BLUE-GREEN DEPLOYMENT COM NGINX NO SISTEMA PETS4EVER.

ANDREW BORGES DE CAMPOS[1];VAGNER PINTO DA SILVA[2];
MARCIA ZECHLINSKI GUSMÃO[3].

[1]Instituto Federal Sul-rio-grandense Câmpus Pelotas – andrewborgescampos@gmail.com
[2]Instituto Federal Sul-rio-grandense Câmpus Pelotas – vagnersilva@ifsul.edu.br
[3]Instituto Federal Sul-rio-grandense Câmpus Pelotas – marciagusmao@ifsul.edu.br

1. INTRODUÇÃO

A hospedagem em nuvem é essencial para disponibilizarmos um sistema web pela internet. Atualmente, plataformas como o Heroku[1] simplificam esse processo, automatizando o *deploy* e a configuração. No entanto, ao optar por máquinas virtuais em vez de plataformas como serviço (PaaS) ou software como serviço (SaaS), é necessário configurar o sistema manualmente para evitar problemas, como o *downtime*. O *downtime* ocorre quando o sistema fica indisponível, especialmente durante atualizações, impactando negativamente a experiência do usuário ao interromper sua interação com o sistema.

Para evitar o problema de *downtime* uma técnica se destaca na implantação do *back-end*, chamada de *blue-green deployment*[2]. Essa técnica envolve manter duas máquinas virtuais idênticas, com versões diferentes do software, permitindo a troca sem interrupções. Aliada a esta técnica, está a utilização do proxy reverso *Nginx*[3] que direcionará os usuários do sistema para a versão nova do software.

A *blue-green deployment* foi implementada no sistema Pets4Ever, uma rede social dedicada exclusivamente a postagens sobre animais domésticos. O Pets4Ever é um sistema web desenvolvido com as seguintes tecnologias para *front-end e back-end*, respectivamente, React e Spring Boot. Para o banco de dados foi utilizado o PostgreSQL.

2. ATIVIDADES REALIZADAS

Através da plataforma *Oracle Cloud*, foram criadas duas máquinas virtuais com o sistema operacional Ubuntu. Ambas possuem configurações iguais, tanto de hardware quanto de software, executando serviços como o *PostgreSQL*, *Spring Boot, Firewall* e *Nginx*. Para que o acesso às máquinas seja realizado de forma segura, é necessário a utilização de certificados *ssl*[4], pois, sem este procedimento, apenas a conexão através de protocolo HTTP será possível, o que deve ser evitado pois não oferece segurança. A máquina com a versão azul, de IP 150.230.76.234, foi configurada com o domínio api.pets4ever.site, enquanto a versão verde, com IP 132.226.167.187, recebeu o domínio apigreen.pets4ever.site. Ambos endereços são subdomínios do domínio principal pets4ever.site.

Através dos subdomínios, as máquinas foram disponibilizadas aos usuários do sistema Pets4Ever mediante configuração de *proxy* reverso *Nginx*. O proxy

1 Informações sobre o Heroku no link: https://www.heroku.com/
2 Sobre blue-green deployment: https://docs.aws.amazon.com/whitepapers/latest/overview-deployment-options/bluegreen-deployments.html
3 Informações sobre Nginx no link: https://nginx.org/en/
4 Informações sobre certificado SSL no link: https://www.cloudflare.com/pt-br/learning/ssl/what-is-an-ssl-certificate/

atua como um portão de entrada do sistema, ou seja, a partir dele, as requisições serão direcionadas ao destino determinado. Quando uma nova versão do *back-end* está pronta, o comando *split clients* no Nginx é ajustado para redirecionar metade das requisições (50%) para a nova versão, hospedado na máquina virtual de domínio https://apigreen.pets4ever.site, enquanto a versão anterior recebe a metade restante (50%) do tráfego hospedado na máquina virtual de domínio https://api.pets4ever.site. Nesse sentido, a Figura 1 demonstra a configuração do proxy reverso *Nginx* para roteamento dos usuários para a nova versão do software em suas respectivas máquinas virtuais.

Figura 1: Configuração do Proxy Reverso Nginx.

```
split_clients $remote_addr $target {
    50% http://127.0.0.1:8443;
    50% https://apigreen.pets4ever.site:443;
 }

    server {
    listen 443 ssl;
    server_name api.pets4ever.site;

    ssl_certificate /etc/letsencrypt/live/api.pets4ever.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/api.pets4ever.site/privkey.pem;

    resolver 8.8.8.8 8.8.4.4 valid=300s;
    resolver_timeout 10s;

    location / {
        proxy_pass http://127.0.0.1:8443;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Fonte: do Autor (2024).

Caso ocorram erros na nova versão, que está hospedada em https://apigreen.pets4ever.site, o tráfego dos usuários poderá ser imediatamente redirecionado de volta à versão estável em https://api.pets4ever.site. Todo este processo não ocasiona *downtime*, ou seja, o tempo inativo do sistema durante a mudança, pois a mudança de roteamento através da configuração do *proxy* reverso *Nginx* é instantânea.

3. CONSIDERAÇÕES FINAIS

A técnica de *blue-green deployment* resultou em uma experiência de usuário significativamente melhorada, pois permitiu que as atualizações ocorressem sem que o usuário experimentasse qualquer interrupção no acesso ao sistema. Além de eliminar o tempo de inatividade, essa abordagem reduziu o risco de erros críticos ao permitir testes em produção de forma controlada. Em conjunto, o Nginx atuou como um facilitador crucial, garantindo a distribuição instantânea da versão atualizada, o que não apenas otimizou a manutenção contínua do sistema, mas também possibilitou o retorno imediato à versão anterior em caso de falhas. A integração do GitHub para o versionamento de código provou ser igualmente essencial, já que, por meio das GitHub Actions, foi possível automatizar o *deploy* para as máquinas virtuais de forma ágil e eficiente, com simples ajustes no arquivo de configuração deploy.yml. Esse processo está detalhado na Figura 2, que demonstra visualmente como o arquivo deploy.yml foi utilizado para direcionar os softwares às respectivas máquinas virtuais, reforçando a automação e simplificação do processo. Essa automação também reduziu a chance de erro humano durante a fase de implantação, promovendo maior consistência e confiança no processo de atualização.

Figura 2: Configuração Parcial do Arquivo Deploy.yml.

```
- name: Setup SSH
  env:
    # MUDAR PARA SSH_GREEN QUANDO SUBIR NOVA FEATURE
    # PADRÃO SSH_PRIVATE_KEY
    PRIVATE_KEY: ${{ secrets.SSH_PRIVATE_KEY }}
    PORT: ${{ secrets.SSH_PORT }}
    # MUDAR PARA GREEN_HOST
    # PADRÃO SSH_HOST
    HOST: ${{ secrets.SSH_HOST }}
    USERNAME: ${{ secrets.SSH_USERNAME }}
  run: |
    mkdir -p ~/.ssh
    echo "$PRIVATE_KEY" > ~/.ssh/id_rsa
    chmod 600 ~/.ssh/id_rsa
    ssh-keyscan -p $PORT $HOST >> ~/.ssh/known_hosts

- name: SCP to Oracle VM
  env:
    PORT: ${{ secrets.SSH_PORT }}
    # MUDAR PARA GREEN_HOST QUANDO SUBIR NOVA FEATURE
    # PADRÃO SSH_HOST
    HOST: ${{ secrets.SSH_HOST }}
    USERNAME: ${{ secrets.SSH_USERNAME }}
```

Fonte: do Autor (2024).

Concluída esta fase, os próximos passos visam o gerenciamento do fluxo de acesso ao sistema, permitindo acesso a portas específicas e o bloqueio de faixas de IP cujo acesso seja direcionado a portas não utilizados publicamente, visando aprimorar a segurança do sistema.

4. REFERÊNCIAS BIBLIOGRÁFICAS

AMAZON WEB SERVICES, INC. **Blue/Green Deployments – Overview of Deployment Options on AWS.** Amazon, setembro de 2024. Online. Acessado em setembro de 2024. Disponível em: https://docs.aws.amazon.com/whitepapers/latest/overview-deployment-options/bluegreen-deployments.html.

GITHUB. **Documentação do GitHub Actions**. GitHub, setembro de 2024. Online. Acessado em setembro de 2024. Disponível em: https://docs.github.com/pt/actions.

NGINX. **nginx**. nginx, setembro de 2024. Online. Acessado em setembro de 2024. Disponível em: https://nginx.org/en/.