Kubernetes na Globo.com

Uma visão geral

Cezar Sá
Desenvolvedor, Globo.com

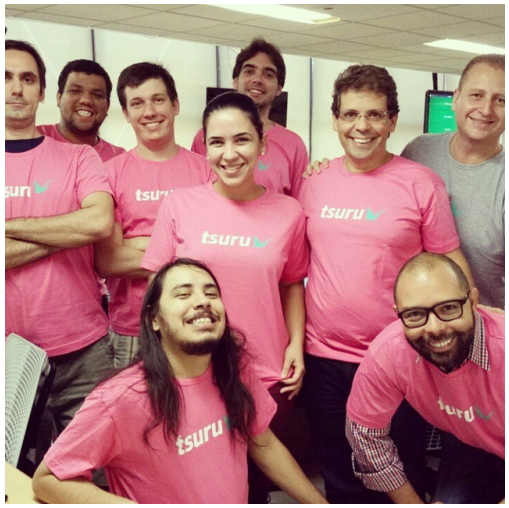
Quem sou eu?

github.com/cezarsa (https://github.com/cezarsa)

Plataforma de aplicações da Globo.com

Plataforma de aplicações

- + de 500 desenvolvedores
- + de 2000 aplicações
- + de 200 deploys por dia



Quem usa essa plataforma

E o Kubernetes?

~30% desse volume

Equipe

- <10 pessoas;
- Uma só equipe para desenvolver a plataforma, manter ela em produção e cuidar da sua operação;
- Papéis mistos, dev ❤ ops.

Timeline

- Como chegamos até aqui
- Onde estamos agora
- Para onde queremos ir

2012

- 1 deploy a cada ~2 semanas;
- Cada time mantendo seus proprios scripts de provisionamento;
- Pelo menos 6 maquinas físicas para cada aplicação:

```
3 prod (HA)
1 staging
1 qa
1 dev
```

- Até 1 mês para setup inicial.

2012 - Idéias

- Abstrair a infraestrutura
- Dar poder aos desenvolvedores
- Open source

2012 - PaaS

github.com/tsuru/tsuru (https://github.com/tsuru/tsuru)

```
$ tsuru app create <minha app> <minha plataforma>
$ tsuru app deploy <meu código>

Done!
```

- Não tem Dockerfile;
- Conceito de containers/maquinas virtuais ou físicas não é exposto;

2012

- Na verdade, containers não existem ainda como conhecemos eles;
- Docker foi anunciado em 2013;
- Primeiras versões do tsuru usavam o Juju da Canonical para orquestrar VMs;
- Lento, não confiável.

2013

- Docker chega e decidimos investir nisso.
- Como orquestrar containers docker em múltiplos Hosts?

github.com/tsuru/docker-cluster (https://github.com/tsuru/docker-cluster)

2014

- Primeira aplicação acessível pelo mundo externo em produção.

forum.techtudo.com.br/ (https://forum.techtudo.com.br/)

2015 - 2016

- Foco em resiliência e confiabilidade;
- Além de orquestrar containers, orquestrar também maquinas virtuais;
- Identificar maquinas virtuais problematicas e substitui-las automaticamente;
- Integração com IaaS para conseguir fazer isso;
- Mais aplicações em produção (Cartola, Globoplay);
- Migração física de datacenters.

2017 - 2018

- Kubernetes! Mas por que?
- Precisamos de mais features:

Aplicações stateful
Storage
Networking

- Isso tudo já esta pronto em outros orquestradores;
- Opções: docker swarm, kubernetes, mesos, etc.

2017 - 2018

- A premissa inicial ainda é mesma: não expor containers para o usuário final;
- Suporte no tsuru a múltiplos orquestradores:

```
$ tsuru cluster-add cluster1 kubernetes --addr https://<meu kubernetes>
$ tsuru cluster-add cluster2 swarm --addr https://<meu swarm>
```

- POC inicial com docker swarm; Diversos problemas de consistência e estabilidate.
- POC seguinte com Kuberentes; Tudo funciona!

Kubernetes hoje

- Cerca de 5 clusters em produção, gerenciados com terraform+puppet;
- Um repositório git para cada cluster;
- Helm sem Tiller;
- Charts renderizados com `helm template` e versionados;
- Usuário final sem acesso direto ao cluster, todo uso através do tsuru.

tsuru?

```
$ tsuru app-create minhaapp python # (ruby, go, php, ...)
```

Kubernetes:

- Criar CR app:minhaapp

```
$ tsuru app-deploy <meu código>
```

Kubernetes:

- Criar pod com base da imagem `plataform/python` + código;
- Dentro do pod criado instalar dependências da app;
- Gerar imagem da aplicação com código + dependências: `app/minhapp:v1`;
- Criar `deployment` usando imagem `app/minhapp:v1`;
- Criar `service` para tornar esse deployment acessível.

Migração

- Usuário não sabe se a app está sendo orquestrada pelo `docker-cluster` ou `kubernetes`;
- Migração transparente;
- Mas por que migrar?
- `docker-cluster` apesar de sua idade e ser pouco mantido funciona bem;
- Sem pressa e feita baseado em necessidade de features só presentes no kubernetes.

Kuberentes hoje

- Calico usado para rede;
- Rodando em VMs em sua maioria pequenas (4 CPUs, 8GB RAM)
- Maior cluster com ~90 VMs, ~1500 Pods;
- Por que não maquinas físicas:

Orquestração mais difícil. (solucionável com APIs de bare metal: MaaS, Openstack Ironic...)
Muitos ovos numa cesta só.
Containers não fornecem isolamento perfeito, muita coisa "vaza":

- Limites de IO não funcionam bem (cgroupv2 pode resolver)(https://andrestc.com/post/cgroups-io/);
- Kernel compartilhado é kernel panic compartilhado;
- OOM se torna bem mais imprevisivel em caso de overcommit de memória.

Lições aprendidas

- Testes, muitos testes automatizados.
- Unitários usando fake kubernetes e fake docker e de integração;
- Todos os dias são criados clusters `docker-cluster` e `kubernetes` no Google Cloud, AWS, Azure e Cloudstack(on-premise);
- A plataforma inteira do tsuru passa por esses testes em cada uma das clouds.

Lições aprendidas

- Instalações do kubernetes funcionam muito bem em clouds públicas, em um datacenter próprio não é tão fácil;
- Precisamos criar um cloud controller próprio:

github.com/tsuru/custom-cloudstack-ccm (https://github.com/tsuru/custom-cloudstack-ccm)

- Arquitetura de redes do datacenter influencia diretamente na configuração do seu plugin de redes:

github.com/tsuru/remesher (https://github.com/tsuru/remesher)

Lições aprendidas, falando em redes

- Sua rede vai quebrar de muitos jeitos, exemplos:
- kube-dns/coredns: Uma falha do seu DNS afeta o seu cluster inteiro e ele precisa ser escalado dimensionado de forma apropriada;
- Races no SNAT e no DNAT no kernel podem piorar mais ainda as latências no resolver:

tech.xing.com/a-reason-for-unexplained-connection-timeouts-on-kubernetes-docker-abd041cf7e02 (https://tech.xing.com/a-reason-for-unexplained-connection-timeouts-on-kubernetes-docker-abd041cf7e02)

blog.quentin-machu.fr/2018/06/24/5-15s-dns-lookups-on-kubernetes/ (https://blog.quentin-machu.fr/2018/06/24/5-15s-dns-lookups-on-kubernetes/)

- Usar o node local dns ajuda com esses 2 problemas, resolver nomes fora do cluster vai direto Host -> Resolver externo, sem passar pelo resolver do cluster:

github.com/kubernetes/kubernetes/tree/master/cluster/addons/dns/nodelocaldns

(https://github.com/kubernetes/kubernetes/tree/master/cluster/addons/dns/nodelocaldns)

Lições aprendidas, falando em redes

- Bugs em equipamentos de rede (F5) negociando MTU com redes calico (cuidado com o MTO para levar em considerção o encapsulamento IPIP);
- É preciso ter uma noção do tamanho do cluster no momento de alocar as redes/máscaras para os Pods e Services;
- Load Balancers dependem do cloud controller diretamente, e quanto menos conhecida for sua cloud mais trabalho isso vai causar para você.

Lições aprendidadas

- Soluções de storage próprias em datacenters próprios também dependem de integrações externas:

kubernetes-csi.github.io/docs/ (https://kubernetes-csi.github.io/docs/)

Futuro

- Clusters criados dinâmicamente
- Aumentar uso e integração com clouds públicas

Thank you

Cezar Sá
Desenvolvedor, Globo.com
cezarsa@gmail.com (mailto:cezarsa@gmail.com)
https://github.com/cezarsa (https://github.com/cezarsa)
Estamos contratando!