Postmortens

Análises de causa de problemas com a infra do IMEsec para futuras referências.

- Postmortem 13/04/2020: nginx-proxy & novo site do IMEsec
- Postmortem 18/05/2020: reboot inesperado, crontab zoado, glusterfs não montado

Postmortem 13/04/2020: nginx-proxy & novo site do IMEsec

Métricas

- **Tempo fora do ar**: 31 minutos
- **Serviços afetados**: O site do imesec (https://imesec.ime.usp.br).
- **O quão foi afetado?**: Queda total

O que aconteceu

Estávamos felizes em colocar o novo site em produção. No momento, estávamos em uma infra um pouco bamba: todos os containers estavam rodando em um `docker-compose` na `torradeira`, nosso servidor com 4 CPU/8 GB RAM.

Fizemos uma simples mudança no nosso ambiente de deploy. Apenas duas linhas que causaram o problema todo. Estava assim antes:

```
nginx-website:
  image: richarvey/nginx-php-fpm:latest
  deploy:
    replicas: 2
    restart_policy:
      condition: any
      delay: "0"
  environment:
    GIT_EMAIL: imesec@ime.usp.br
    GIT_NAME: imesec
    GIT_REPO: https://github.com/IMEsec-USP/website.git
    VIRTUAL_HOST: imesec.ime.usp.br
```

```
      LETSENCRYPT_HOST: imesec.ime.usp.br
      WEBROOT: /var/www/html/website/
```

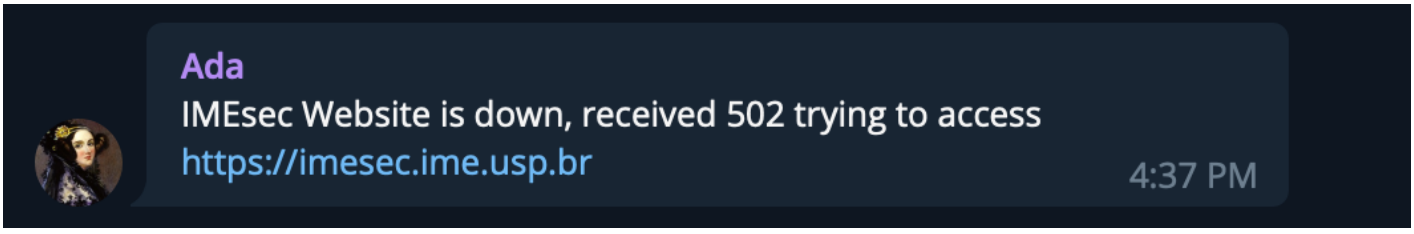
E depois da mudança:

```
  nginx-website:
    image: imesec/website # mudança aqui!
    deploy:
      replicas: 2
      restart_policy:
        condition: any
        delay: "0"
    environment:
      GIT_EMAIL: imesec@ime.usp.br
      GIT_NAME: imesec
      GIT_REPO: https://github.com/IMEsec-USP/website.git
      VIRTUAL_HOST: imesec.ime.usp.br
      LETSENCRYPT_HOST: imesec.ime.usp.br
      # Tiramos a variável webroot, já que não precisariamos mais dela.
```

Após a aplicação dessa mudança, a ada logo apitou nos seus logs:

```
[2020-04-13 15:33:14.018261] IMEsec Website returned 502
[2020-04-13 15:35:14.726066] IMEsec Website returned 502
[2020-04-13 15:37:18.408670] IMEsec Website returned 502
[2020-04-13 15:37:18.408754] IMEsec Website was considered HealthStatus.DOWN
[2020-04-13 15:37:18.408830] [TelegramHandler] broadcasting message in conversations
[131845033, -1001284501077, 211525815]
```

O site estava fora, e a ada apitou sobre:



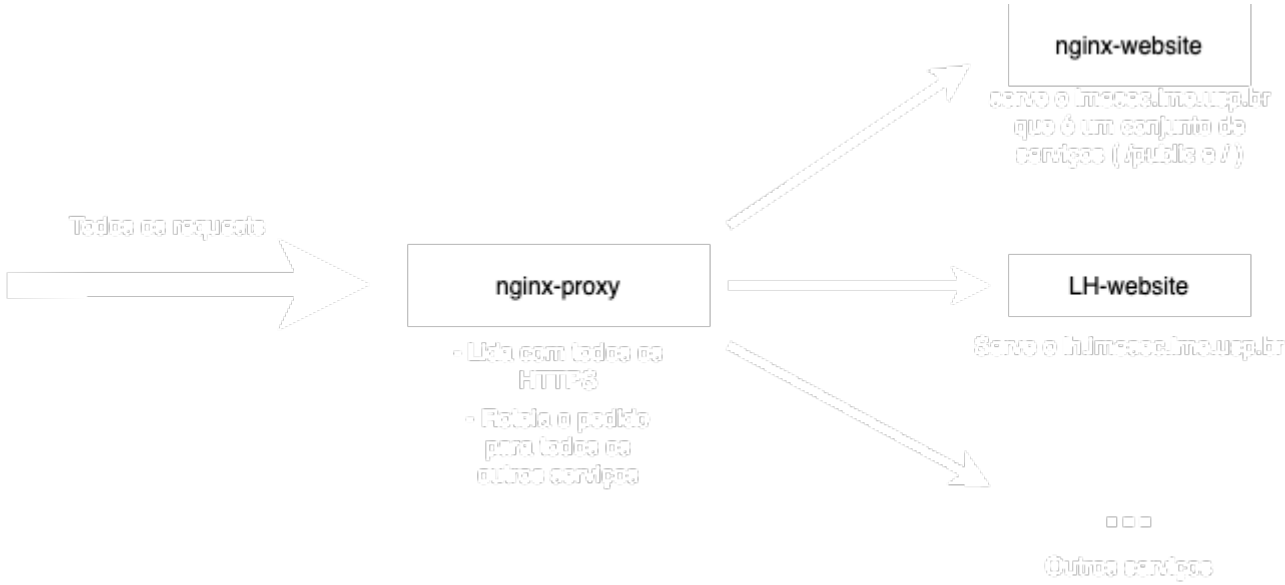
nota: o log está em GMT (UTC) e o horário da mensagem está em GMT+1 (British Summer Time).

O `razgrizone` e o `r0zbot` foram tentar resolver a situação.

processo de resolução

0. O que sabíamos

- Nossa infraestrutura desejada no momento era essa:



- Todos esses serviços estavam declarados na stack `imesec-full-stack.yml` declarada aqui.
- O Dockerfile da imagem nova era esse.
- Tudo descrito anteriormente.

1. desescalando o problema

Primeira ideia: **Rollback, depois pense**. Ou seja, primeiro tente voltar para a versão anterior e resolver a crise imediata. Só **depois**, com tudo estável, resolveríamos o problema.

Fomos o que tentamos. Mas a situação não normalizou. O website voltou com um 404:



Nessa hora, paramos pra **pensar**. Em uma avaliação rápida, achamos que resolver o problema e colocar o site novo no ar seria ainda mais rápido.

2. Analisando o ponto de falha

Já que conhecíamos a infraestrutura que estávamos mexendo, a primeira coisa foi tentar identificar o **ponto de falha** nela. Afinal, não sabíamos se o problema estava *dentro* da nova imagem `imesec/website` (logo do serviço `nginx-website` no gráfico) ou se era a comunicação entre a proxy de borda e o serviço (a seta que vai do `nginx-proxy` até o `nginx-website`).

De dentro do servidor, tentamos fazer o mesmo fluxo que um navegador faria:

```
curl -k -H 'Host: imesec.ime.usp.br' https://localhost
```

Isso nos devolveu o mesmo problema: `502 Bad Gateway`.

Tentamos então abrir uma porta do servidor e conectar nela diretamente, sem passar pelo proxy de borda. Fizemos isso com essa configuração:

```
nginx-website:
  image: imesec/website
  deploy:
    # ...
  environment:
    # ...
  ports:
```

```
      - "9001:5000"
```

Ou seja, a porta do servidor 9001 redirecionaria para a porta 5000 dentro do container. Fizemos isso pois a porta 5000 é a que está exposta na imagem.

Fizemos isso, e o site ficou visível para todos.



O problema estava então entre a conexão entre a proxy de borda e o serviço. Tentamos então acessar o site normalmente algumas vezes pra que isso ficasse óbvio nos logs. E ficou mesmo. Apareceu isso nos logs:

```
nginx.1    | imesec.ime.usp.br 172.19.0.1 - - [13/Apr/2020:16:01:39 +0000] "GET / HTTP/2.0"
502 559 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.118 Safari/537.36"
nginx.1    | 2020/04/13 16:01:44 [error] 58#58: *44 connect() failed (111: Connection
refused) while connecting to upstream, client: 172.19.0.1, server: imesec.ime.usp.br,
request: "HEAD / HTTP/2.0", upstream: "http://172.19.0.25:80/", host: "imesec.ime.usp.br"
```

Tomamos atenção especial ao fato de não conseguir se conectar em `http://172.19.0.25:80/`. A porta estava como porta 80 e não 5000!

tentamos fazer os dois comandos:

```
curl -I HEAD http://172.19.0.25:80/
curl -I HEAD http://172.19.0.25:5000/
```

O de cima não funcionou (ele deu `Connection refused`), mas o segundo funcionou. Era o que esperávamos, afinal, nosso serviço está configurado pra rodar na porta 5000.

Tinhamos então que configurar a proxy pra acessar o serviço na porta 5000. Isso foi feito com uma variável de ambiente:

```
nginx-website:
  # ...
  environment:
      VIRTUAL_PORT: 5000
```

Ao restartarmos o container do website, tudo voltou a funcionar normalmente.

Descobrindo a causa raíz

Depois de consertarmos da forma mais rápida possível, demos uma olhada mais a fundo no que tinha acontecido.

A imagem que usamos na proxy de borda utiliza o manifesto dos outros containers para automagicamente se configurar. Ela faz isso analisando as variáveis de ambiente e dockerfiles dos outros containers.

A imagem do nginx que usamos como base do nosso serviço também exporta a porta 80. Logo, nosso proxy não conseguiu identificar automaticamente qual delas teria que usar, e como fallback usou a primeira porta exportada: a porta 80, onde não tinha nada rodando.

Follow-ups

O follow up foi a resolução do problema: adicionar a porta 5000 no manifesto da nossa stack.

Nós também entendemos o porque do rollback não funcionar: a imagem antiga (`richarvey/nginx-php-fpm`) usa a variável de ambiente `GIT_REPO: https://github.com/IMEsec-USP/website.git` para clonar o repositório antes de rodar. Como a `master` do nosso repositório não tinha mais as pastas que ele esperava, ele não funcionou.

Uma lição aprendida é: quando possível, **não usar imagens que se constroem em runtime e não em tempo de build**. Uma imagem, quando funciona da primeira vez, é esperada a funcionar

em todas as vezes seguintes. Como mudamos o ambiente em volta do novo container da imagem `richarvey/nginx-php-fpm`, ele parou de funcionar. Se tivéssemos construído uma imagem estática com o nosso site antes (e é o que temos agora), isso não teria acontecido.

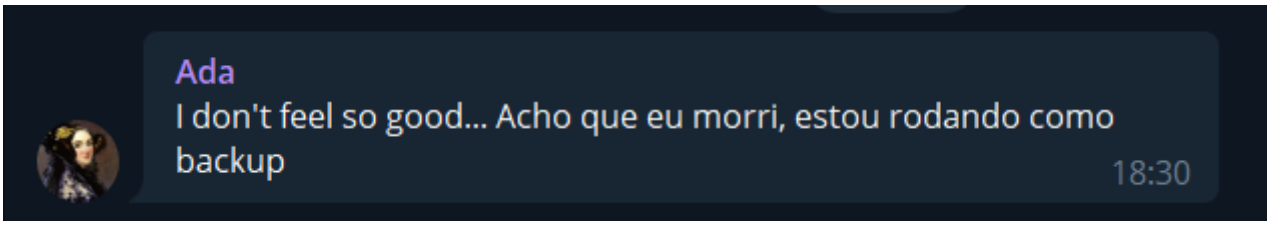
Postmortem 18/05/2020: reboot inesperado, crontab zoado, glusterfs não montado

Métricas

- **Tempo fora do ar**: 9 minutos
- **Serviços afetados**: T O D O S
- **O quão foi afetado?**: Queda total

O que aconteceu

Estávamos felizes e contentes em uma de nossas discussões diárias sobre qual ~~editor de texto~~ serviço de comunicação utilizar, quando de repente recebemos uma mensagem da nossa querida Ada, que é um bot de telegram que monitora os serviços e repositórios do IMEsec:



processo de resolução

0. O que sabíamos

A primeira coisa a ser feita foi verificar se foi só a Ada que caiu ou se houve algum problema no servidor. Uma rápida visita a imesec.ime.usp.br nos disse que nem o servidor de DNS sobreviveu.

Logo que logamos no servidor por SSH ficou claro o problema: ele reiniciou inesperadamente e o GlusterFS não conseguiu auto-montar, e portanto nenhum container conseguiu subir.

1. desescalando o problema

Como foi só uma falha em inicializar, a solução foi simples: iniciar manualmente o GlusterFS e subir nossos serviços.

Isso se resume a dois comandos:

```
mount -t glusterfs localhost:services /services && docker-compose up -d
```

E com isso tudo voltou ao normal :)

Analisando o ponto de falha

O GlusterFS não está listado em nossa /etc/fstab pois caso algum problema desse tipo ocorresse, o sistema inicializaria em modo de recuperação sem rodar nenhum serviço, então não conseguiríamos conectar por SSH.

A solução temporária pra isso foi colocar a linha `@reboot sleep 20 && mount -t glusterfs localhost:services /services` na crontab de root, para esperar alguns segundos para a stack de rede inicializar e em seguida montar o GlusterFS. Mas como já dizia Alan Turin: "Não há nada mais permanente do que uma solução temporária" [citation needed], então ela ficou lá. Eventualmente a stack de rede demorou um pouco mais de 20 segundos para subir e deu ruim.

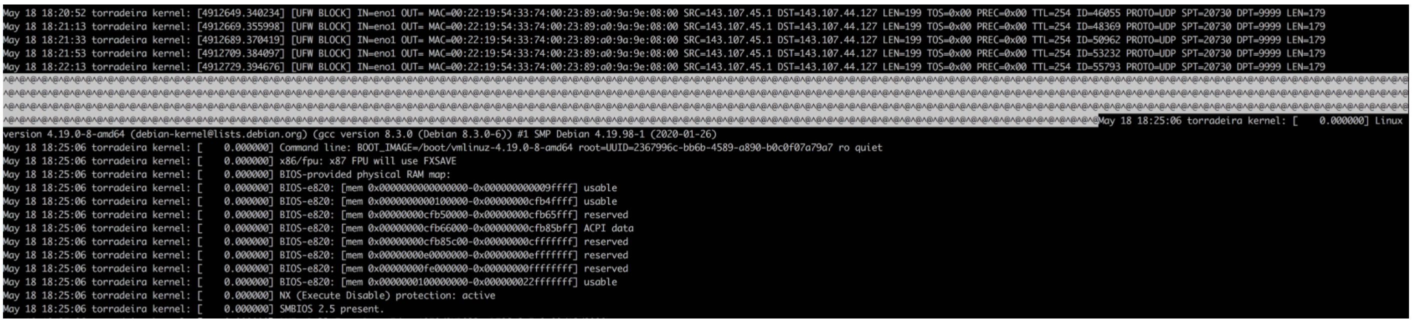
A solução certa seria provavelmente fazer uma unidade do systemd e colocar a stack de rede como dependência.

O que foi feito?

```
- @reboot sleep 20 && mount -t glusterfs localhost:services /services
+ @reboot sleep 60 && mount -t glusterfs localhost:services /services
```

Follow-ups

Ainda não sabemos ao certo o motivo do servidor ter reiniciado, mas suspeitamos do hardware ~~pré histórico~~, pois ao ver o arquivo `/var/log/syslog` percebemos que ele estava corrompido logo antes da inicialização (depois que reiniciou):



```
May 18 18:20:52 torradeira kernel: [4912649.340234] [UFW BLOCK] IN=eno1 OUT= MAC=00:22:19:54:33:74:00:23:89:a0:9a:9e:08:00 SRC=143.107.45.1 DST=143.107.44.127 LEN=199 TOS=0x00 PREC=0x00 TTL=254 ID=46055 PROTO=UDP SPT=20730 DPT=9999 LEN=179
May 18 18:21:13 torradeira kernel: [4912669.355998] [UFW BLOCK] IN=eno1 OUT= MAC=00:22:19:54:33:74:00:23:89:a0:9a:9e:08:00 SRC=143.107.45.1 DST=143.107.44.127 LEN=199 TOS=0x00 PREC=0x00 TTL=254 ID=48369 PROTO=UDP SPT=20730 DPT=9999 LEN=179
May 18 18:21:33 torradeira kernel: [4912689.370419] [UFW BLOCK] IN=eno1 OUT= MAC=00:22:19:54:33:74:00:23:89:a0:9a:9e:08:00 SRC=143.107.45.1 DST=143.107.44.127 LEN=199 TOS=0x00 PREC=0x00 TTL=254 ID=50962 PROTO=UDP SPT=20730 DPT=9999 LEN=179
May 18 18:21:53 torradeira kernel: [4912709.384097] [UFW BLOCK] IN=eno1 OUT= MAC=00:22:19:54:33:74:00:23:89:a0:9a:9e:08:00 SRC=143.107.45.1 DST=143.107.44.127 LEN=199 TOS=0x00 PREC=0x00 TTL=254 ID=53232 PROTO=UDP SPT=20730 DPT=9999 LEN=179
May 18 18:22:13 torradeira kernel: [4912729.394676] [UFW BLOCK] IN=eno1 OUT= MAC=00:22:19:54:33:74:00:23:89:a0:9a:9e:08:00 SRC=143.107.45.1 DST=143.107.44.127 LEN=199 TOS=0x00 PREC=0x00 TTL=254 ID=55793 PROTO=UDP SPT=20730 DPT=9999 LEN=179
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@May 18 18:25:06 torradeira kernel: [    0.000000] Linux version 4.19.0-8-amd64 (debian-kernel@lists.debian.org) (gcc version 8.3.0 (Debian 8.3.0-6)) #1 SMP Debian 4.19.98-1 (2020-01-26)
May 18 18:25:06 torradeira kernel: [    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.19.0-8-amd64 root=UUID=2367996c-bb6b-4589-a890-b0c0f07a79a7 ro quiet
May 18 18:25:06 torradeira kernel: [    0.000000] x86/fpu: x87 FPU will use FXSAVE
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-provided physical RAM map:
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009ffff] usable
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x0000000000100000-0x00000000cfb4ffff] usable
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x00000000cfb50000-0x00000000cfb65fff] reserved
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x00000000cfb66000-0x00000000cfb85bff] ACPI data
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x00000000cfb85c00-0x00000000cfffffff] reserved
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x00000000e0000000-0x00000000efffffff] reserved
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x00000000fe000000-0x00000000ffffffff] reserved
May 18 18:25:06 torradeira kernel: [    0.000000] BIOS-e820: [mem 0x0000000100000000-0x000000022fffffff] usable
May 18 18:25:06 torradeira kernel: [    0.000000] NX (Execute Disable) protection: active
May 18 18:25:06 torradeira kernel: [    0.000000] SMBIOS 2.5 present.
```

TO BE CONTINUED.... I hope......