacta positivamente no processamento de dados pelo servidor, outra vantagem do Ubuntu Server é a existência de uma grande comunidade de desenvolvedores que compartilham informações por meio de fóruns e todas essas informações são facilmente encontradas na internet.

Este Sistema Operacional é formado pelas seguintes configurações:



**Putty**

Para acesso ao Sistema Operacional foi utilizado um programa que emula terminais por meio de diversos tipos de conexão. Para que a conexão entre o Putty e o Sistema Operacional que estava hospedado dentro do Azure fosse feita de maneira segura, foi criado um usuário e senha no momento em que o Sistema Operacional é instalado.

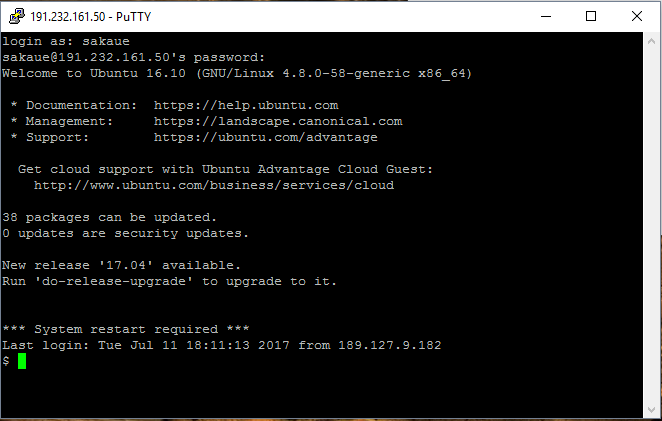


Figura 5. Login Azure Ubuntu Server VM através do Putty.

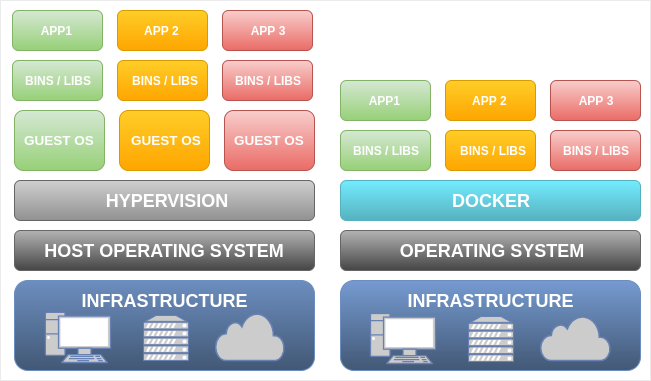
**Docker**

Um dos softwares que serão utilizados para garantir a disponibilidade do aplicativo enquanto hospedado no servidor. Docker é um sistema de virtualização, onde são criados containers isolados que compartilharão recursos com o SO.

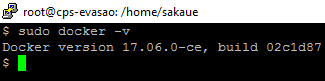
Com o Docker é possível o empacotamento de uma aplicação ou um ambiente inteiro dentro de um contêiner, compartilhando com outras aplicações o que irão utilizar num mesmo já configurado. Essa prática diminuirá drasticamente o tempo de deploy de uma aplicação, pois não será necessário configurar novamente o ambiente para que ele funcione corretamente.

Com o docker também reduzimos o custo do projeto, pois teremos somente um servidor com o Docker instalado, e todos recursos que serão necessários instalados dentro dele, assim, não é necessário ter computadores robustos ou mais de um computador para executar tarefas diferentes.

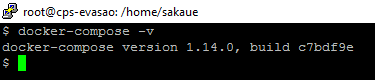
O modelo de isolamento utilizado no Docker é a virtualização a nível do sistema operacional, um método de virtualização onde o kernel do sistema operacional permite que múltiplos processos sejam executados isoladamente no mesmo host. Esses processos isolados em execução são denominados no Docker de container.



Para instalação do Docker no Ubuntu Server, segui o tutorial disponível no *https://docs.docker.com/engine/installation/linux/ubuntulinux.* Foi adicionado o repositório para download do aplicativo e configurada todas variáveis de ambiente para execução do aplicativo.



Foi instalado também o Docker Compose que faz parte do Ecosistema Docker, com ele foi possível criar todo ambiente, configurando cada container individualmente e inicia-los de uma só vez. Para instalação do Docker Compose foi seguido o tutorial disponibilizado no site <https://docs.docker.com/compose/install/>*.*



Uma imagem é um template ou molde que será baixado do repositório do Docker, o Docker hub. Neste repositório estão imagens oficiais e não oficiais. Como solução foi necessário baixar e configurar diferentes imagens, sendo estas: Mysql 5.6, Tomcat8, Haproxy, Nexus e cAdivisor. Todas imagens baixadas foram configuradas e geradas outras imagens a partir delas. E disponibilizadas no Docker Hub para que possam ser feitos o download de todo ambiente de forma rápida e fácil, desta forma, todo ambiente necessário para o Software Mapskills está disponível a qualquer um que deseja utiliza-lo.

Para realizar o download de uma imagem no Docker Hub, é necessário apenas ter instalado o Docker e a seguinte sintaxe:

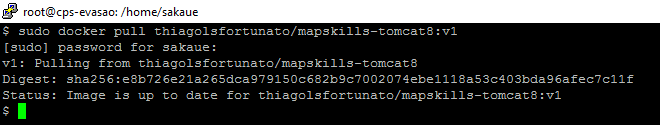


Figura 6. Exemplo de uma Imagem Tomcat no Docker Hub

**Containers**

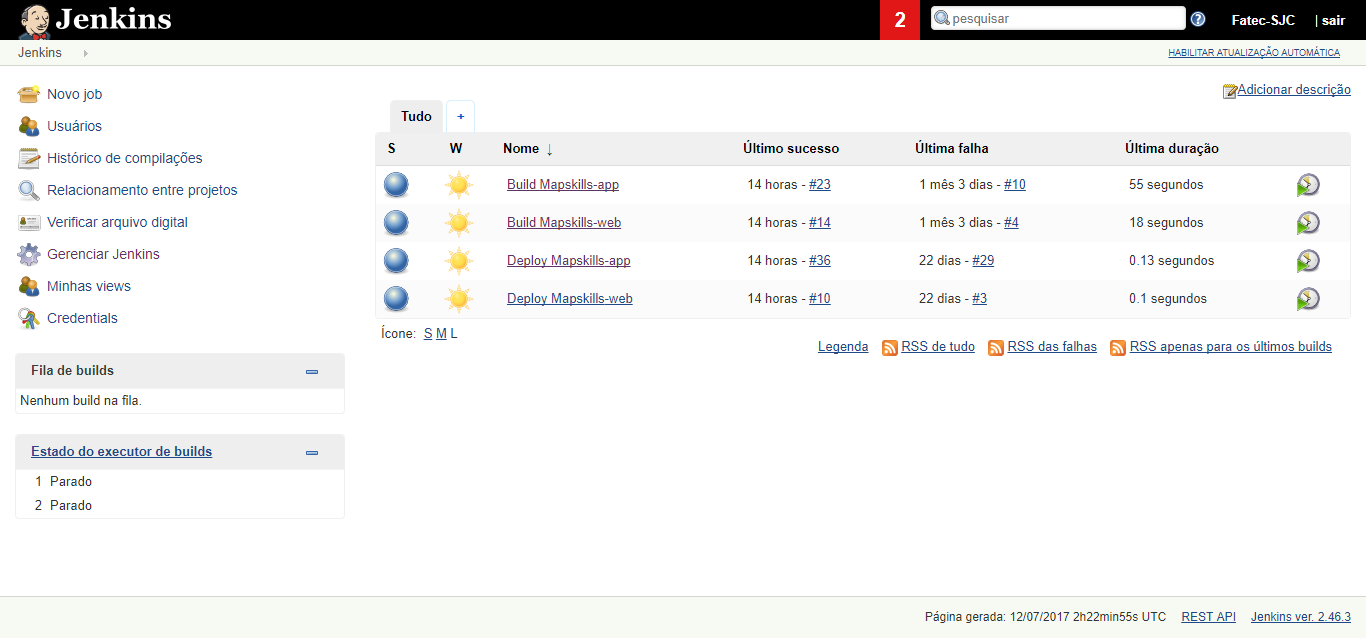
Os containers são instancias das imagens, neles são alocados todos recursos necessário para que sua aplicação funcione. É o container que faz toda magia do Docker acontecer. Cada usuário cria o contêiner do jeito que precisa, com outras imagens e recursos. Dessa forma não existe aquela famosa desculpa: “no meu computador funcionava”, pois, os containers após serem construídos podem ser disponibilizados no repositório do Docker.

Para inicialização dos containers que utilizei foram necessários comandos específicos, pois são recursos de configuração web. Sendo necessário configurar as portas já liberadas no Azure com as portas que escutarão os serviços dentro de dos containers.

Figura 7. Containers em execução no Docker

**Jenkins**

Aplicação instalada no host, o Jenkins é um dos principais aplicativos quando se fala de Integração Contínua hoje em dia, seu trabalho é fundamental no controle dos deploys realizados durante a implementação do software. Após configurado, o Jenkins tem o trabalho de realizar builds automáticos. Esse build funciona de forma instantânea, com testes sendo executados e falhas detectadas.



Após feita a liberação da porta, foi configurado o Jenkin pelo Web Browser, acessado através do ip-do-host:8585.

**Job Build Mapskills-app**

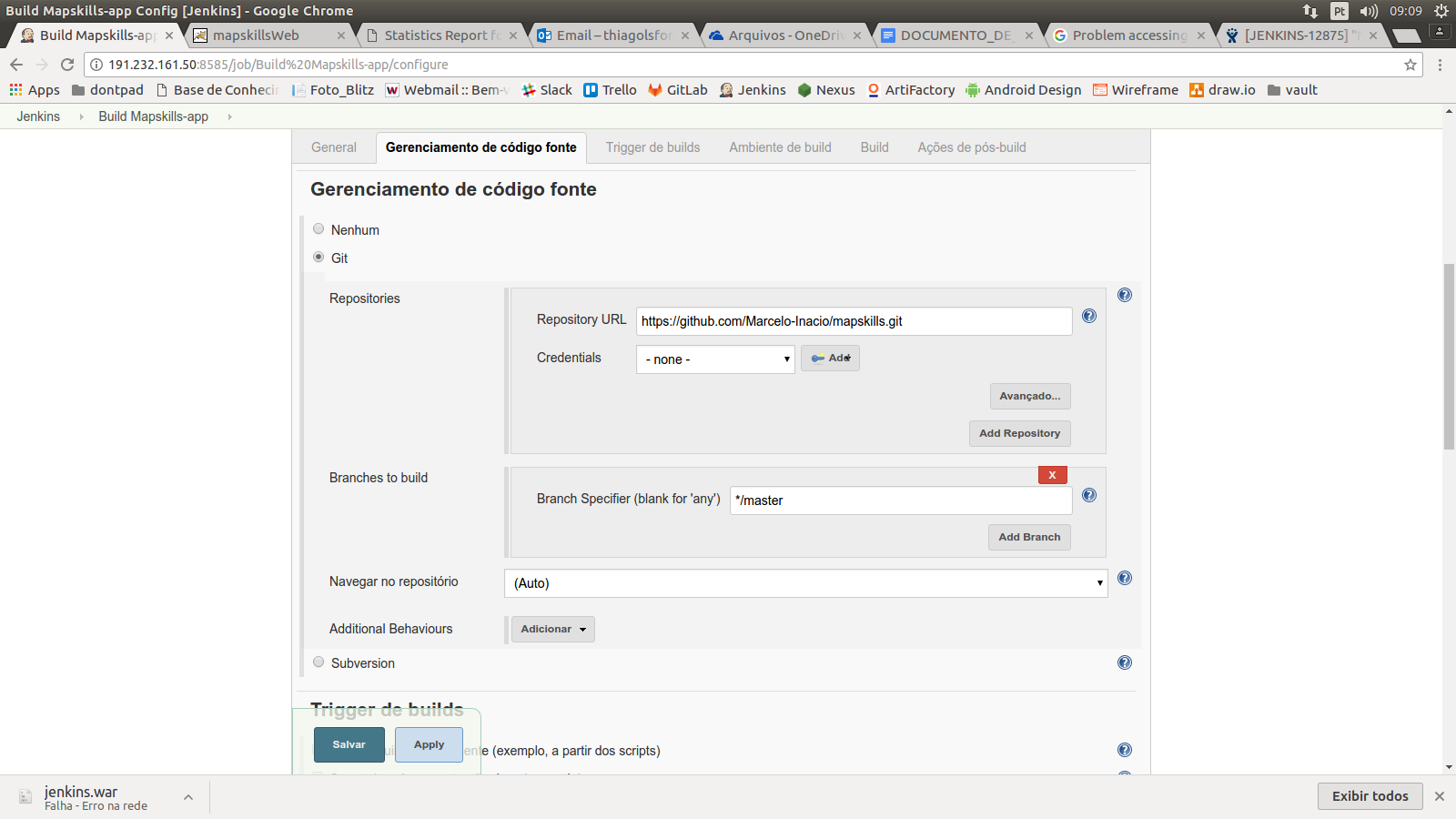
Este job é responsável por buildar o projeto mapskills através de comandos Maven.

**Gerenciamento de Código Fonte**

Para que o processo de build aconteça é necessário apontar o Repositório de Gerenciamento de código fonte, neste caso, ele está localizado no seguinte endereço:

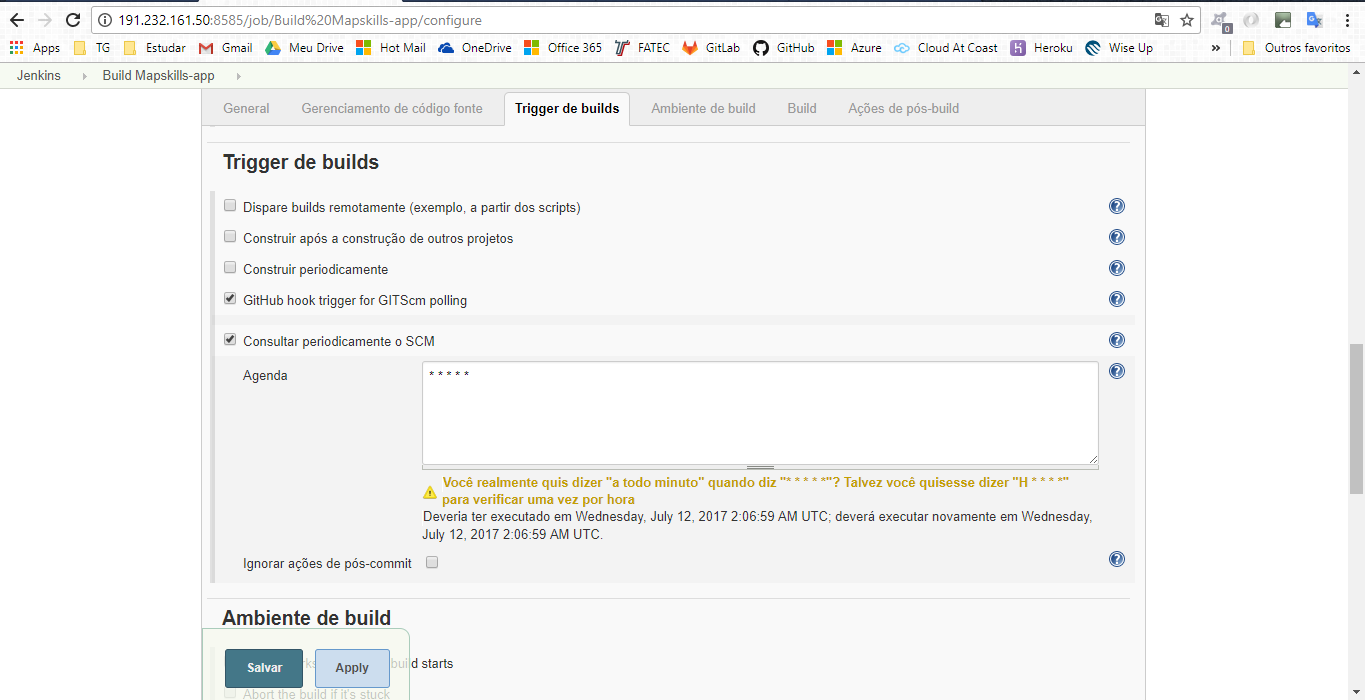
<https://github.com/Marcelo-Inacio/mapskills.git>

Sendo definida que a Branch Master como principal para este Job



**Trigger de Builds**

Para que o build aconteça foi configurado que o Jenkins consultará periodicamente o Repositório e verificará qualquer mudança, caso encontre algum commit na Branch Master o processo de Build acontecerá automaticamente.



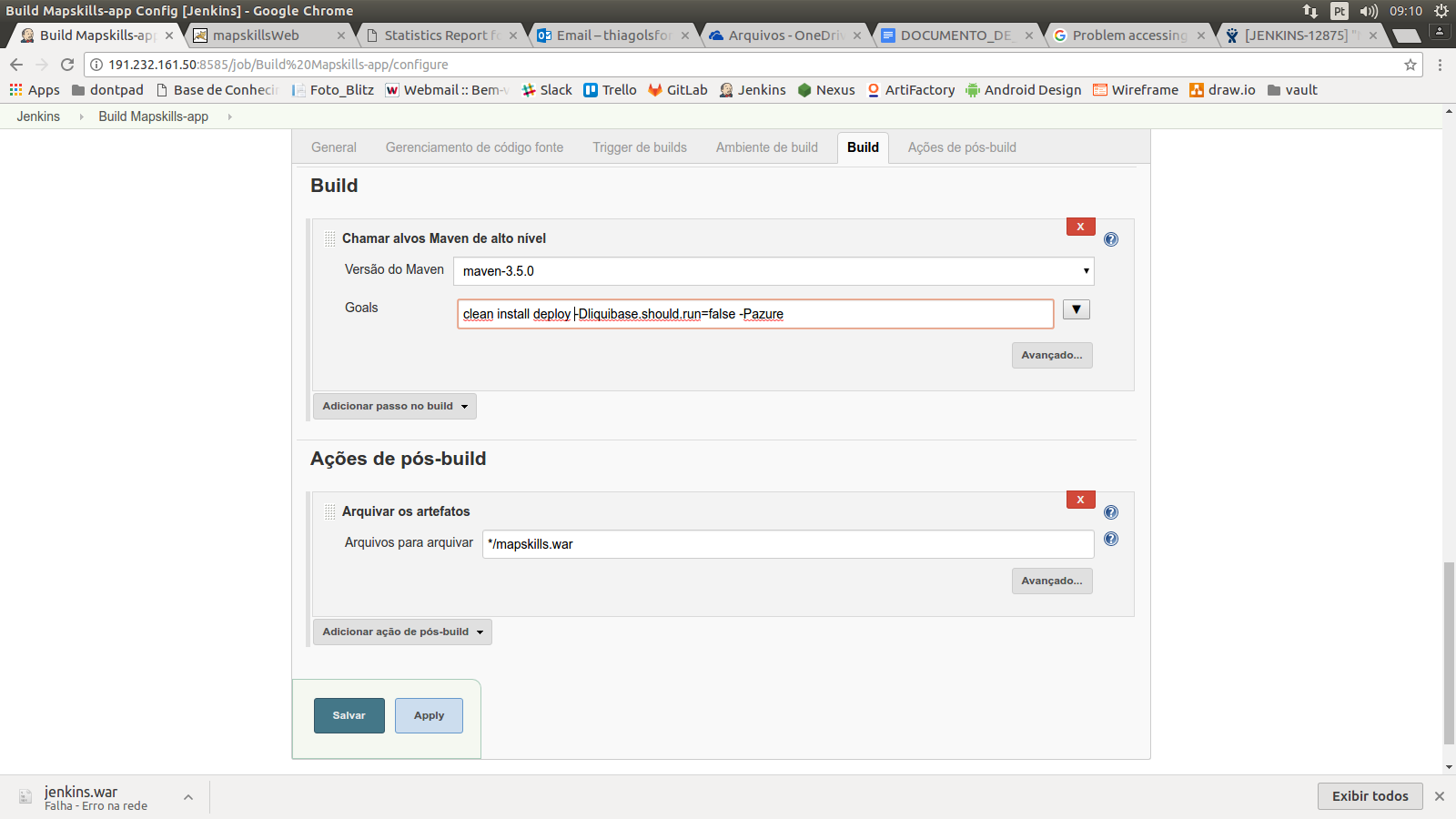
**Build**

Para que o Maven realize o processo de build do Software Mapskills, foi necessário configura-lo. Este processo é composto por identificar qual versão do Maven será responsável por rodar o comando maven e por definir qual o comando será executado.

O comando utilizado, irá limpar a pasta Target, instalar os pacotes nos respectivos repositórios, realizar o deploy do arquivo .war no Gerenciador de Repositório Nexus e não rodar o script de criação do Banco de Dados, pois desta forma é garantido que os dados nunca serão apagados. Todos esses comandos são rodados pelo usuário Azure, configurado na aplicação.

**Ações de pós-build**

Esta configuração é responsável por armazenar todos builds executados com sucesso pelo Jenkins, ele guardará no diretório */var/lib/jenkins/jobs/'Build Mapskills-app'/workspace/target/* o arquivo mapskills.war.

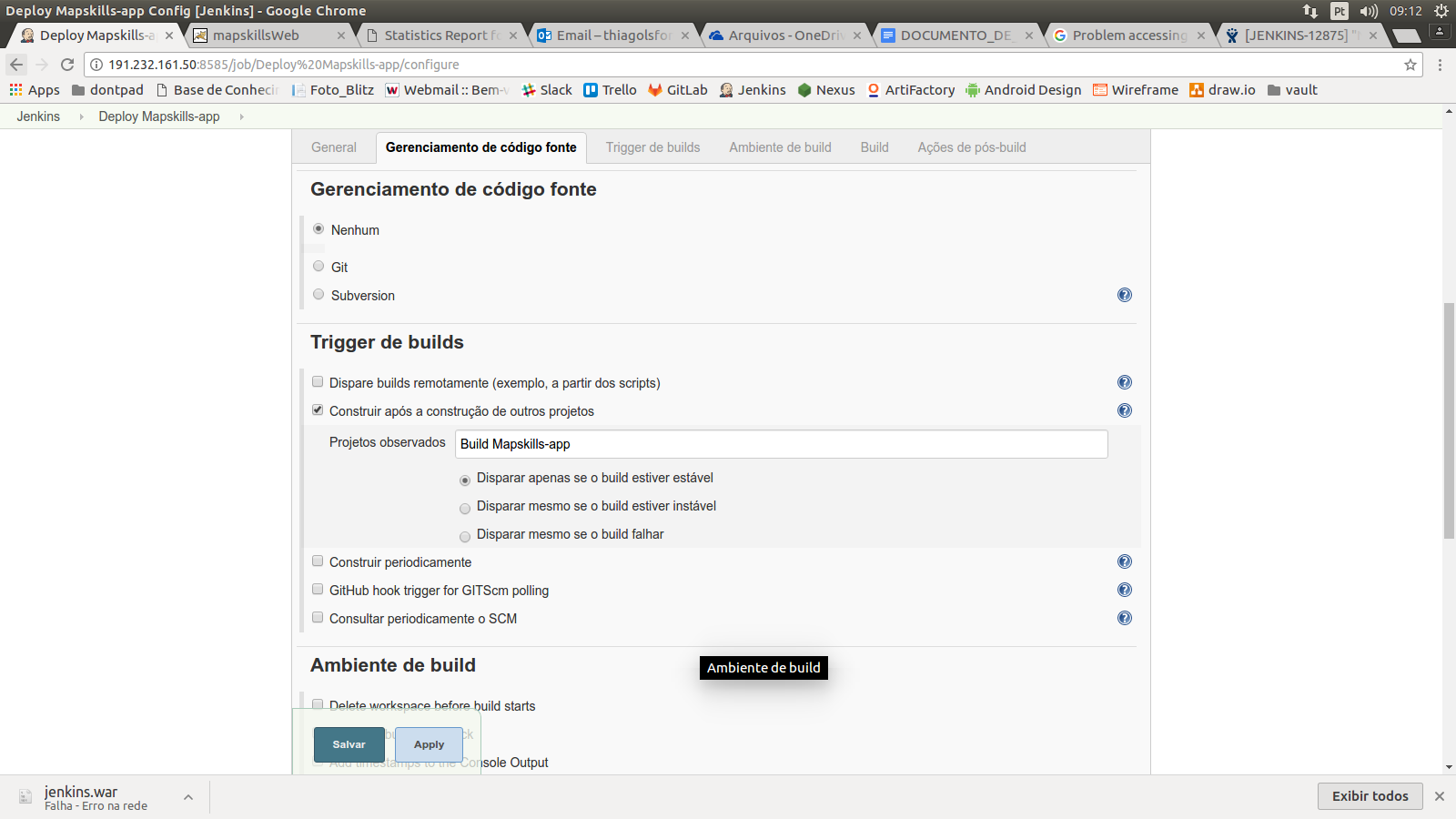


**Job Deploy Mapskills-app**

Este job tem a finalidade de realizar o deploy do arquivo mapskills.war de maneira automática.

**Trigger de Builds**

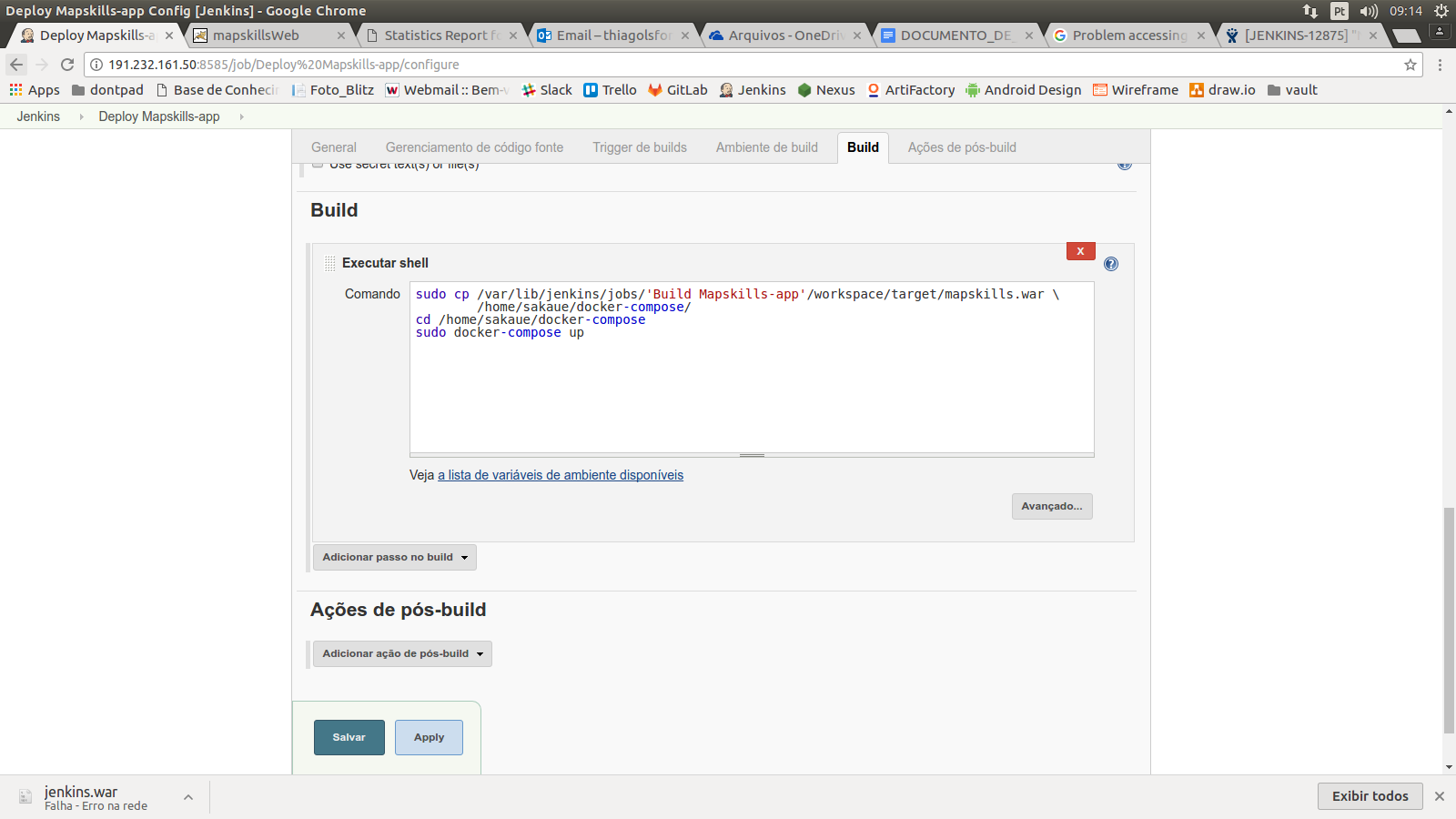
Este job é através da inicializado somente após a execução com sucesso do Job Build Mapskills-app, conforme imagem abaixo.



**Build**

O comando executado neste build tem a finalidade de copiar o arquivo mapskills.war para o diretório onde encontra-se o arquivo docker-compose.yml, pois é através dele que todo ambiente necessário para que o Software Mapskills entre em produção.

Sendo o finalmente executado o comando “docker-compose up”, colocando em produção todos containers definidos dentro do documento docler-compose.yml.



**Job Build Mapskills-web**

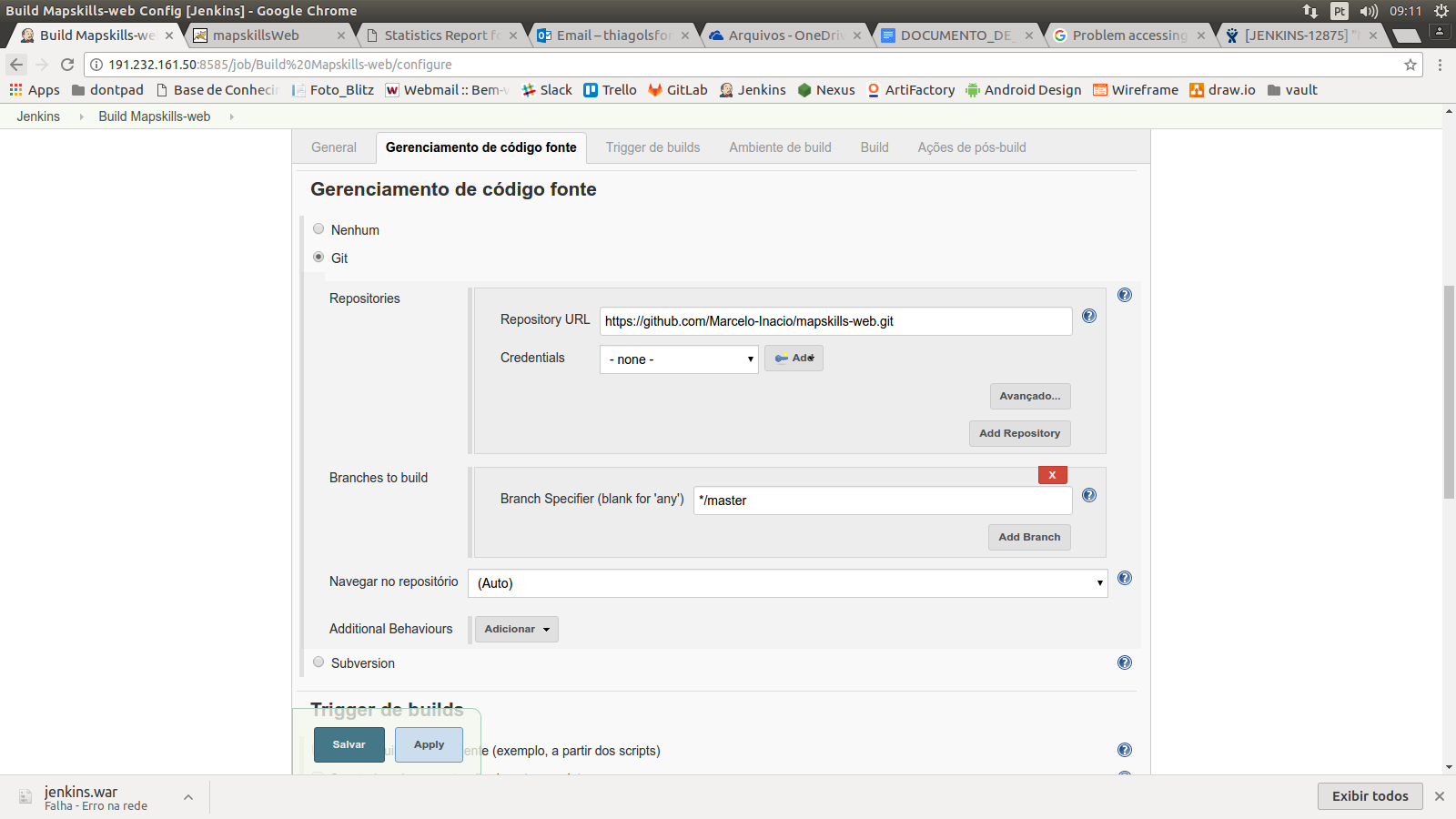
Este job é responsável por buildar o projeto mapskills através de comandos Maven.

**Gerenciamento de Código Fonte**

Para que o processo de build aconteça é necessário apontar o Repositório de Gerenciamento de código fonte, neste caso, ele está localizado no seguinte endereço:

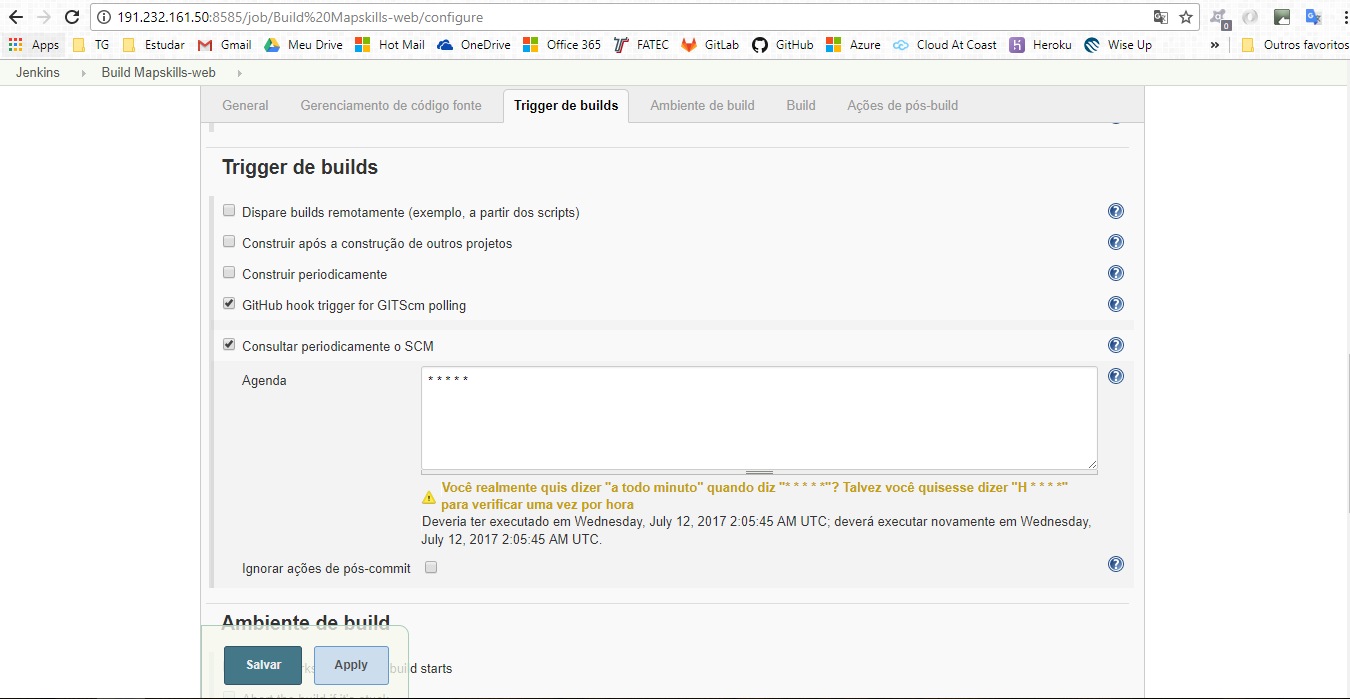
<https://github.com/Marcelo-Inacio/mapskills-web.git>

Sendo definida que a Branch Master como principal para este Job



**Trigger de Builds**

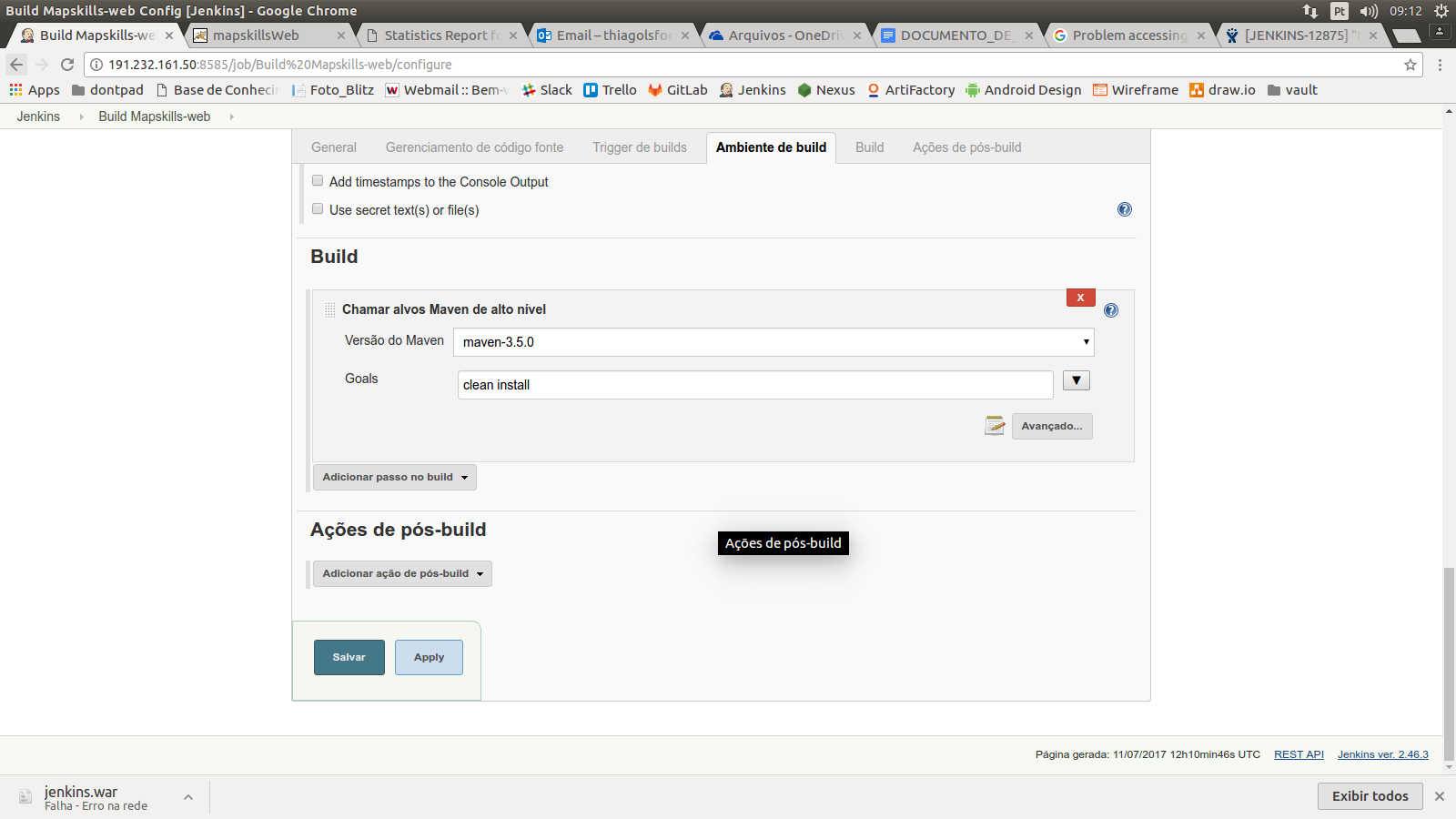
Para que o build aconteça foi configurado que o Jenkins consultará periodicamente o Repositório e verificará qualquer mudança, caso encontre algum commit na Branch Master o processo de Build acontecerá automaticamente.



**Build**

Para que o Maven realize o processo de build do Projeto Mapskills-web, foi necessário configura-lo. Este processo é composto por identificar qual versão do Maven será responsável por rodar o comando maven e por definir qual o comando será executado.

O comando utilizado, irá limpar a pasta Target e instalar os pacotes nos respectivos repositórios.

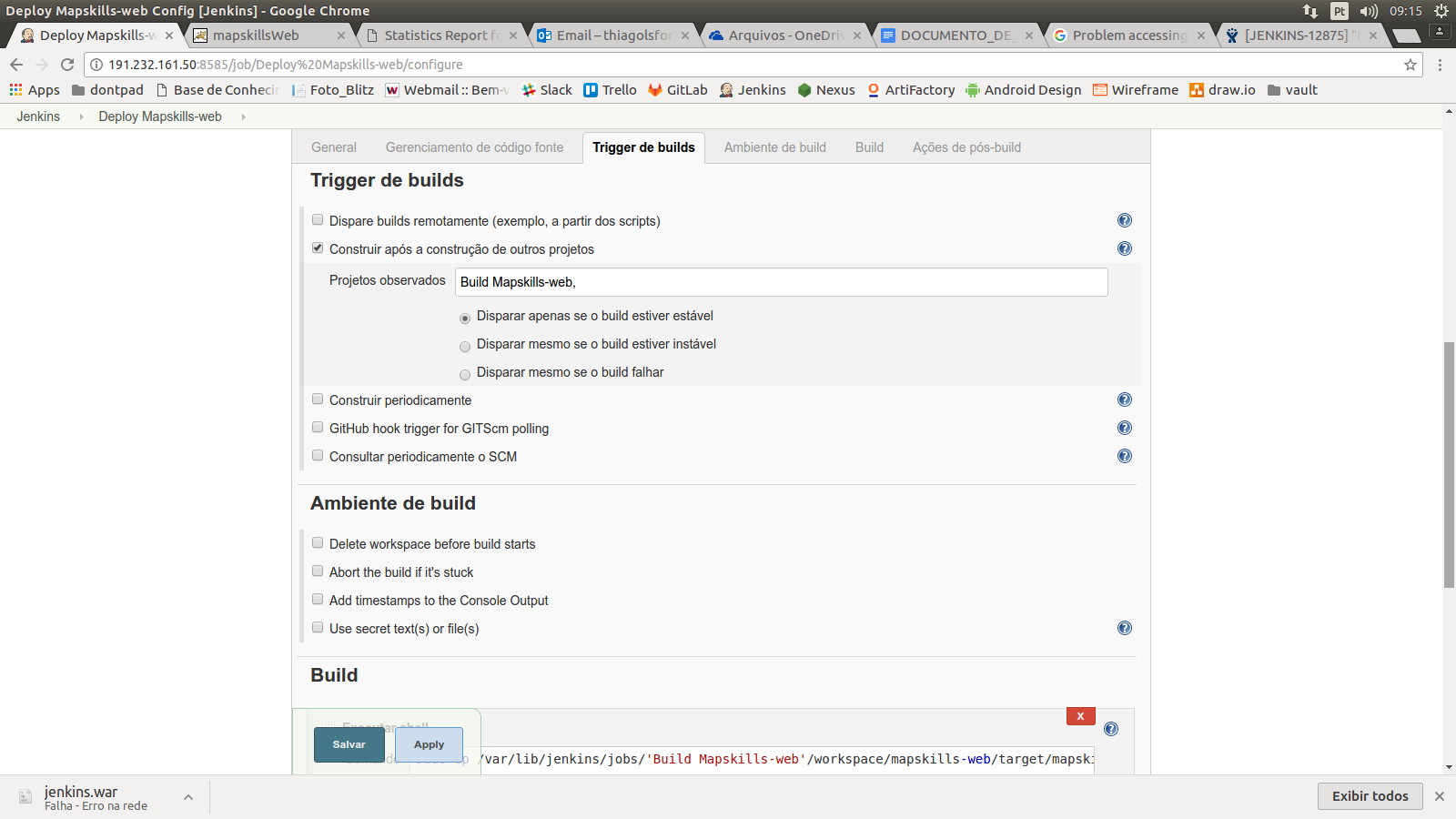


**Job Deploy Mapskills-web**

Este job tem a finalidade de realizar o deploy do arquivo mapskills.war de maneira automática.

**Trigger de Builds**

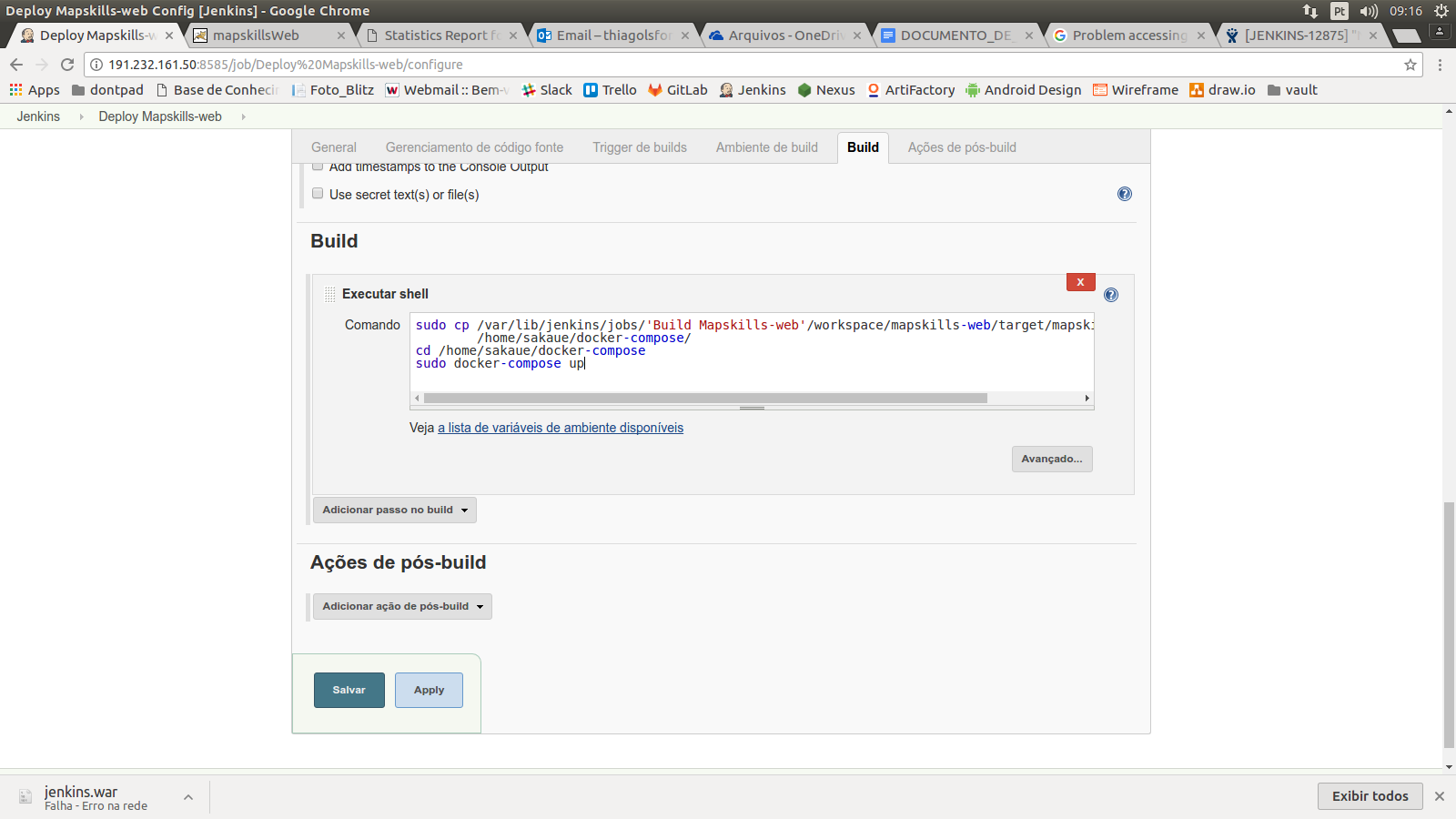
Este job é através da inicializado somente após a execução com sucesso do Job Build Mapskills-web, conforme imagem abaixo.



**Build**

O comando executado neste build tem a finalidade de copiar o arquivo mapskills-web.war para o diretório onde encontra-se o arquivo docker-compose.yml, pois é através dele que todo ambiente necessário para que o Software Mapskills entre em produção.

Sendo o finalmente executado o comando “docker-compose up”, colocando em produção todos containers definidos dentro do documento docler-compose.yml.



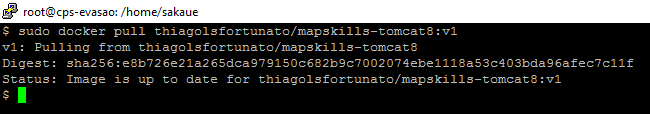
**Imagem Mapskills-Tomcat8**

Imagem responsável por gerar os containers que hospedarão os arquivos mapskills.war e mapskills-web.war. É acessada através da porta 8080, contendo todos arquivo de configurações padrão do Servidor Tomcat-8.5 com Java 8 já instalado. Esta imagem tem como base a Imagem Alpine, que tem a finalidade de reduzir o seu tamanho, rodando apenas 1 processo java, para execução da aplicação. O tamanho final desta imagem é de 334 Mb.

Para disponibilizar a Imagem no Repositório Docker foi é necessário utilizar do comando Docker push, conforme imagem abaixo



Para baixar a imagem do Repositório Docker é necessário rodar o seguinte comando:



**Imagem Mapskills-mysql**

Imagem base para o container que está contigo o Banco de Dados Relacional Mysql 5.6, para configura-lo primeiramente foi baixado a versão da Imagem Oficial do Mysql e realizado todas configurações como:

- Configuração de Senha para acesso ROOT;

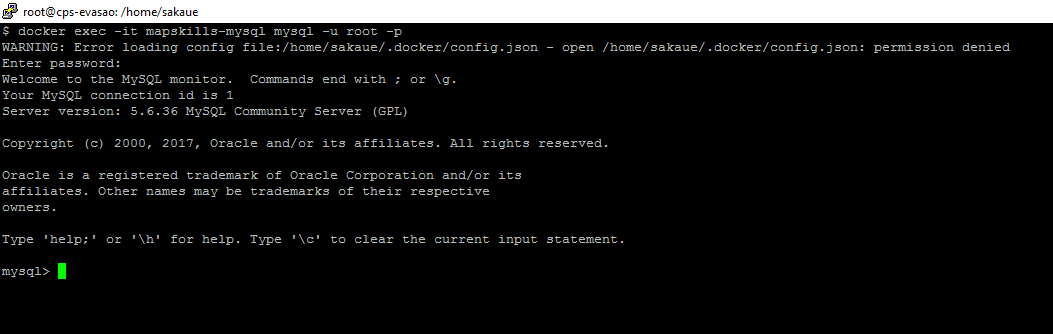
- Definição do nome do container;

- Definir porta para acesso remoto da Base de Dados;

O comando abaixo baixou a Imagem mysql:5.6, configurou e inicializou o container.



Para acessar o Banco de Dados foi utilizado o seguinte comando:

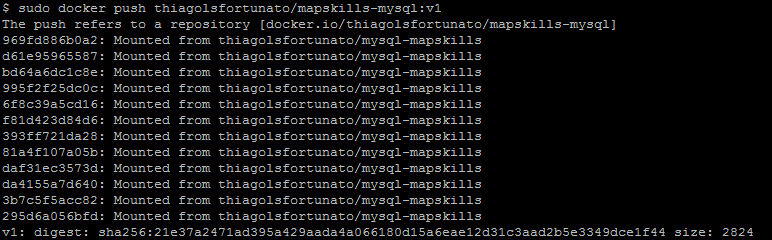


Nele, foi criado o usuário Mapskills, dado permissões de acesso remoto e criado a Base de Dados para que a Aplicação Mapskills.war persista todos dados.

Após todas realizar todas configurações no container e deixa-lo pronto para uso, foi utilizado o comando *docker commit*, e gerado uma imagem totalmente customizada do container.



E por último, realizado o deploy desta imagem no Docker Hub.



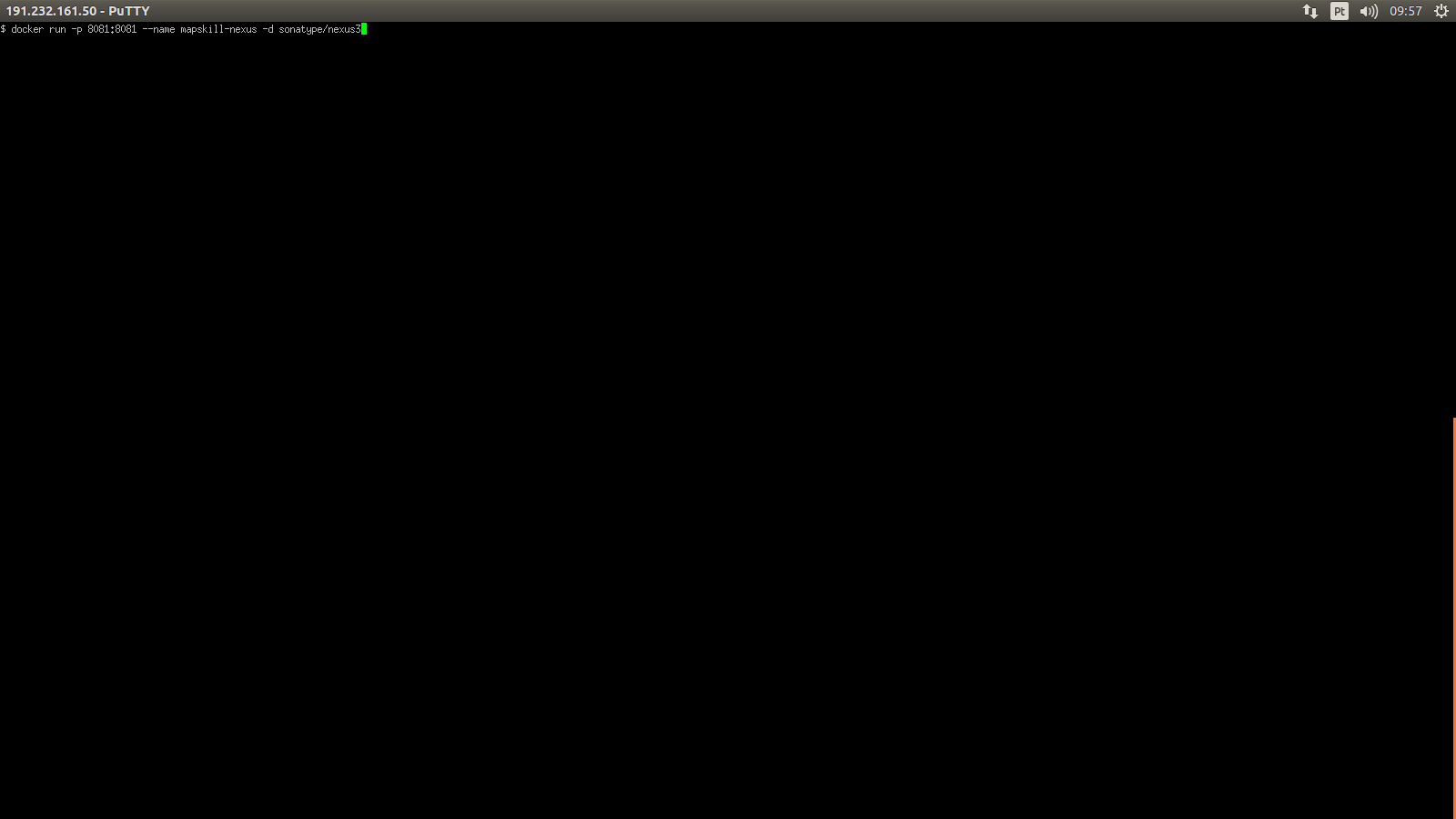
Para realizar o download dessa Imagem Mysql é necessário rodar o comando *docker pull*. Esta imagem já conterá todas configurações realizadas nos comandos acima.



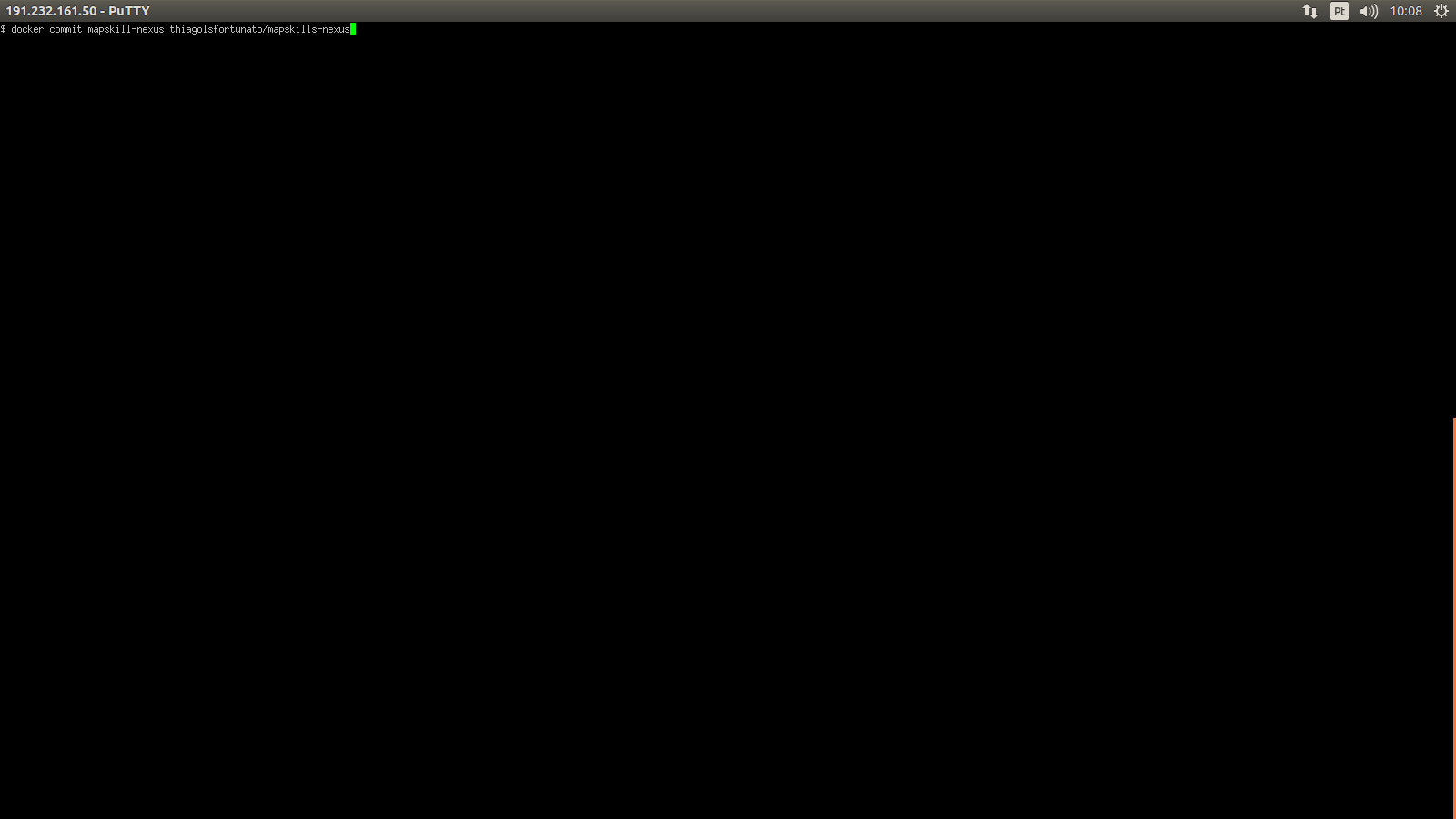
**Imagem Mapskills-Nexus**

Imagem responsável por armazenar todos artefatos que foram gerados após o build finalizado com sucesso feito pelo Jenkins. Para acesso a esta imagem é necessário liberar a porta 8081.

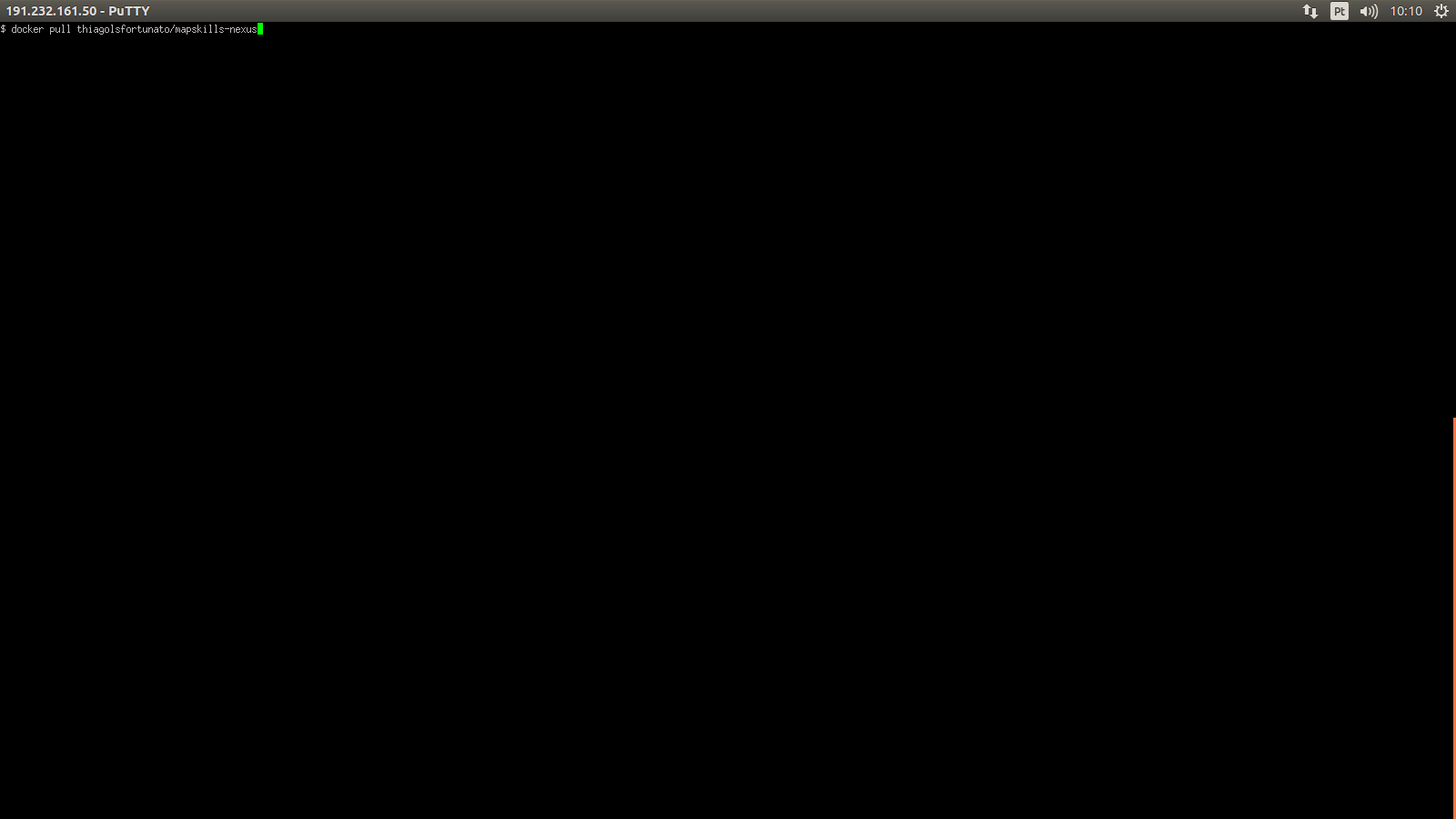
Primeiramente foi utilizado o comando *docker run* para fazer download da imagem ofical do Nexus Sonotype 3 disponibilizada no Docker Hub e criado um container de nome mapskills-nexus.



Apos configurado o container liberando a porta 8081 e criado um usuario Mapskills para acesso, foi gerado uma Imagem baseada neste container através do comando *docker commit*.



O comando descrito a baixo permite ao usuario baixar a Imagem Mapskills-Nexus do Repositorio Docker.



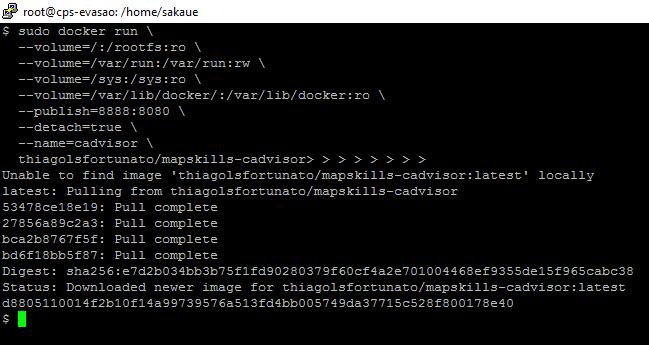
**Imagem Mapskills-cAdivisor**

Mapskills-cAdivisor e uma imagem que auxiliara no gerenciamento das imagens e containers utilizados no host. Com esta imagem e possivel monitorar todos processos e saber em tempo real todo funcionamento do host alocado.

Seguindo o conceito aplicado na Imagem Mapskills-Nexus foi feito o download da imagem no Repositorio Docker da Imagem Oficial e customizada para satisfazer as necessidades do Centro Paula Souza.

Todos volumes criados sao necessarios para que o cAdvisor configure seu ambiente.

A porta 8888 foi configurada para acesso a aplicacao.



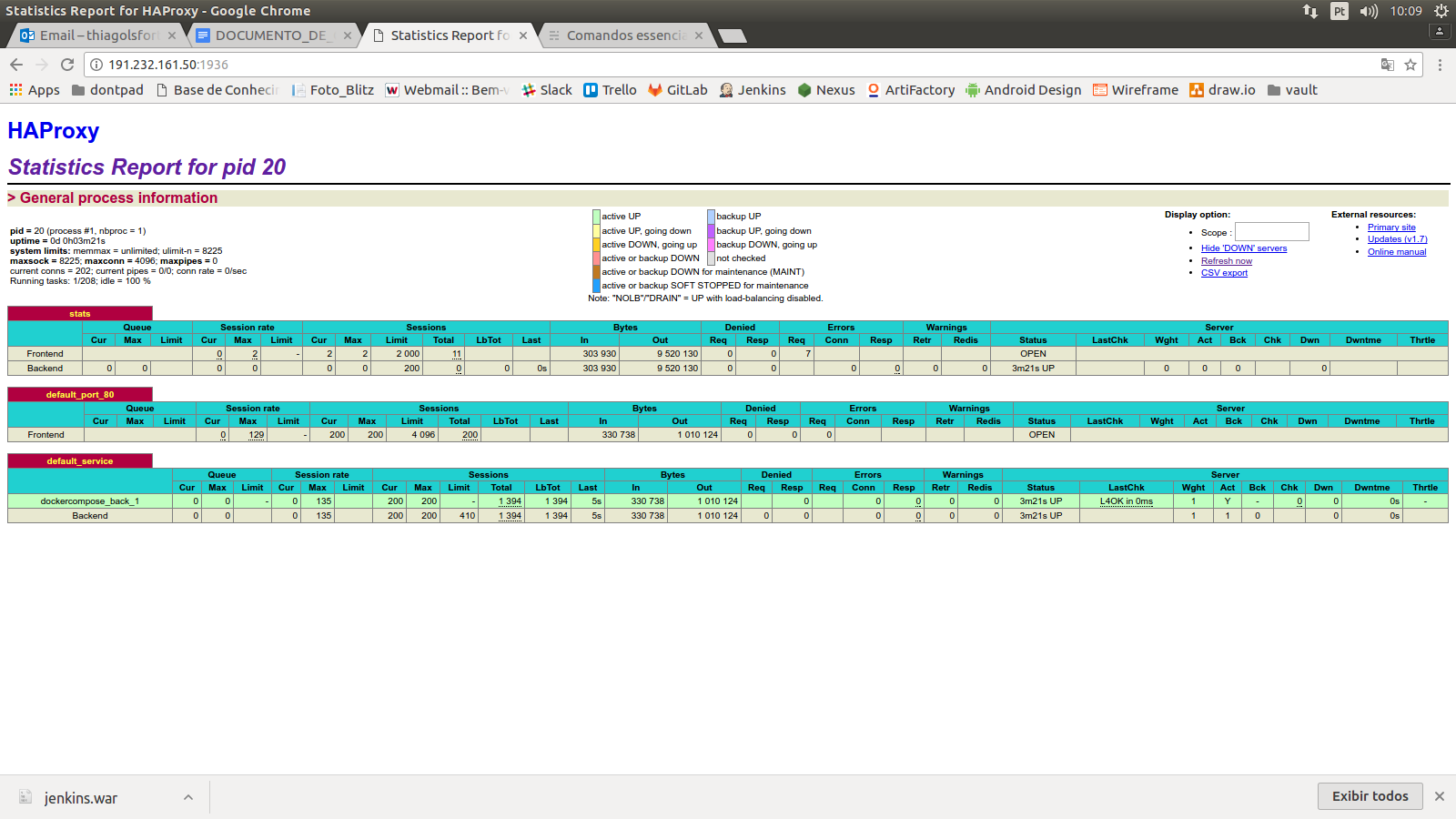
**Imagem Mapskills-Haproxy**

Imagem que tem a finalidade gerar o container que irá gerenciar o Load Balance dos Containers de Mapskills-app. Toda requisicao destinada ao Software Mapskills quem controlorá a carga sera o Haproxy por meio do um algoritmo de balanceamento de carga.

O Haproxy trabalhara como um Proxy Reverso, recebendo todas requisicoes atraves da porta 80 e redirecionando internamente a porta 8080 destinada ao Tomcat que contem o mapskills-war. O balanceamento de carga utilizado e de camada 4 (camada de transporte) da tabela OSI, desta forma o Haproxy encaminhara o trafego do usuario com base no alcance e na porta do IP, no caso definida como 80.

Aplicacao tambem fornece um Dashboard para visualizacao dos servers, numeros de requisicoes com sucesso e falha, alem de poder monitorar a quantidade que foi trafegada pela rede, conforme figura abaixo.

Alem da liberacao da porta 80 no host, a porta 1936 tambem foi configurada no Haproxy, pois e atraves dela que pode-ser visualizar o seu Dashboard e monitorar todo ambiente utilizando o usuario Mapskills.



**Round Robin**

Algoritmo utilizado para load balance que tratar os servers como iguais, independente do numero de conexoes solicitadas, sempre redirecionando a proxima requisicao ao server seguinte, desta forma, todos servers terao o mesmo numero de conexoes.

**Docker Compose**

Este arquivo de configuraçao tem a finalidade de automatizar o deploy de todo ambiente de produçao necessario para que o Mapskills funcione. O arquivo docker-compose.yml criara e iniciara todos serviços definidos.

O arquivo inicia os seguintes servicos: Container Nexus dando start ao Repositrio, Container Tomcat para o Front-End, composto pelo arquivo mapskills-web.war, Container Tomcat para o Back-End, composto pelo arquivo mapskill.war, Container Haproxy composto pelo software que ira realizar o Load Balancer, e Container cAdvisor que ira monitorar os recursos utilizado por todos containers.

No cabeçalho deste documento foi definido que utilizara a versao 2 do docker-compose e apos isso os servicos que serao configurados e inicializados.



**Serviços**

**Container Nexus**

Este Container sera inicializado a partir da Imagem thiagolsfortunato/mapskills-nexus, sendo restartado toda vez que ocorrer algum problema no docker e com acesso remoto da aplicacao atraves da porta 8081, sendo mapeada para a porta 8081 localmente no host.



**Container Tomcat Front End**

Este Container sera inicializado a partir da Imagem thiagolsfortunato/mapskills-tomcat8:v1, sendo restartado toda vez que ocorrer algum problema no docker, foi criado um volume para que seja copiado o arquivo mapskills-web.war para o diretorio */usr/local/tomcat/webapps/* do container, realizando assim o deploy da aplicaçao. A porta configurada para acesso remoto deste da aplicaçao web foi a 80 e sendo mapeada internamente para a porta 8080.



**Containers Tomcat Back End**

Este Container sera inicializado tambem a partir da Imagem thiagolsfortunato/mapskills-tomcat8:v1, configurado para ser restartado toda vez que o docker sair de operaçao. O deploy do arquivo mapskills.war acontece atravez do volume criado, onde o aquivo foi copiado do diretorio do localhost para o path */usr/local/tomcat/webapps/* do container. Nao foi mapeada nenhuma porta para este container, pois o Container Haproxy controlara todas requisicoes destinadas ao Container Tomcat Back End.

Este container sera e escalado atraves do comando docker-compose scale back=<qtd de containers>. Desta forma o Container Haproxy ira controlar todas requisicoes realizando o Load Balance.



**Container Haproxy**

Este container e proveniente da Imagem thiagolsfortunato/mapskills-haproxy:v1, sendo restartado toda vez que o docker sair de operacao. Configurada o usuario e senha para acesso a Pagina de Estatisticas do Haproxy, configurado um link entre o Container Haproxy, para que o aconteça o Load Balace no Container Tomcat Back End, criado um volume e compartilhado entre o container e o host o arquivo docker.sock. As portas definidas para acessar a aplicacao remotamente sao 8080:80 e 1936:1936, onde a porta 8080 sera responsavel por redirecionar todas requisicoes que tiverem como destino os Containers Tomcat Back End e mapear internamente para a porta 80, acontecendo de forma parecida com a porta 1936 sendo acessada remotamente e redirecionada para a porta 1936 internamente.



**Container Cadvisor**

**Capitulo 4**

**Referências**

Lyman,Jay.  https://blogs.the451group.com/opensource/2010/03/03/devops-mixing-dev-ops-agile-cloud-open-source-and-business/ 451 CAOS Theory.

  Nasrat, Paul. <https://www.infoq.com/presentations/agile-infrastructure. InfoQ. 05 March 2010>.

http://www.devmedia.com.br/gestao-de-projetos-e-integracao-continua-com-devops/34180

http://www.ibm.com/developerworks/br/rational/library/defining-deployment-deliverable-devops/

http://blog.justdigital.com.br/devops-qual-a-diferencas-entre-continuous-delivery-continuous-integration-e-continuous-deployment/

<http://www.infowester.com/cloudcomputing.php>

Catteddu, Daniele. "Cloud Computing: benefits, risks and recommendations for information security." Web Application Security. Springer Berlin Heidelberg, 2010. 17-17.

Humble, Jez, and David Farley. Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.

http://www.dell.com/learn/br/pt/brbsdt1/sb360/what-is-a-server