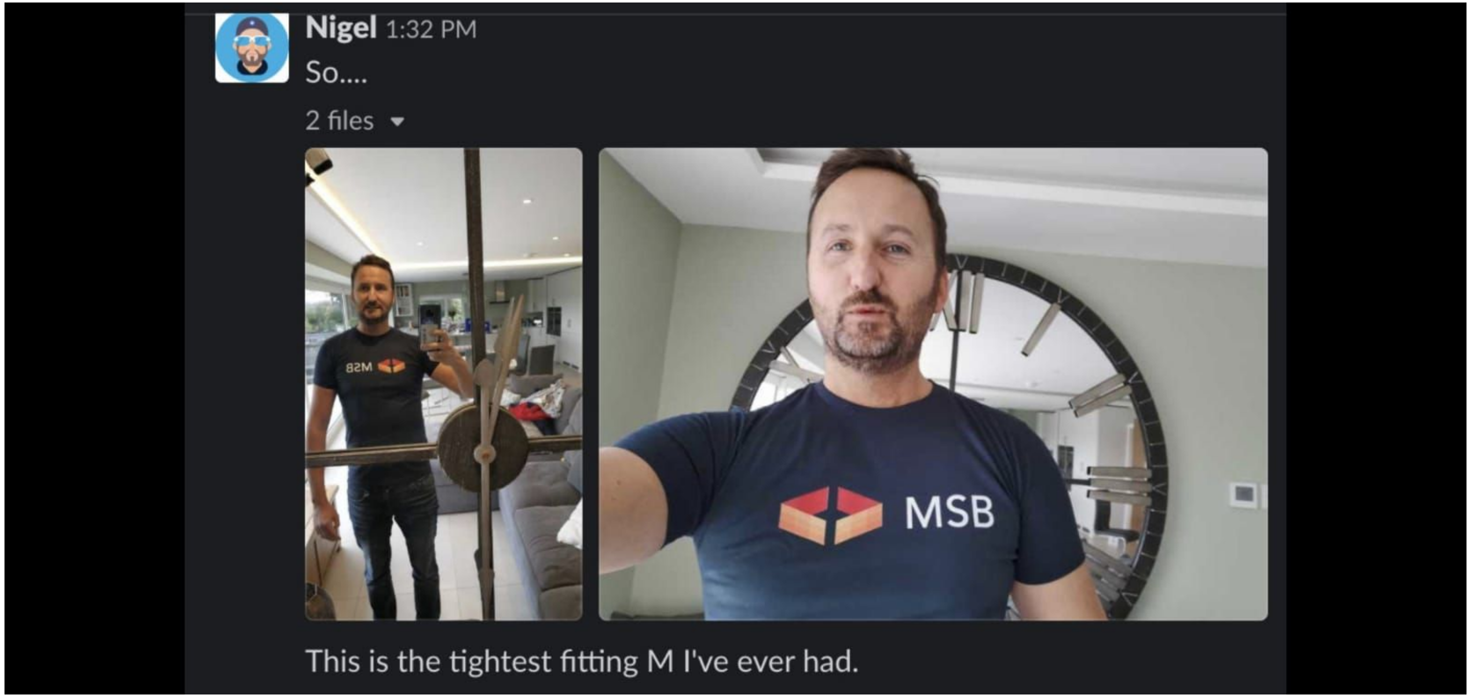
MSB

Creating a Kubernetes cluster farm

MSB

What is MSB

- Hands-on immersive kubernetes training platform on real infrastructure
- Whole tech stack is built for the purpose of teaching Kubernetes
- Content developed by Nigel Poulton, the author of “The Kubernetes book”
- Started out of frustration with the current state of online learning



Nigel 1:32 PM

So....

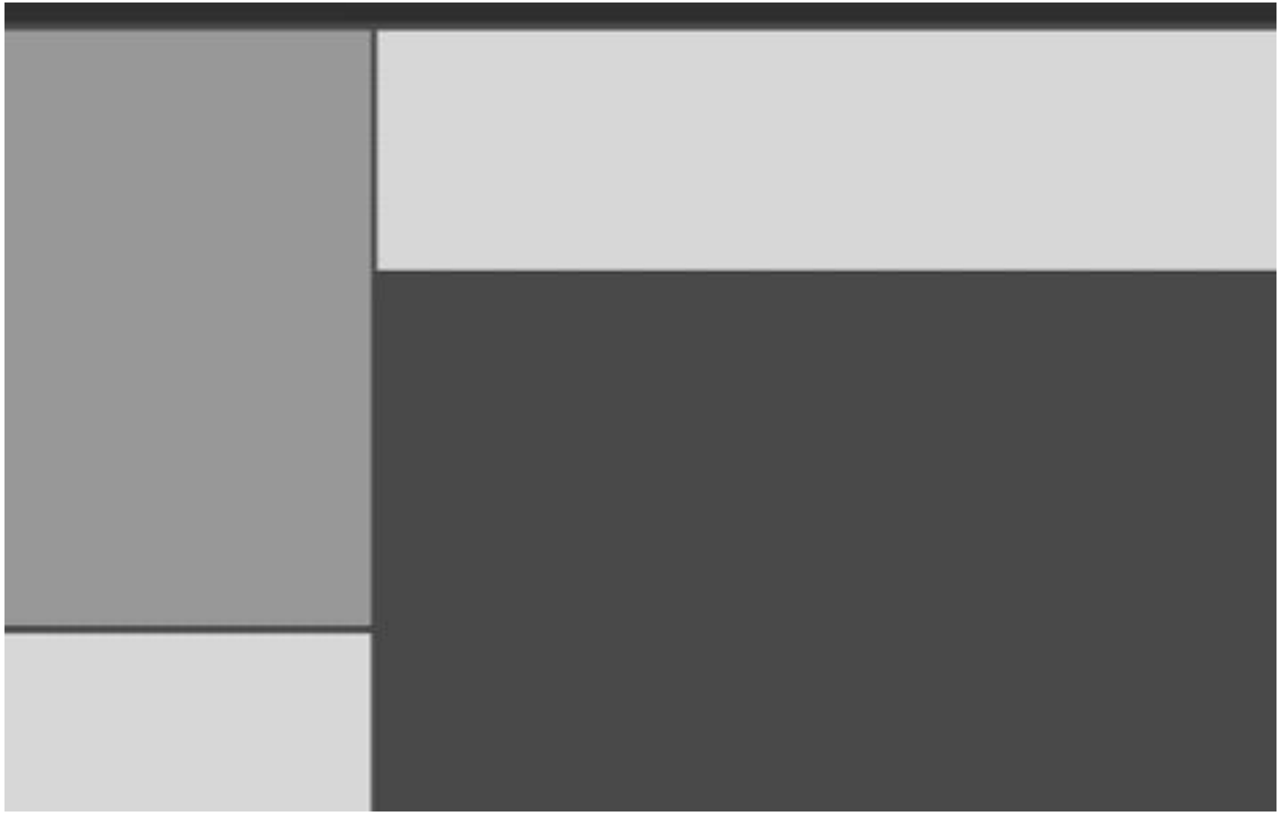
2 files

This is the tightest fitting M I've ever had.

MSB

Short MSB demo

Evolution of MSB



MSB

Infrastructure evolution

- MVP version (hacky version on top of Digital Ocean)
- Kubeception using VMs
- Kubeception using containerd

MSB

The MVP version

Goals

- User of Magic Sandbox should get their own cluster to train on
- The cluster should have an active terminal connection and a live visual representation of its state
- Create a proof-of-concept to gather proof points for Demo day

Constraints

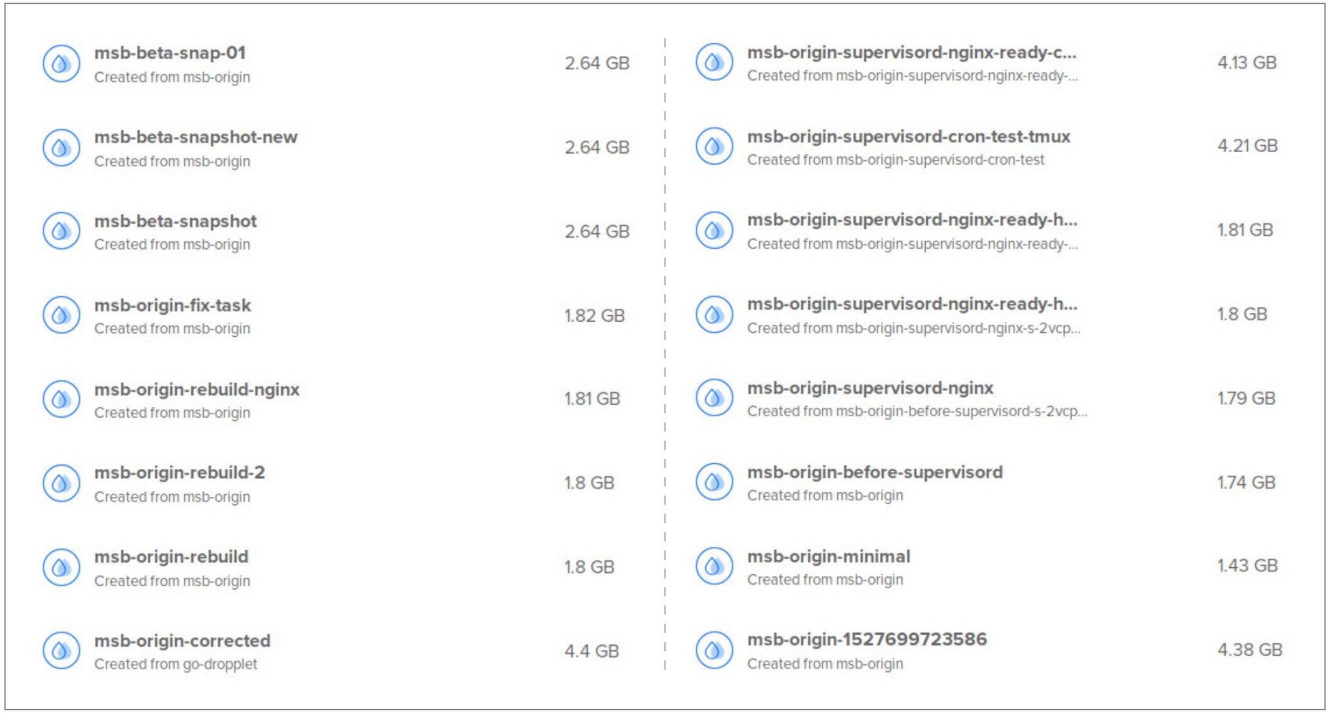
- Very limited time to implement both frontend and backend
- Limited knowledge how to actually implement disposable k8s clusters on demand
- Limited knowledge if this implementation is possible
- Limited idea of what actually Magic Sandbox does :)

MSB

Our solution

- Generating a base Digital Ocean image with a minikube init script
- Scaling it manually
- Routing using traefik.io, then a custom stateless solution
- Using Digital Ocean labels to keep track of the clusters
- “Trivial” polling of cluster state every few seconds

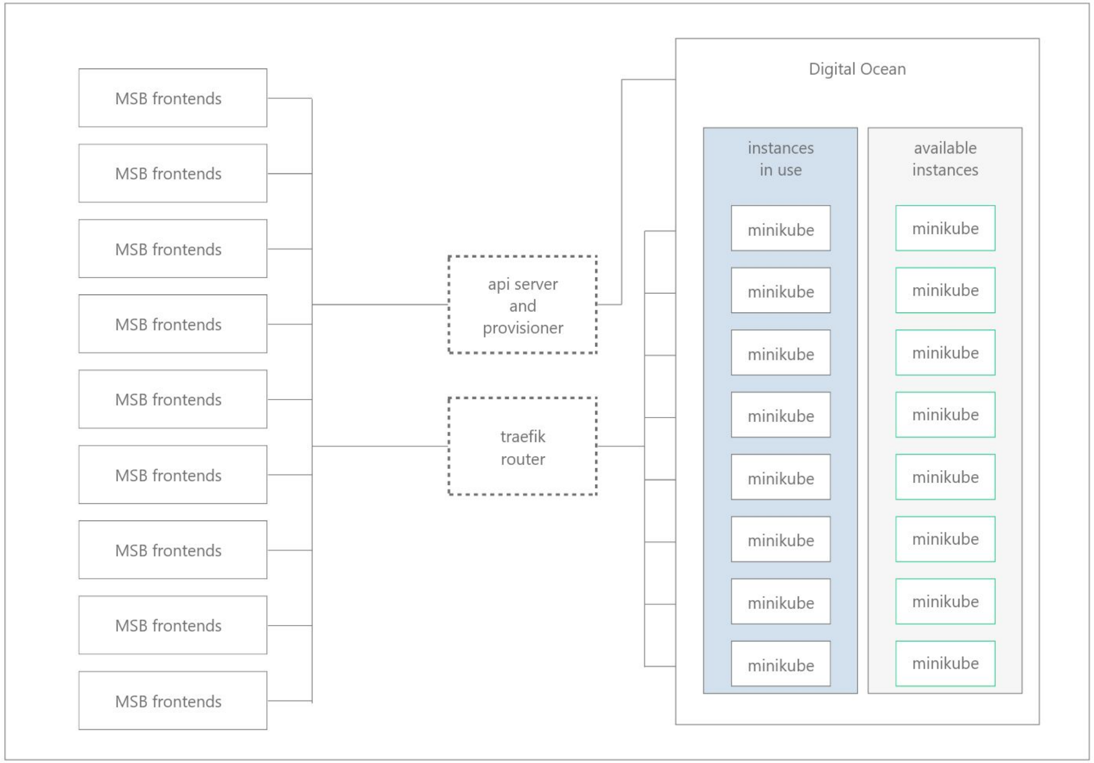
Hacky Digital Ocean snapshots



msb-beta-snap-01 Created from msb-origin	2.64 GB
msb-beta-snapshot-new Created from msb-origin	2.64 GB
msb-beta-snapshot Created from msb-origin	2.64 GB
msb-origin-fix-task Created from msb-origin	1.82 GB
msb-origin-rebuild-nginx Created from msb-origin	1.81 GB
msb-origin-rebuild-2 Created from msb-origin	1.8 GB
msb-origin-rebuild Created from msb-origin	1.8 GB
msb-origin-corrected Created from go-dropplet	4.4 GB
msb-origin-supervisord-nginx-ready-c... Created from msb-origin-supervisord-nginx-ready-...	4.13 GB
msb-origin-supervisord-cron-test-tmux Created from msb-origin-supervisord-cron-test	4.21 GB
msb-origin-supervisord-nginx-ready-h... Created from msb-origin-supervisord-nginx-ready-...	1.81 GB
msb-origin-supervisord-nginx-ready-h... Created from msb-origin-supervisord-nginx-s-2vcp...	1.8 GB
msb-origin-supervisord-nginx Created from msb-origin-before-supervisord-s-2vcp...	1.79 GB
msb-origin-before-supervisord Created from msb-origin	1.74 GB
msb-origin-minimal Created from msb-origin	1.43 GB
msb-origin-1527699723586 Created from msb-origin	4.38 GB

Architecture overview

MSB



Our “scaling” solution

```go
// a buffered channel containing available droplets
var AvailableDroplets chan godo.Droplet = make(chan godo.Droplet, 10)

func DropletBuffer(client *godo.Client) {
  // collecting available droplets in case of a restart
  availableDroplets, _ := AvailableDropletList(context.Background(), client)

  // goroutines creating new instances and pushing into the buffered channel
    go DropletGeneratorAndPusher()
    go DropletGeneratorAndPusher()
    go DropletGeneratorAndPusher()
    go DropletGeneratorAndPusher()
    go DropletGeneratorAndPusher()
    go DropletGeneratorAndPusher()
    go DropletGeneratorAndPusher()
    go DropletGeneratorAndPusher()

    for index, droplet := range availableDroplets {
        AvailableDroplets <- droplet
    }

}
```

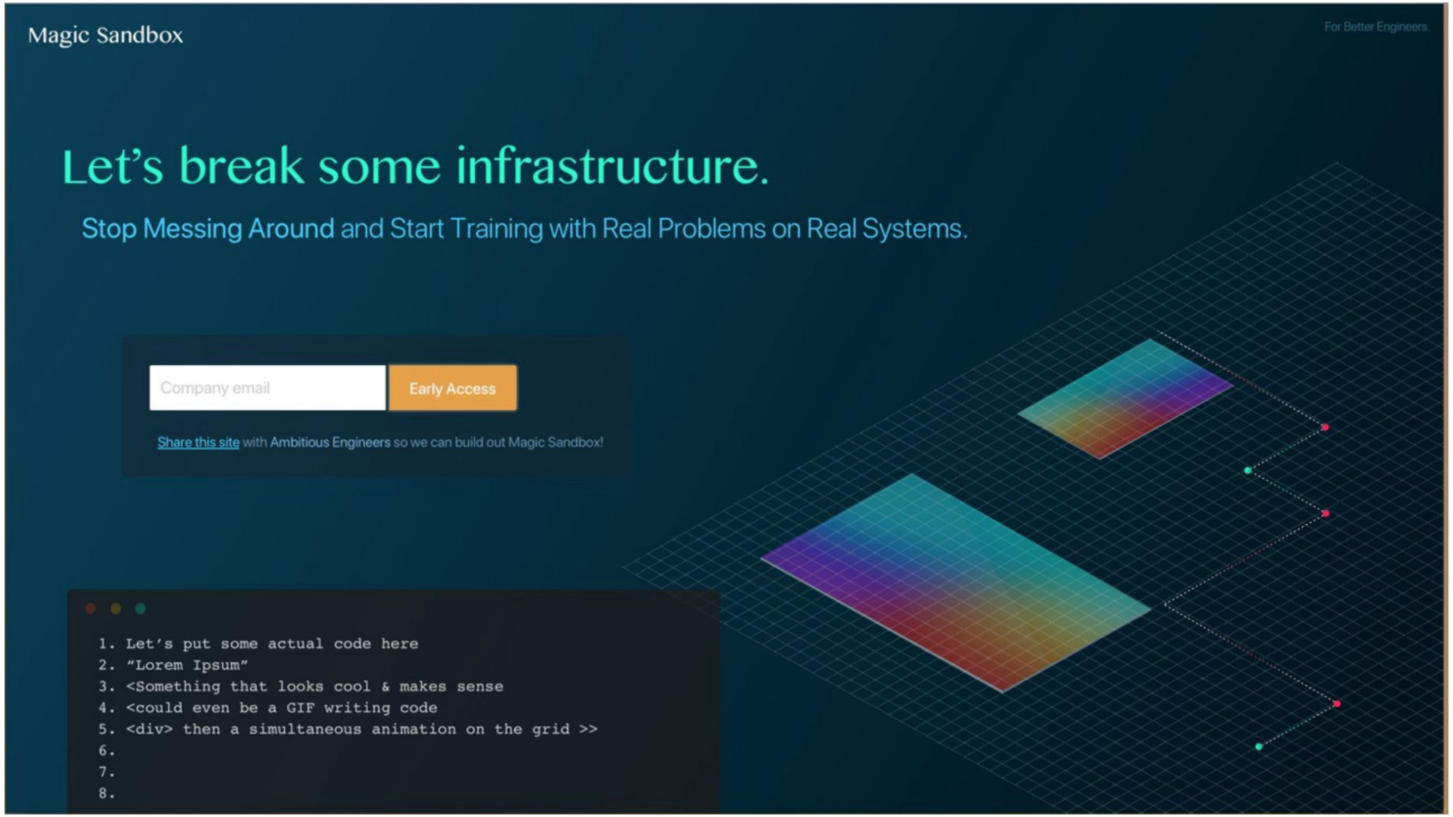
MSB

Problems we were encountering

- DO labeling system was failing
- Issues with traefik.io and it’s bad failure scenario - ended up rewriting the router to a stateless router
- Very limited scalability - Hacker News launch exceeded our capacity in 2 minutes
- Lots of manual monitoring and interventions needed

MSB

MSB landing page at the time



Magic Sandbox

For Better Engineers.

Let’s break some infrastructure.

Stop Messing Around and Start Training with Real Problems on Real Systems.

Share this site with Ambitious Engineers so we can build out Magic Sandbox!

```
Let’s put some actual code here
“Lorem Ipsum”
<Something that looks cool & makes sense
<could even be a GIF writing code
<div> then a simultaneous animation on the grid >>
```

MSB platform at the time

MSB

What worked well

- Least problems were with Kubernetes itself, minikube booted properly every time
- Initial MVP showed enough traction and interest to complete fundraising
- Basic ideas of MSB have been solidified
- We’ve realized that a lot of problems we’ve been having would have been handled well by Kubernetes itself

MSB

Initial Kubeception version

MSB

Goals

- With proper funding and team in place, build a infrastructure setup
- Infrastructure should support future MSB goals
- Create a large k8s cluster to serve smaller customer clusters
- Offload as much work to Kubernetes as possible
- Redesign frontend <-> backend communication
- More resources but bigger expectations

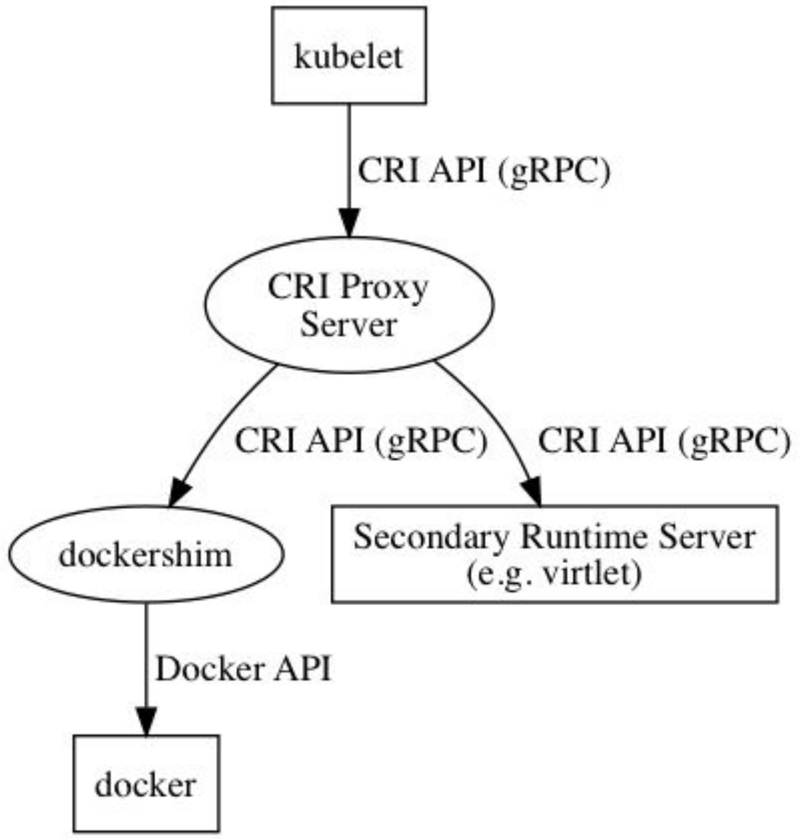
MSB

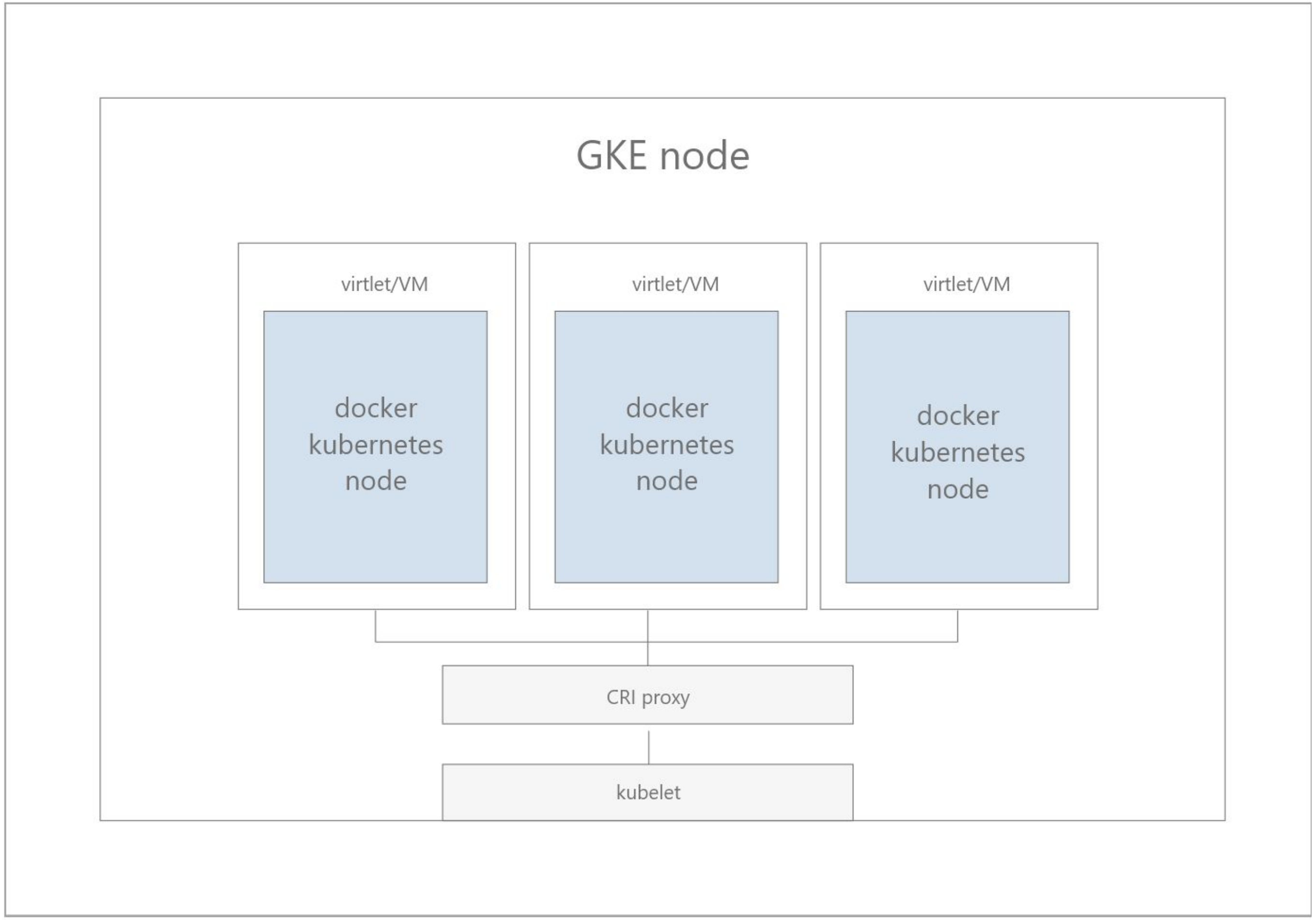
Constraints

- Speed to market is again a very important driver
- Limited knowledge how to implement such a system, apart from a few blog posts
- Limited knowledge of whether large cloud providers support such “advanced” features

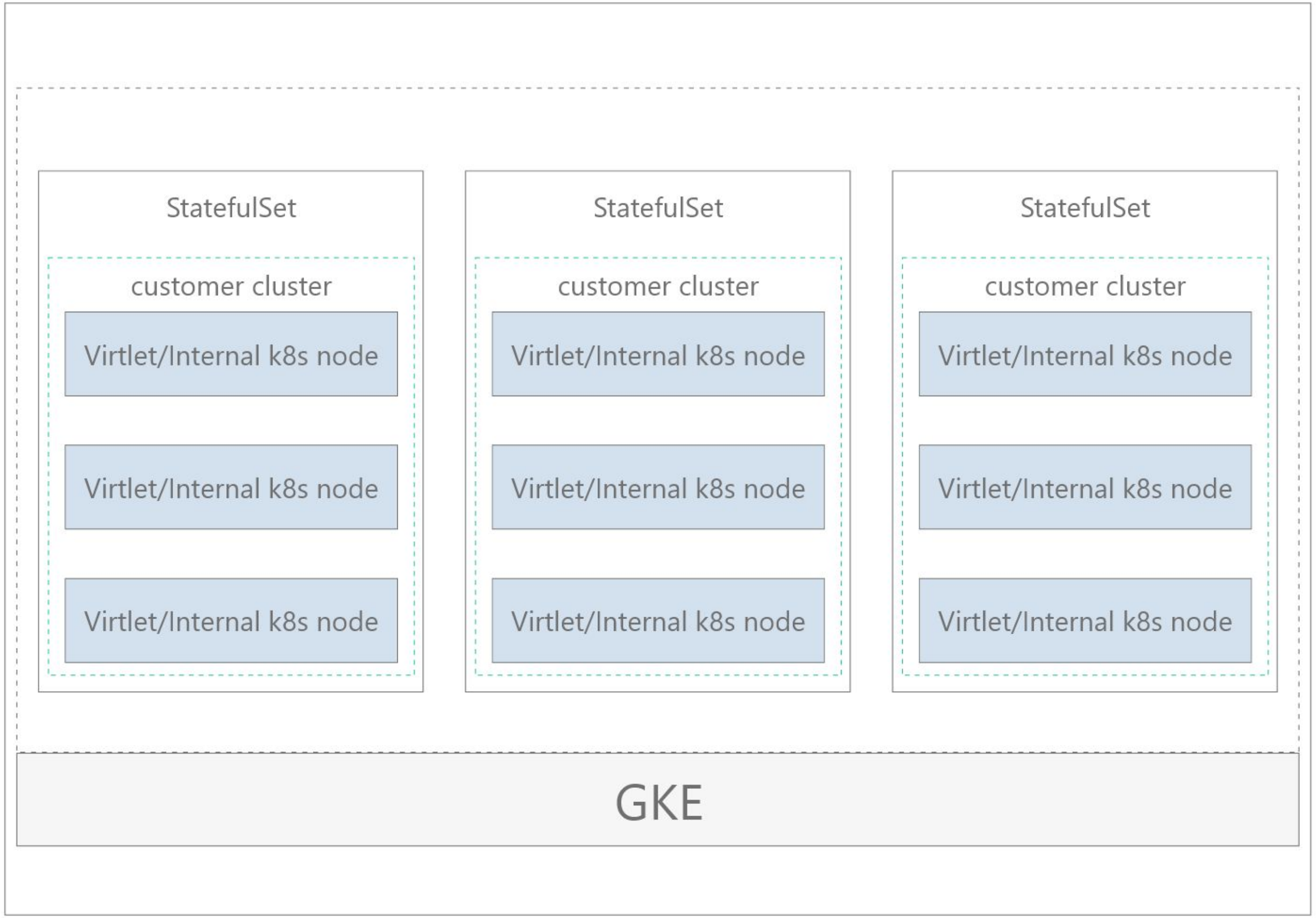
Solution came with virtlets

- Kubernetes CRI implementation for running VM workloads
- Container Runtime Interface (CRI) – a plugin interface which enables kubelet to use a wide variety of container runtimes, without the need to recompile
- Allowing running VMs as regular pods, with full integration with other workload objects




GKE node
virtlet/VM
docker kubernetes node
virtlet/VM
docker kubernetes node
virtlet/VM
docker kubernetes node
CRI proxy
kubelet

MSB


StatefulSet
customer cluster
Virtlet/Internal k8s node
Virtlet/Internal k8s node
Virtlet/Internal k8s node
StatefulSet
customer cluster
Virtlet/Internal k8s node
Virtlet/Internal k8s node
Virtlet/Internal k8s node
StatefulSet
customer cluster
Virtlet/Internal k8s node
Virtlet/Internal k8s node
Virtlet/Internal k8s node
GKE

MSB

A custom operator to tie cluster resources together

- A custom operator ensures setup of the cluster with supporting objects (proper labels and annotations, networking, ingress)
- Ensuring there are enough “ready” clusters available in the pool
- Tracks utilization and has a percentage-based scheduling to account for spikes in usage
- Assigning a cluster to a user is a matter of changing the labels
- Build using kubernetes-sigs/kubebuilder which made implementation much easier

MSB

Frontend communication using API watchers

- Cluster state polling moved to websocket-based K8s api watchers
- Using event sourcing to recreate the state on the cluster
- Significantly reduced traffic
- All handled by traefik.io ingress controller on the main cluster without issues
- Implemented live traffic capture by hooking into docker network interfaces

MSB

Things that went well

- Again no issues came from Kubernetes itself, rather our misuse of its mechanisms
- Networking/services/ingress is rock solid
- We could finally start implementing proper monitoring and alerting
- Once all infra scripts and yamls were done, it was easy to dispose and recreate the whole “big” cluster in minutes
- GKE works as expected and gave us no issues at all
- Proper backend architecture was starting to take shape

MSB

Issues faced

- Running a docker container inside a VM inside a K8s cluster didn’t feel like the best solution
- Boot up times were unnecessarily long (~3 minutes)
- VM setup not fully managed by k8s mechanisms
- Low configuration flexibility - a 3 Pod StatefulSet whose first “Pod” was hardcoded to be the master node

MSB

Switching to containerd

MSB

Goals

- Simplify architecture
- Try to get rid of VMs
- Make the clusters configurable (different number of nodes, different k8s versions...)
- Make the clusters as close to production-grade as possible (support for LoadBalancers, Ingress, PersistentVolumes, Helm support...)

MSB

Constraints

- Seamless transition for our existing users
- Not trivial to get a container runtime functional inside another container
- Try and make it the “final” rework :)

MSB

Containerd to the rescue

- A k8s node needs only 3 things:
  - container runtime
  - kubelet
  - kube-proxy
- Internal container runtime was hard to set up with Docker
- containerd allowed us to set up an internal container runtime in the container, simplifying the architecture by removing the VM

MSB

Assembling clusters on demand

- Internal operator now manages two pools of nodes - master and worker nodes, which get connected on demand
- Connection occurs when a user requests a cluster and within 10 seconds the Worker nodes are usable
- This allows per-class customization of features

MSB

Heavily dependent on Kubernetes

- Listening to kubernetes events to enable things like LoadBalancer services
- Extending the API for most introspection needs - which fits well in existing tooling
- Using community provided tooling (like the Prometheus operator) for out of the box solutions
- By choosing to rely on Kubernetes mechanisms we could easily switch cloud vendors currently

MSB

Issues faced

- Networking was a big issue, solved by switching to Cilium and BPF
  - Caused by inability to run in privileged mode
- Solving some issues required “senior-level” engineering to solve
- Not easy always to find documentation or shared experiences online for advanced issues

MSB

What we’ve learned

- Kubernetes is a very good and very flexible solution
- Work with k8s, not against it
- Use all available mechanisms! (init- and readniessProbes, k8s events, DaemonSets, Custom Operators...)
- Labels and annotations are very reliable and a good way to decouple systems
- **Without k8s this system would be much harder to build**, our experience was usually the bottleneck
- Frontend is also very hard
- Having a great team is the most important thing!

MSB

Thank you!

Try us out at msb.com