

Docker for fun + profit

A story about (more generally) deploying and managing containers in prod



You (hopefully)

What's a container?!

Target Audience

I am container god



Speaker Notes

- 0 should be like "I have kind of used Linux but I don't really know what this container lark is about"
- 3 (audience start) should be like "I have used containers! I created a docker compose."
- 7 (audience end) is like "I have shipped things to Kubernetes/Swarm/Whatever and they're facing traffic"
- 10 is like home brewing clusters with kubeadm, handling storage abstractions and so fourth.

Please Interrupt.



Speaker Notes

Action Required

- Put up the “raise questions” URL.

Talking Points

- The presentation is a guide. We might finish it, or we might not - it doesn't matter. Please ask questions
- The goal of this is to spark interest, not really to answer all questions. Please, ask them -- but also don't worry too much if you don't get it; the slides will be sent around, and they mostly have links where you can learn more.

Please Contribute!



Speaker Notes

Talking Points

- I know some things. But not all things.
- If you know things (or if your knowledge contradicts mine) please voice it!
We're all a collaborative team here.

So who is responsible for production?



Speaker Notes

Interactive instructions

- Hands up if you're responsible for production
- Keep them up if you built it?
- Keep them up if you broke it?

The problem



Speaker Notes

- Skip these. This. Finish on the slide with the giant mountain.

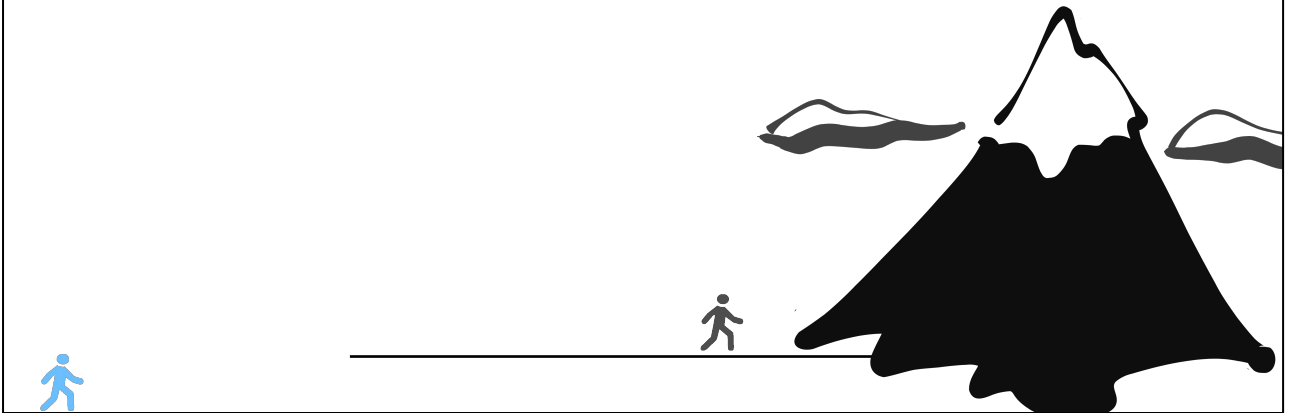
Building a server is hard



Speaker Notes

- Skip these. This. Finish on the slide with the giant mountain.

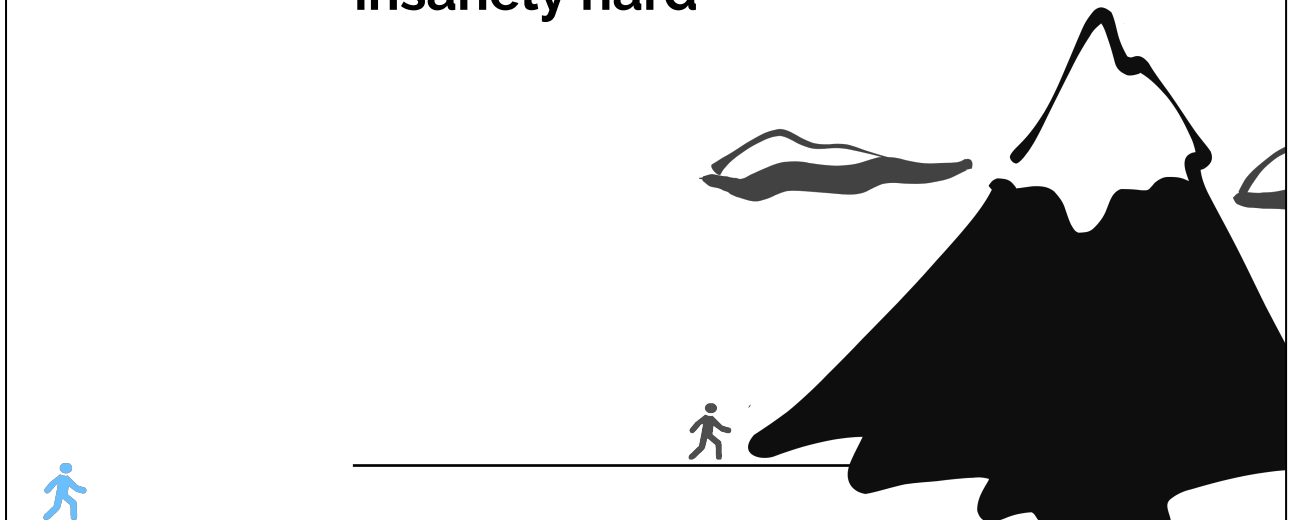
Maintaining servers is super hard



Speaker Notes

- Skip these. This. Finish on the slide with the giant mountain.

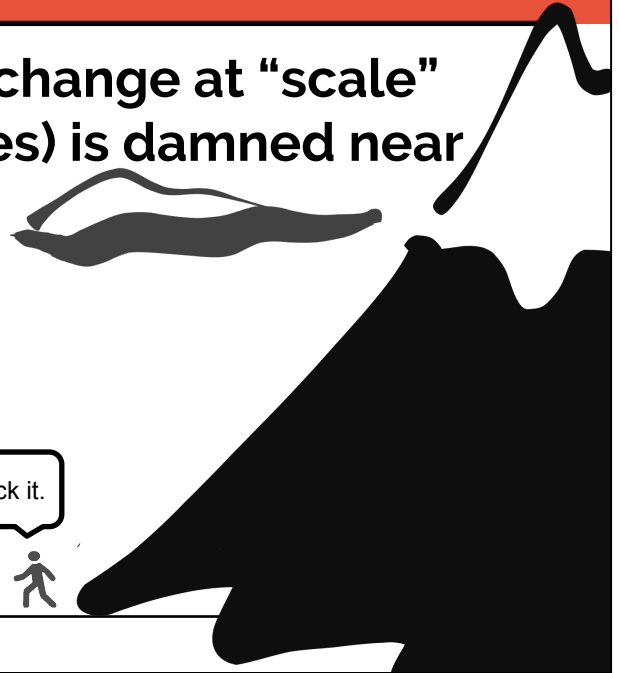
Maintaining servers for years is insanely hard



Speaker Notes

- Skip these. This. Finish on the slide with the giant mountain.

Introducing change at “scale”
(30+ machines) is damned near
impossible.



Speaker Notes

Talking Points

- Running Linux machines finds problems other software eng has solved:
 - Version Control
 - Code Review
 - Collaboration
 - Change Management

Further Reading

- <https://www.safaribooksonline.com/library/view/site-reliability-engineering/9781491929117/ch05.html>

Solution

Containers!



Further Reading

- <https://www.opencontainers.org/>

So who's used containers

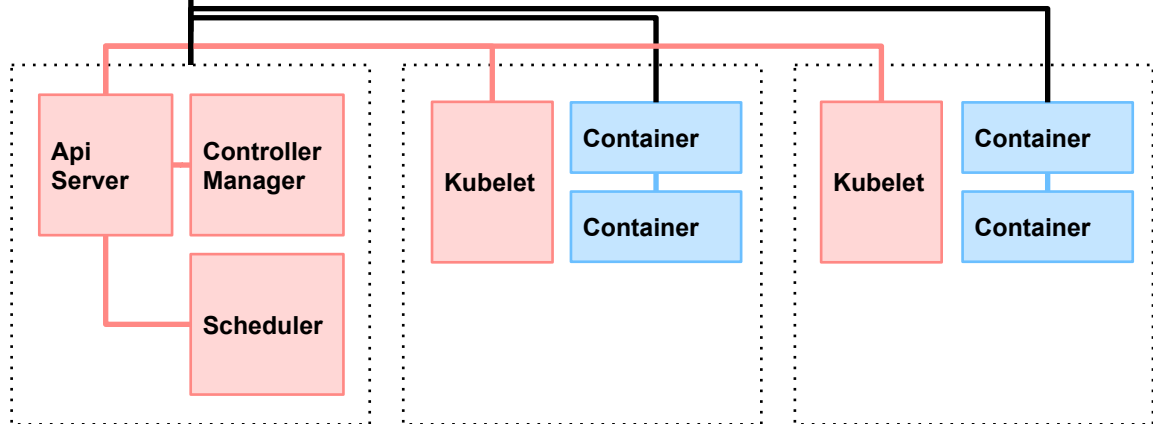


Speaker Notes

Interactive instructions

- Hands up if you have used docker before?
- Keep them up if you use them in dev
- Keep them up if you use them in prod
- Keep them up if you use Kubernetes
- Keep them up if you homebrew your own clusters? (You're going to be bored here)

Architecture Diagram



Speaking Notes

- We're going to cover containers, then Kubernetes. The blue, then the pink
- We'll cover how to use them, how they work and any other questions you might have
- Don't worry about the terms just yet.

Pray to the live demo gods



Speaker Notes

- \$ helm install stable/\${SOFTWARE}.
- A good one is wordpress

Here's one I prepared earlier

<http://35.205.247.239/>

Admin: /admin

User: user

Password: dwH99M9hfE



I also did wordpress.
Magento was ...
slow.

<http://35.205.166.8>



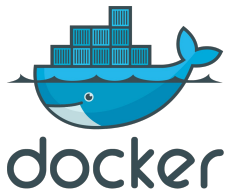
Containers

namespaces + cgroups



Speaker Notes

- Build once
- Deploy anywhere



A nice wrapper around containers.
Not the only one.



Speaker Notes

- Docker is most popular, and well supported. There are also others! But perhaps the most important thing to know is there is a standard format for them, and they're interchangeable.

Further Reading

- <https://www.docker.com/>
- <https://github.com/coreos/rkt>
- <https://linuxcontainers.org/>
- <https://github.com/opencontainers>

Who has played snake



Speaker Notes

- Hands up who's played snake?
- Keep them up if you played it on a nokia 3310
- Let's all go back to nokia 3310's

Fun!



Build

Locally, or in CI build an “image”
- a deployable container



Push

Push the image to a “registry” - a
place to put these containers
until they’re deployed



Deploy

Update the deployment
configuration so it uses the new
container



Speaker Notes

- This process lends itself extremely well to automation.
- Mostly, developers won’t do this -- they’ll simply mark their code as deployable, and it’ll be deployed.

Build

```
FROM nginx:latest

# Define the snake repo in a handy environment variable for easy replacement
ENV SNAKE_REPO="https://github.com/PKief/Snake.git"

# Install git so we can download the snake repo
RUN apt-get update && \
    apt-get install --yes \
        git

# Clone the repo to a directory that NGINX can access
RUN git clone ${SNAKE_REPO} /var/www/snake

# Modify NGINX configuration to serve our snake game
RUN sed --in-place 's/usr/share/nginx/html/var/www/snake/' /etc/nginx/conf.d/default.conf
```



```

Dec-13 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libspn100-amd64_3.9.0-4+deb9u2 [98 kB]
Dec-13 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libxvutil1 amd64_3.9.0-4 [12.4 kB]
Dec-13 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libklibc-support amd64_1.35-2+deb9u1 [41.9 kB]
Dec-13 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libcrypto2 amd64_1.0.2g-2 [115 kB]
Dec-17 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libncursesw5 amd64_1.3.92-4+deb9u1 [131 kB]
Dec-18 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libasplos-kprobs amd64_1.35-4+deb9u1 [3.6 kB]
Dec-19 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libstrgrfsgrf amd64_6.0.0-realtime-9.1.1 [276 kB]
Dec-19 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libncurses6 amd64_6.0-8+deb9u1 [130 kB]
Dec-21 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libmsl2-modules-amd64_2.37.107-gf979040e-dfarg3 [308.3 kB]
Dec-22 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libssl3-2 amd64_2.1.27-1+b1-gf979040e-dfarg3 [309 kB]
Dec-23 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 liblibio-perl amd64_2.2.4-4+deb9u2 [54.4 kB]
Dec-23 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libncurses5 amd64_6.0-8+deb9u1 [130 kB]
Dec-23 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libstring2-4 amd64_1.3.0-1 [70.1 kB]
Dec-26 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libzstd1 amd64_0.7.9-3 [41.0 kB]
Dec-26 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libncursesw5 amd64_6.0-8+deb9u1 [130 kB]
Dec-28 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libssl3-3 amd64_1.1.0-3 [138 kB]
Dec-29 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libicu57-variants amd64_7.2.1-2-1+deb9u1 [120 kB]
Dec-30 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 liberror-perl amd64_0.1002-1 [36.9 kB]
Dec-30 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 git-man amd64_2.11.0-3+deb9u1 [1.1 MB]
Dec-32 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 git amd64_1:2.11.9-3+deb9u2 [419 kB]
Dec-33 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libpopt0 amd64_1.16-9+deb9u2 [49.4 kB]
Dec-33 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libncurses6 amd64_6.0-8+deb9u1 [120 kB]
Dec-33 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 netbase-all_1.9-4 [18.1 kB]
Dec-36 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 krb5-locales all_1.15-4+deb9u1 [80.8 kB]
Dec-37 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libc6-amd64_2.1 [126 kB]
Dec-37 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 openssl amd64_1.0.2g-2 [1.1 MB]
Dec-39 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 ca-certificates all_2.0.0.130+nss+deb9u1 [102 kB]
Dec-40 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libmsl2-modules-amd64_2.37.107-gf979040e-dfarg3 [308.3 kB]
Dec-41 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libuv1 amd64_2.1.2-3-1+deb9u2 [52.5 kB]
Dec-41 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 libncursesw5 amd64_6.0-8+deb9u1 [130 kB]
Dec-43 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 patch amd64_2.7.6-2+deb9u1 [112 kB]
Dec-44 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 remove all_1.0-3-1 [2.6 kB]
Dec-44 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 rsyslog-amd64_8.24.0-1 [360 kB]
Dec-46 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 ncurses amd64_1:1.0-9-1+deb9u1 [393 kB]
Dec-47 http://cdm.familj.meb.dedim.org/debian stretch/main amd64 publicsuffix all_20170121-2040-0+deb9u1 [103 kB]

```



Push

Repository Tags

Compact Expanded

☐

1 - 25 of 78

<>

Filter Tags...

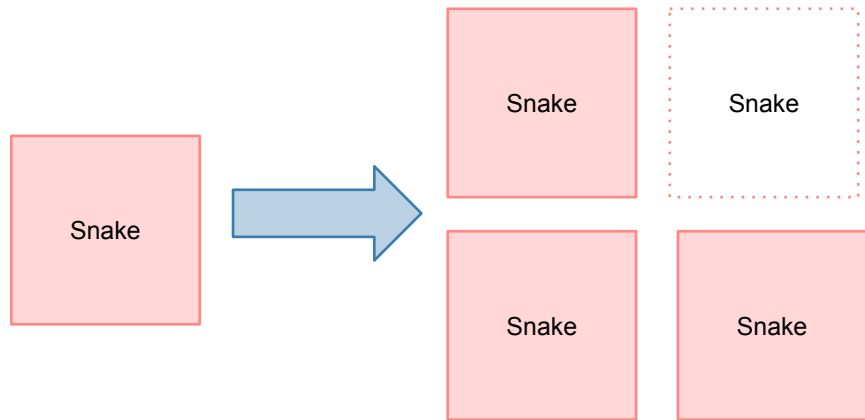
TAG	SIGN	LAST MODIFIED ↓	SECURITY SCAN	SIZE	IMAGE
<input type="checkbox"/> 7.1-latest		19 hours ago	13 High	136.6 MB	SHA256: 925715a21057
<input type="checkbox"/> 7.0-latest		19 hours ago	15 High	136.3 MB	SHA256: 928eb654719e
<input type="checkbox"/> 5.6-latest		19 hours ago	13 High	140.9 MB	SHA256: f2cb2ca6ee21
<input type="checkbox"/> 7.1.15-1		2 months ago	24 High • 45 fixable	184.2 MB	SHA256: c31a08bbf13e
<input type="checkbox"/> ad-hoc-reduce-duplica...		5 months ago	67 High • 143 fixable	167.9 MB	SHA256: 68c688918f65

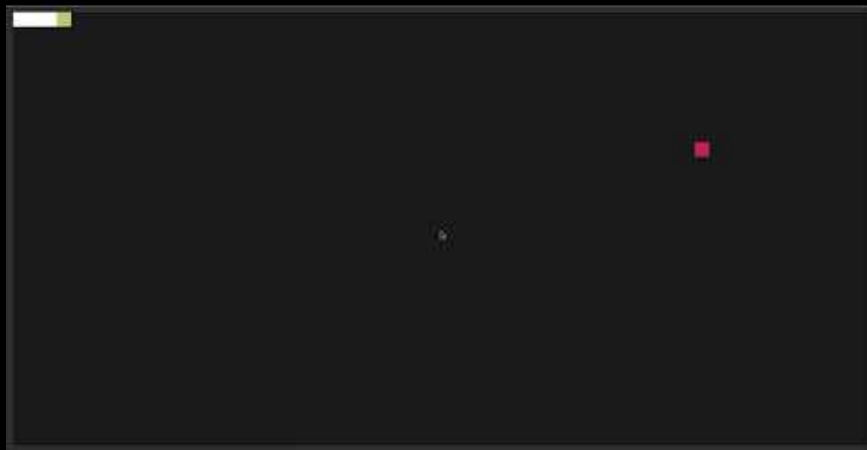



```
hello@cloudshell:~/tmp/na.0r7w4m0k (littleman-co)$ docker push quay.io/littlemeco/snake
The push refers to repository [quay.io/littlemeco/snake]
c2b79a0f05fa: Pushed
c9a320f1a0a: Pushed
2e306ac703da: Pushing [=====] 57.43MB/59.15MB
88a25f0e442e: Pushed
c0b3f9544e4c: Pushing [=====] 1.832MB/55.27MB
```



Deploy

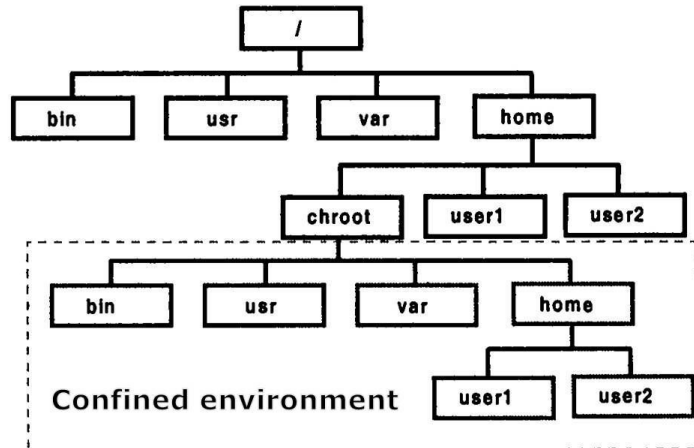




How does that work?!



mount namespace (~chroot)



Speaker Notes

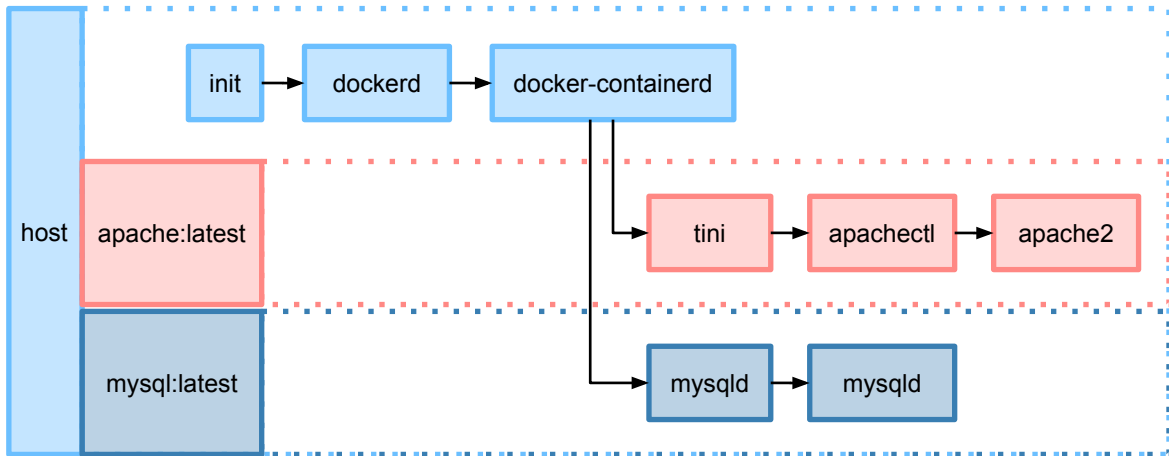
Explanation

- chroot → **change** root
- It's a root file system within a file system
- Traditionally used for jailing in shared hosting, but also used in docker.

Further Reading

- https://en.wikipedia.org/wiki/Operating-system-level_virtualization
- <http://queue.acm.org/detail.cfm?id=2898444>
- <http://crosbymichael.com/creating-containers-part-1.html>

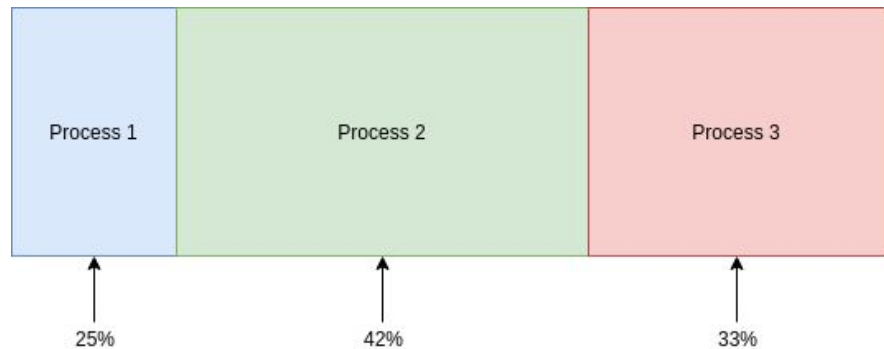
Process Namespace



Speaker Notes

- All namespaces provide some level of isolation from the other namespaces
- The above namespace is a “process” namespace
- The host can see processes in containers. Containers can’t see process on the host, or in other containers.
- All namespaces work with a variation of this flavour.

Control Group



Speaker Notes

- This can be CPU, Memory or a bunch of other things

Design Notes

- I need to add notes here. Memory or whatever.

Further Reading

- <https://en.wikipedia.org/wiki/Cgroups>

Deploy to what?



Speaker Notes

- Containers require a runtime, the most common of which is Docker. However, once that is there you can run it wherever;
 - VM (EC2, Google Cloud, Rackspace, 1und1, Whatever)
 - AWS container service
 - Docker Swarm
 - App engine
- Once we have containers though, we can do some tremendously cool things with Automation
- All of those cool things have already been done with Kubernetes.

We don't have to use Kubernetes to use containers. But it's there, and it makes our lives fundamentally easier -- why not?



Take a bunch of machines,
make them a single logical
machine



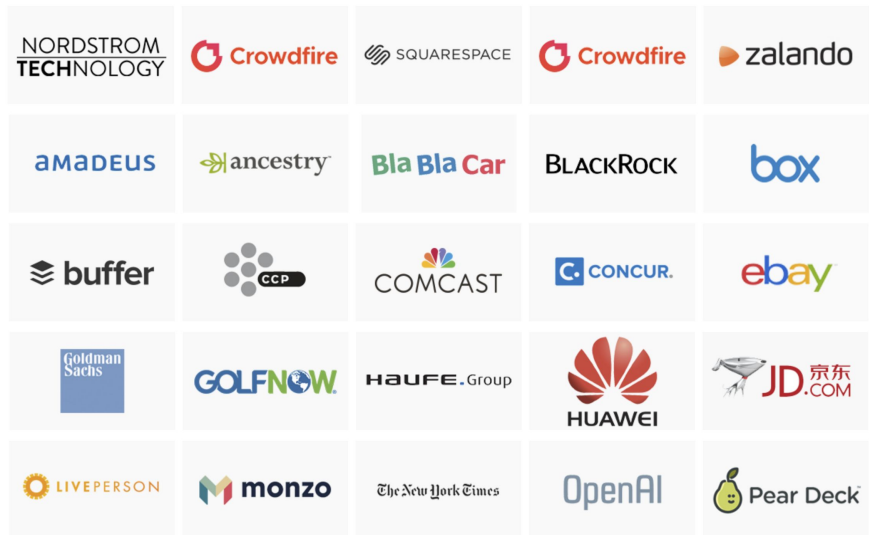
Speaker Notes

- It would be worth adopting containers to get Kubernetes
- It makes operations much, much simpler and more reliable

Further Reading

- <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>
- <https://www.linux.com/news/learn/chapter/Intro-to-Kubernetes/2017/3/what-kubernetes>
- <https://cloudplatform.googleblog.com/2016/09/bringing-Pokemon-GO-to-life-on-Google-Cloud.html>
- <https://kubernetes.io/case-studies/>

Who Uses It?



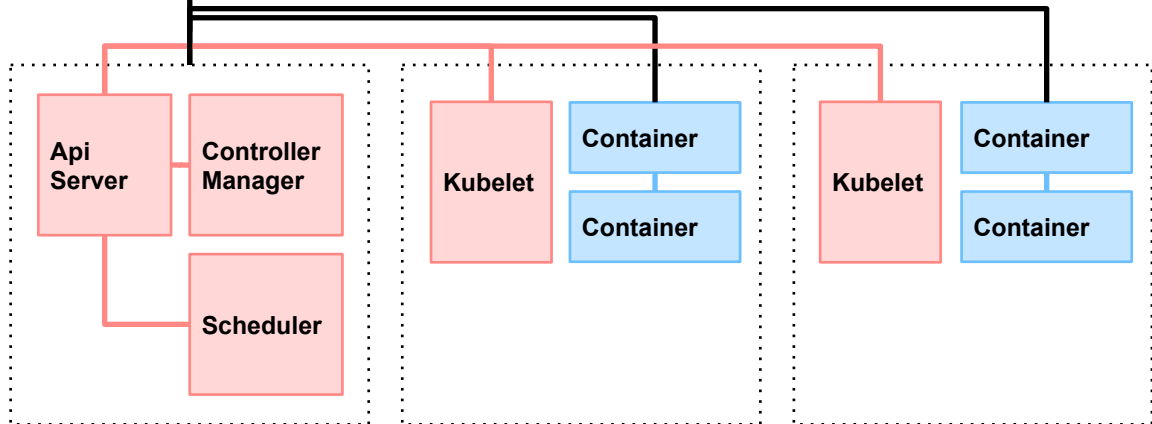
Speaker Notes

- Basically Everyone

Further Reading

- <https://kubernetes.io/case-studies/>

Architecture Diagram



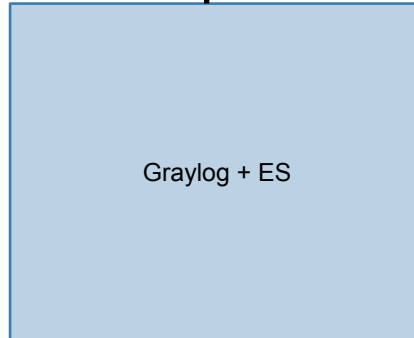
Speaker Notes

- The blue bits are containers, usually docker
- The pink bits are Kubernetes

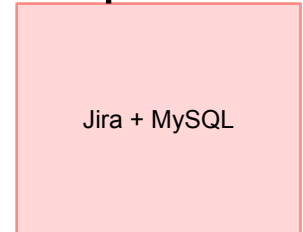
Machine "Pikachu"



Machine "Charmander"



Machine "Bulbasaur"



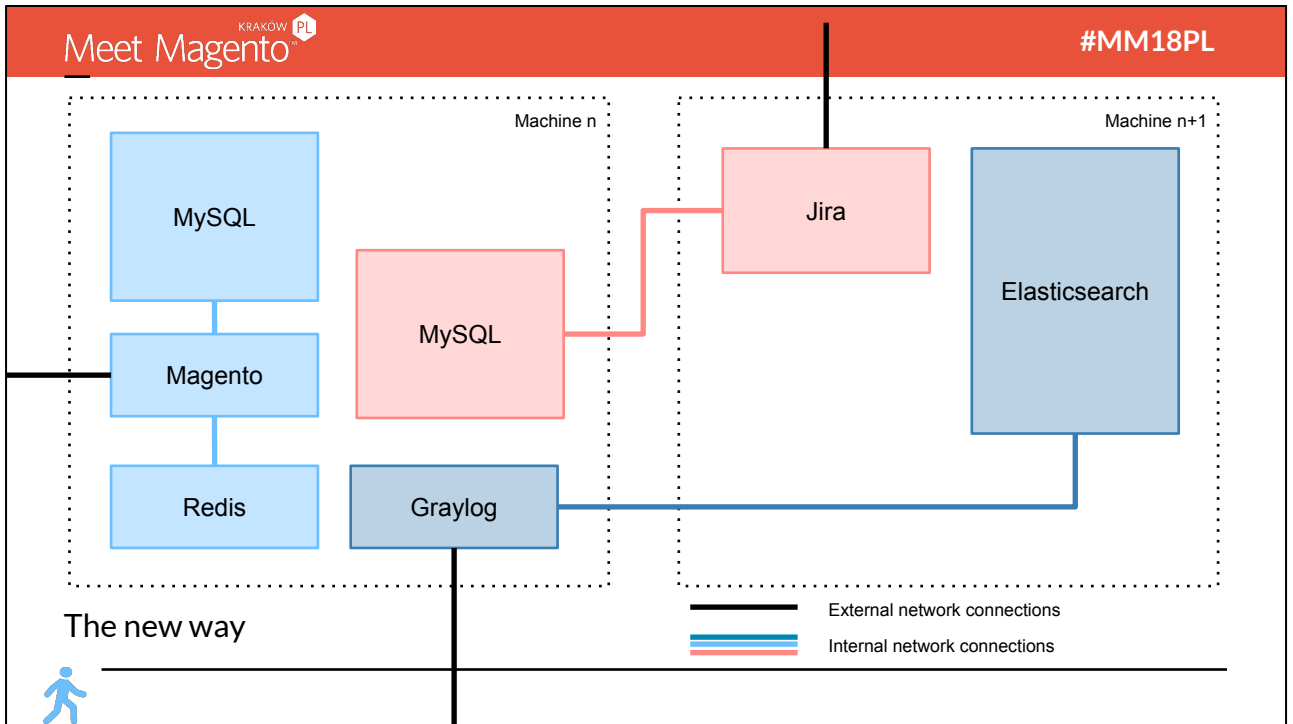
The old way

External network connections



Speaker Notes

- Basically we have no really good way of being accurate about how much resources each application requires
- We usually just bundle everything onto the one machine and give it enough power until it doesn't break
- Efficient?



Speaker Notes

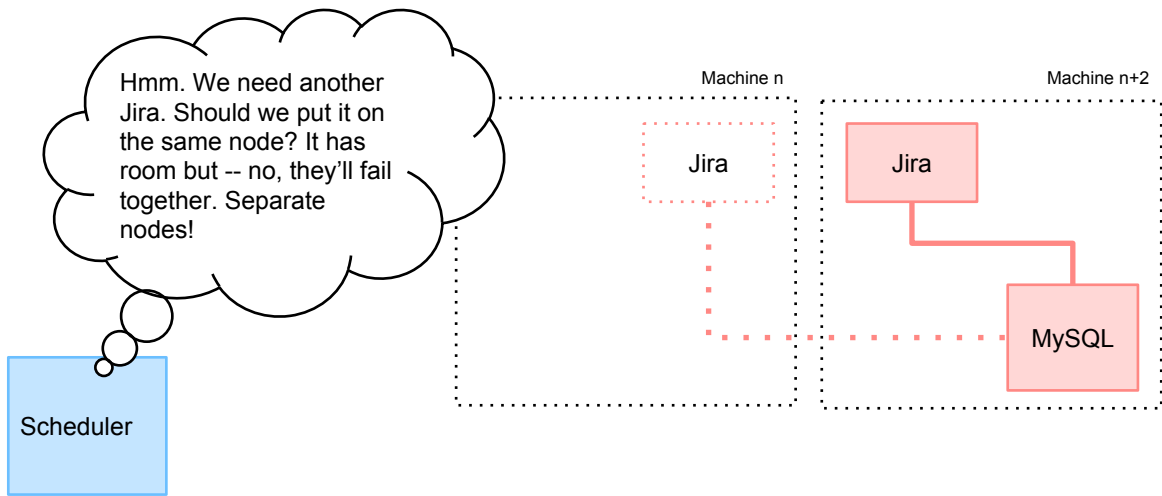
- Kubernetes decouples applications entirely from what host they run on. They run ... somewhere, within an arbitrary set of nodes. Where doesn't matter.
- Each application "reserves" how much it will **usually** need (and we'll come back to why "usually" is important)
- We already get wins here from automation, but this automation is strictly possible with plain EC2 + AWS. It is, however, much harder.
- This also means we can pack staging environments in with prod ones. It doesn't matter -- the isolation is pretty good, and we can make more efficient use of compute this way
- It also means developers can be reasonably self service about what they provision.

Great! But what does this buy us?



Speaker Notes

- Lots of complexity there. What do we get for our complexity purchase?



Software Scheduling

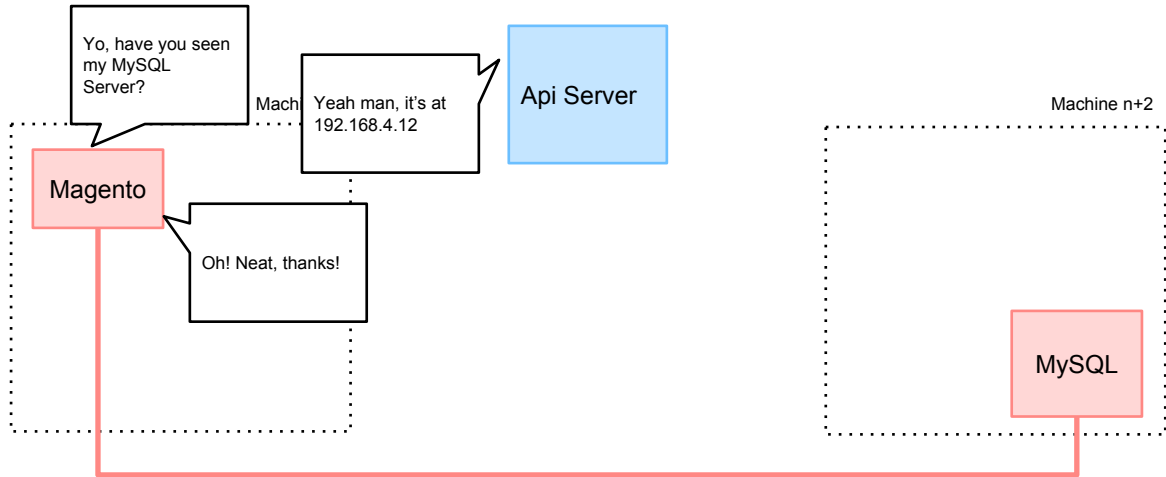


Speaker Notes

- We no longer decide exactly where to place our software
- But something does
- That thing is actually pretty good at making these decisions
- Among other things, it checks:
 - Whether there are disk conflicts on that machine
 - Whether there is enough CPU / Memory available on that machine
 - Whether ports are free on that machine
 - Whether the pod prefers a given machine
- It then ranks the machines based on a bunch of other criteria and picks the best
- If, for some reason, you're not happy with the default scheduler you can write your own

Further Reading

- <https://jvns.ca/blog/2017/07/27/how-does-the-kubernetes-scheduler-work/>

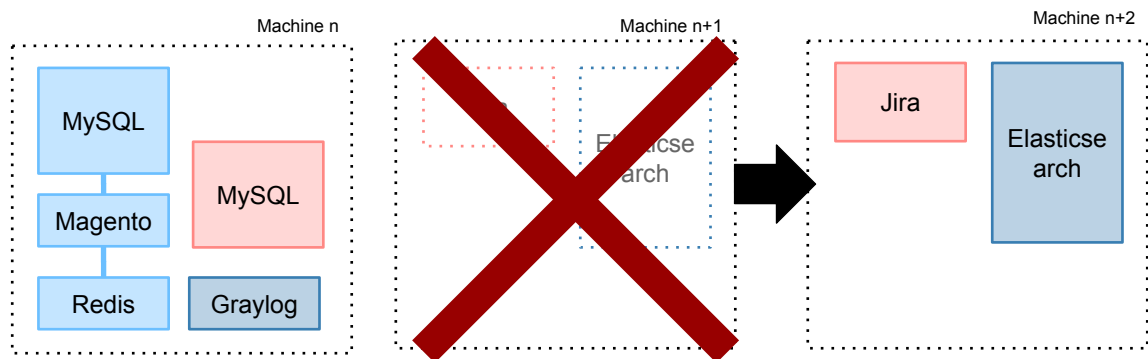


Service Discovery



Speaker Notes

- Kubernetes allows us to create “services”; a kind of load balancer for pods
- Service discovery operates via several mechanisms, but the easiest is simply DNS.



Automated Failure Recovery



Speaker Notes

- Kubernetes works in a control loop (covered later)
- If something breaks, it'll try and fix it

NGINX

NGINX

NGINX



NGINX

NGINX

NGINX

NGINX

NGINX

NGINX

NGINX

NGINX

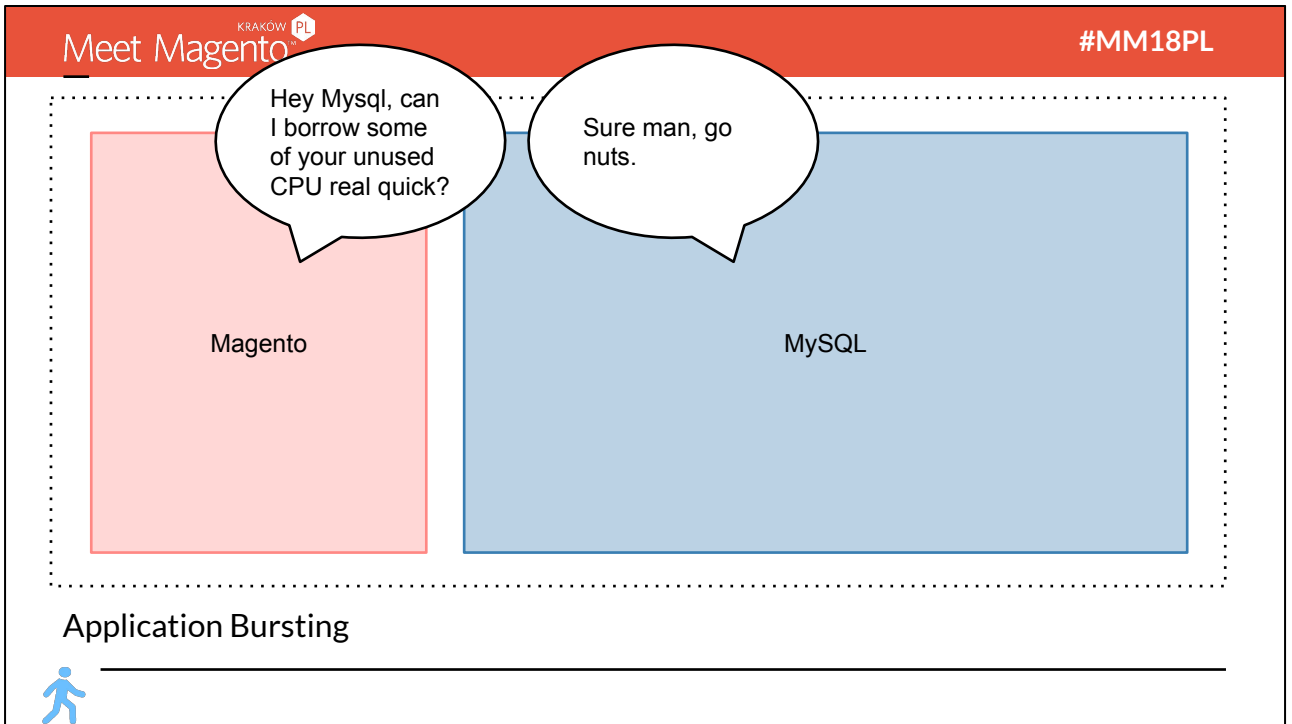
NGINX

Horizontal Scalability



Speaker Notes

- This applies to both stateless services *as well as* nodes
- Additionally, we can autoscale based on arbitrary metrics.

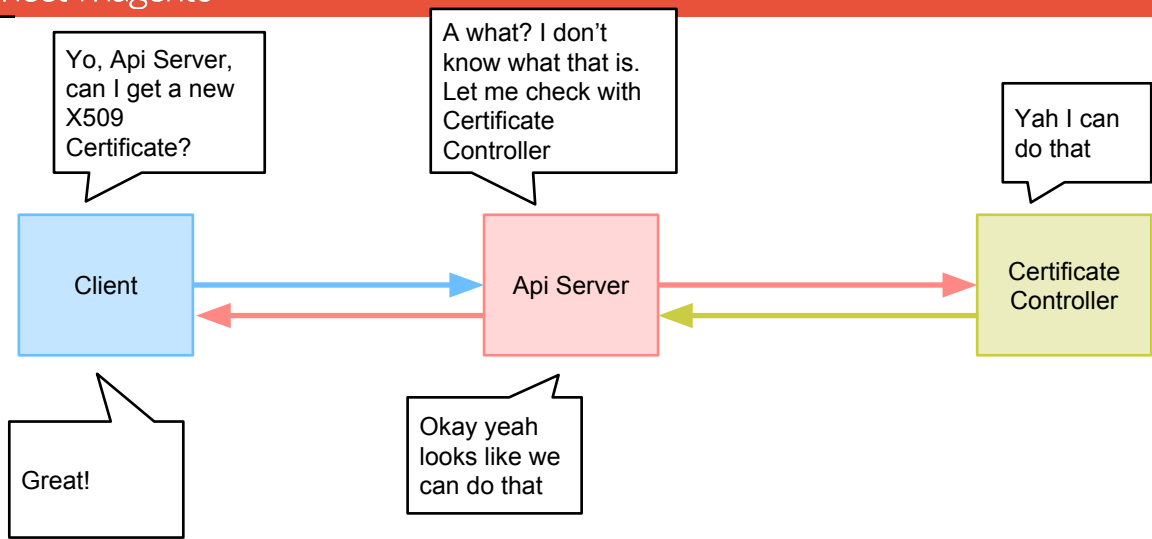


Speaker Notes

- Kubernetes allows applications to burst beyond their existing memory / CPU requirements if those requirements are otherwise unused.
- It will kill applications if memory get too dicey, much like the linux kernel does today.
- It prioritises certain applications according to a QoS policy

Further Reading

- <https://www.ibm.com/developerworks/library/l-completely-fair-scheduler/>
- <https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/>



Extensibility



Speaker Notes

- We can also extend and modify the APIs behaviour, implementing our own feedback loops
- We can additionally extend the root APIs with our own “annotations” to modify their behaviour

+ a bunch more things

Failure Handling · Smart Application Scheduling · Horizontal Scaling · Service Discovery · Application Bursting · Scheduled Jobs · Batch Jobs · Canary Deployments · Rollable Deployments · Health Checking · Managed Configuration · Managed Secrets · Managed Storage · Monitoring · Log Aggregation · Extensible API



<https://kubernetes.io/docs/api-reference/v1/definitions/>

Speaker Notes

- Kubernetes does a hilarious amount of stuff. It will take (literally) years to use it effectively.
- It's also highly extensible. It allows us to build out our own processes on top of it. For example, mark disks as "backup-able".

Ah hm how does this happen?



Controller-manager

Looks at the what the cluster is now, and what it should be, and makes changes so that what it is becomes what it should be.

Scheduler

Decides which workloads should be run on which machine.

Kubelet

Runs and reports on workloads.

Apiserver

Handles communications between the various pieces



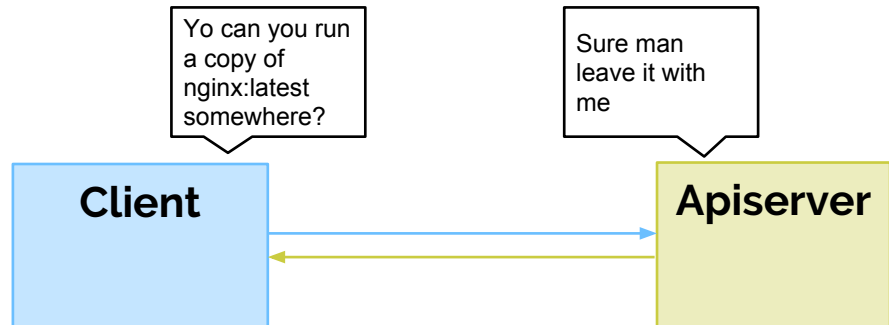
Speaker Notes

- Doing the previous stuff seems pretty magical. But it's backed by a reasonably simple architecture. (explain above)
- Everything is stateless. It's backed by an etcd cluster, which handles state. etcd is a highly available, multi-master key value store with a particularly excellent reputation for distributed consensus.
- There is also the master. It's pretty dumb -- it basically reads and writes to etcd, and it's what other stuff queries to find out the cluster state.

Further Reading

- <https://kubernetes.io/docs/admin/>

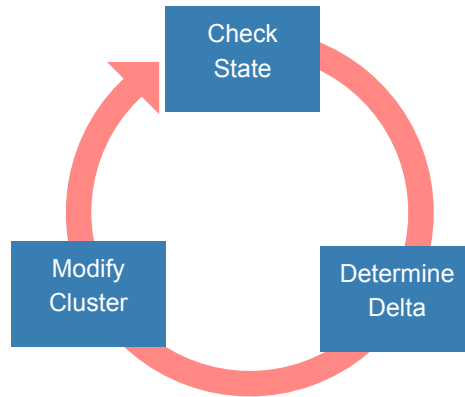
“Please cluster make this happen”



Speaker Notes

- Kubernetes is entirely declarative; you indicate to the cluster “this is what I think should be” and the cluster itself decides how to do make that happen
- This also means the cluster is self-healing; when something goes wrong (such as a machine does) Kubernetes will rearrange itself to be as close to spec as it can manage
- In our case, that probably means everything will just come good as we autoscale worker nodes.
- This lends itself **extremely** well to automation, such as CI/CD

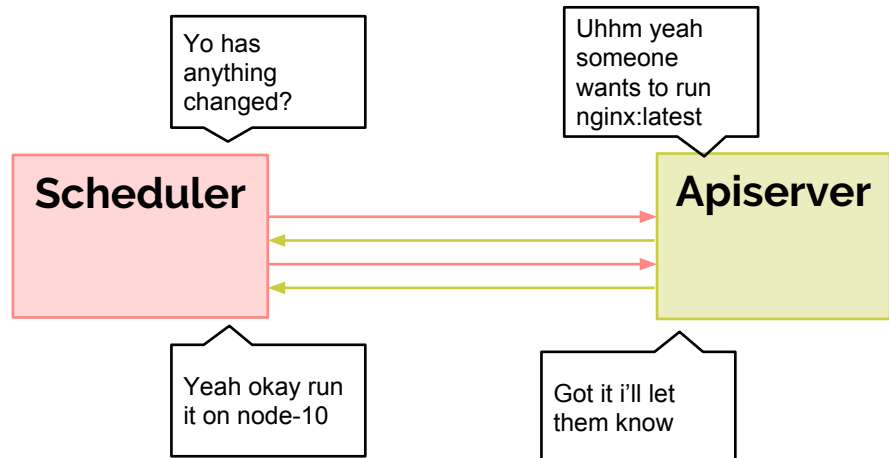
Control Loop



Speaker Notes

- A control loop is kind of like climate control.
 - House gets colder → heat comes on.
 - House gets hotter → cold comes on.
 - House perfect → check again in a few seconds.
- There are lots of control loops in Kubernetes, but they all tend to follow this pattern.

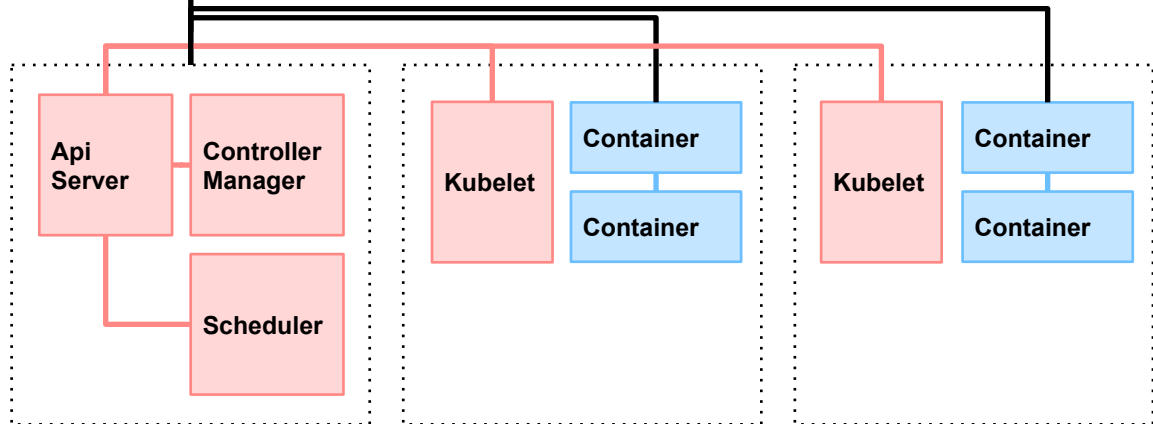
Control Loop



Speaker Notes

- Placing a container is a control loop

Architecture Diagram



Speaker Notes

- The blue bits are containers, usually docker
- The pink bits are Kubernetes

Task Runners

node-10

Kubelet

Apiserver

Yo what
should I be
doing exactly?

Uhh please run
1x nginx:latest

Okay yah sure
1 sec

...

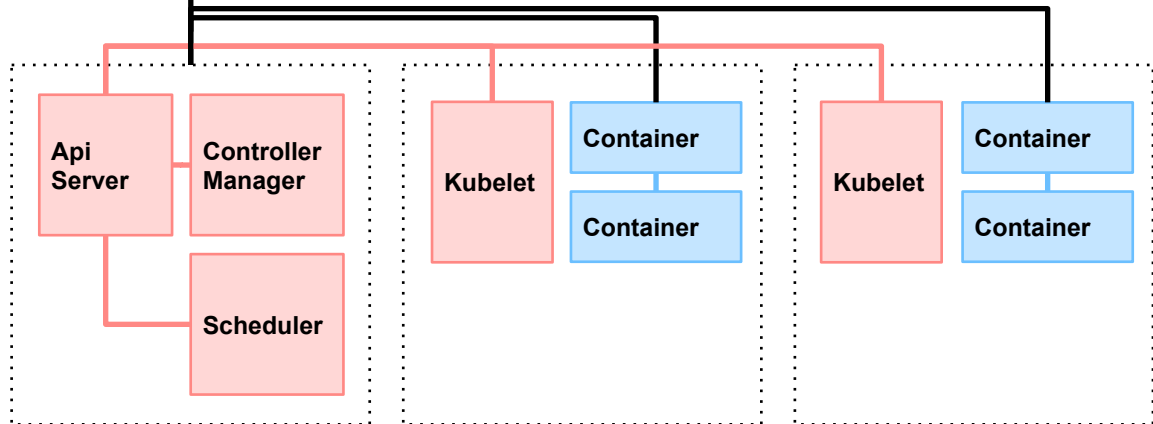
Done.
Anything else?



Speaker Notes

- The task runner is stupid. It just runs tasks. Hooray!
- Reads what it should do from the API
- Also here but not mentioned is the kube-proxy. It's basically the same thing, but for network

Architecture Diagram



Speaker Notes

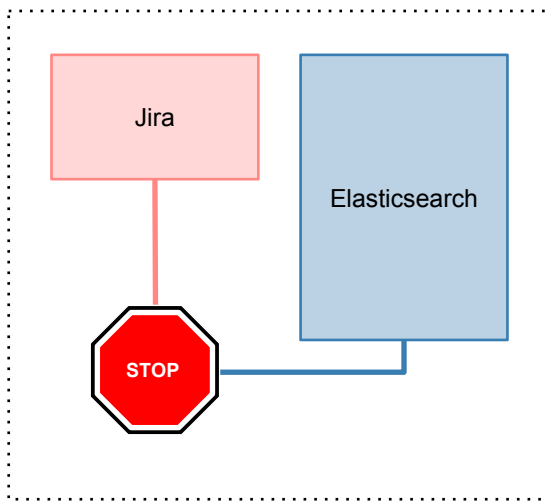
- Now we can see where each bit lives

Security Implications



-- SPEAKER NOTES --

- Yes, yes it was. However there are some ways to mitigate that complexity.



Multitenancy



Applications only see other applications they should*

<https://kubernetes.io/docs/concepts/services-networking/network-policies/>
· <https://blog.jessfraz.com/post/hard-multi-tenancy-in-kubernetes/> · <https://istio.io/>
· <https://spiffe.io/>

-- SPEAKER NOTES --

- The docker isolation is pretty good. But, networks gonna network
- Giant caveats:
 - Network policy needs to be correctly configured
 - There are still some loopholes where applications directly access the Kubernetes API
- See the Istio project, and SPIFFE

Tools for the road to production



Helm

A tool for managing
pre-configured Kubernetes
resources.



Kind of like apt-get or yum for
Kubernetes. It makes the complexity
manageable.

<https://github.com/kubernetes/helm> · <https://github.com/kubernetes/charts> ·
<https://kubernetes.io/docs/tasks/manage-kubernetes-objects/install-repository/>

-- SPEAKER NOTES --

- Just like as currently exists in every major linux distribution, there are people who specialise in packaging software. They expose a limited number of knobs for the consumer to play with.
- For 99% of people in Sitewards, this is Kubernetes.
- There are already quite a number of deployable applications. Checkout the charts repo to see what's available.

GKE

Google Kubernetes Engine

Kubernetes as a service, on top of the already brilliant Google Cloud platform.



Speaker Notes

- GKE will give you \$300.00 and a year to go play with Kubernetes; this is enough to run small workloads for most of that year. It's pretty cheap.
- Be patient. GKE is by far the easiest way of getting started, but it will still take some time to get used to.

Further Reading

- <https://cloud.google.com/kubernetes-engine/>

Profit?



Speaker Notes

- Or, “how to sell this to managers”



Since Squarespace moved to Kubernetes, in conjunction with modernizing its networking stack, deployment time has been reduced by almost 85%. ... Because of that, "productivity time is the big cost saver".



Further Reading

- <https://kubernetes.io/case-studies/squarespace/>



The impact has been considerable: With Kubernetes, the company has experienced a 90% cost savings on Elastic Load Balancer, which is now only used for their public, user-facing services. Their EC2 operating expenses have been decreased by as much as 50%.



Further Reading

- <https://kubernetes.io/case-studies/crowdfire/>



- 20 percent of web tools that account for more than 40 percent of web traffic now run on Kubernetes
- A 25-node cluster that keeps up with each new Kubernetes release
- Thousands of lines of old code have been deleted, thanks to Kubernetes



Further Reading

- <https://kubernetes.io/case-studies/wikimedia/>

In summary

- Containers are good
 - Kubernetes is good
 - They're pretty complex
 - It's worth learning
 - Get a Google Cloud account and start using Kubernetes engine
-



Needs moar info



Jess Frazelle

Keyser Söze of containers



Tim Hockin

Principal SW Engineer,
Kubernetes



Kelsey Hightower

Minimalist



Carter Morgan

Developer Programs
Engineer at Google

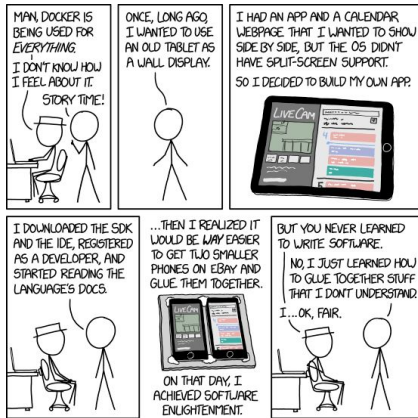
See slides for additional
notes



Further Reading

- Pets versus Cattle: <https://blog.engineyard.com/2014/pets-vs-cattle>
- Udacity Course:
<https://www.udacity.com/course/scalable-microservices-with-kubernetes--ud615> (Free!)
- The Docker Book: <https://www.dockerbook.com/>
- Infrastructure as Code book:
<http://shop.oreilly.com/product/0636920039297.do>
- Site Reliability Engineering book:
<https://landing.google.com/sre/book/index.html> (Free!)

Questions?



<https://xkcd.com/1988/>

Everything is available at the following link:

<https://github.com/andrewhowdencom/talk-using-docker-for-fun-and-profit>

Find the rest of my contact information at

<https://www.andrewhowden.com/>



Speaker Notes

- Warn people that “<joke>Conference speakers are professional enthusiasts!</joke>”. We need a way to follow up as “the rubber hits the road”, or we start putting theory into practice
- Also ask for feedback on this talk.

Thanks



My dev colleagues · Kristoff Ringleff
· Kelsey Hightower · Jessie Frazelle
· Joe Beda · Many others

Further Reading

- <https://kubernetes.io/case-studies/wikimedia/>