# Journey to Containers

Peter Seale
Twitter(𝕏): @pseale
📧 peter@pseale.com
GitHub: pseale

This presentation is a **discussion starter**

So **talk about** your journey!

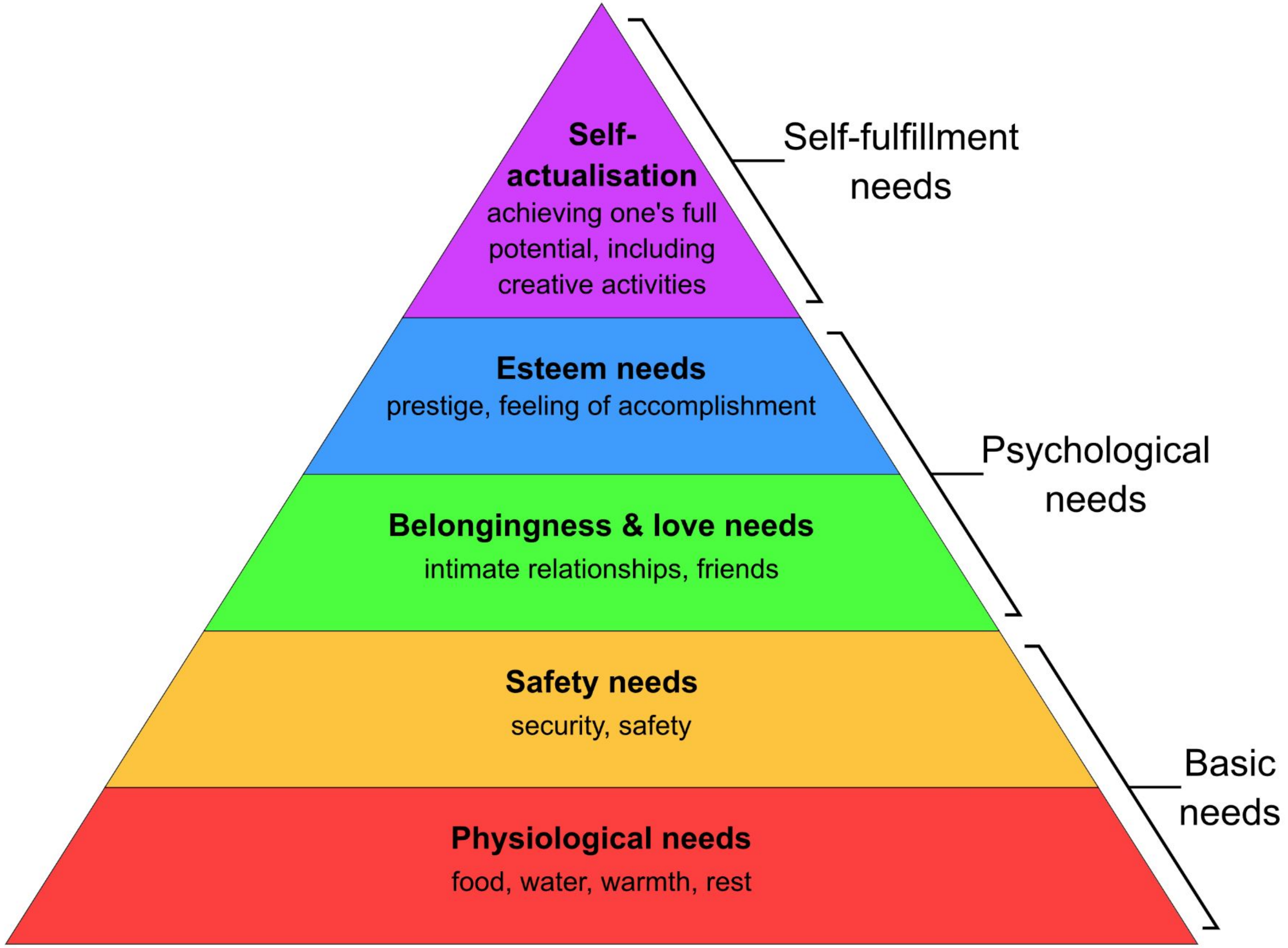
Mort

Elvis

Einstein

$$\begin{tikzcd}\text{Mort} \arrow[d] \\ \text{Elvis}\end{tikzcd}$$

❌ Einstein ❌

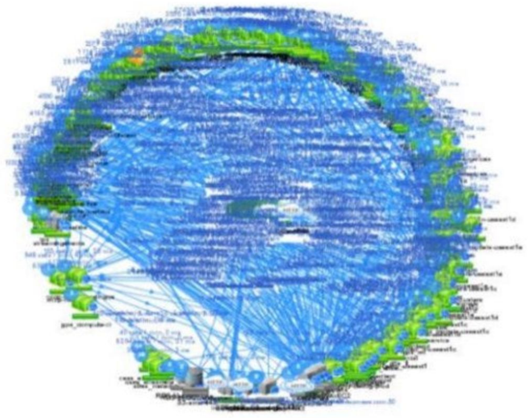

Self-actualisation
achieving one's full potential, including creative activities
Esteem needs
prestige, feeling of accomplishment
Belongingness & love needs
intimate relationships, friends
Safety needs
security, safety
Physiological needs
food, water, warmth, rest
Self-fulfillment needs
Psychological needs
Basic needs

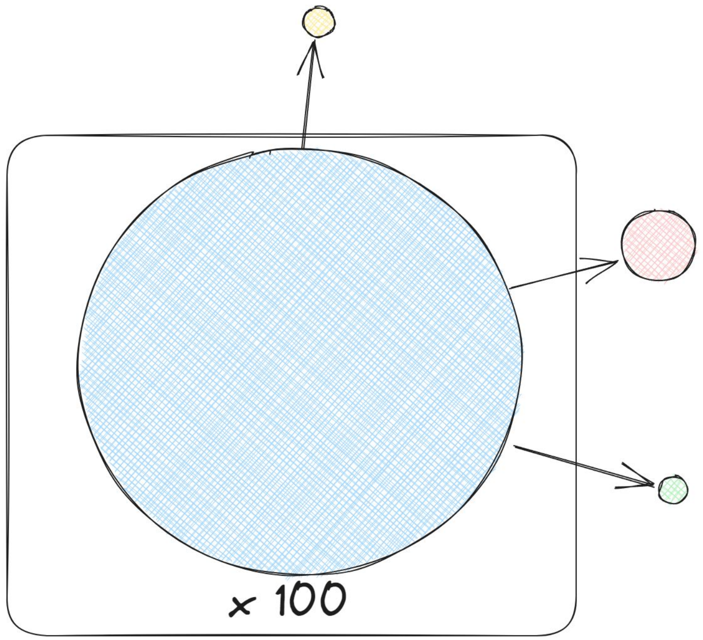
From: https://upload.wikimedia.org/wikipedia/commons/e/ea/Maslow%27s_Hierarchy_of_Needs2.svg

## We are

- small engineering team
- running everything, including the flagship SaaS product
- websites and APIs, a dozen or so autoscaling heavy background jobs - times 100
- multi region - and more regions on the way

# We are not Netflix





## Our story: the beginning

- Pain point: running in Cloud Services and Service Fabric
- Cloud Services: deprecated technology, slow deployments, inefficient Windows VMs
- Service Fabric: an inferior, Windows-based version of Kubernetes - harder to develop against, manage, deploy, scale, had a tiny ecosystem, meanwhile had all the costs and burden maintaining a Kubernetes cluster

## Baby steps

1. docker-compose for local development
2. converted ancillary apps from **.NET Framework to .NET Core**, got feedback and experience
3. Kubernetes: set up a simple AI workload on GPU nodes
4. converted our .NET Framework monolith, over months, in pieces, to .NET Core

## We finished migrating

- Migrated queue consumers slowly - strangler pattern
- The last thing to move: flagship (most complex) website

We're done! Now to live in Kubernetes, for better or worse

## Aside: Upgrading from .NET Framework to dotnet 7 is **difficult, but doable**

- nuget: packages.config to <PackageReference />
- SDK-style project files
- Nuget package inventory and upgrade
- Upgrade shared libraries to netstandard
- Convert unit test projects (why not)
- Convert console apps, worker services
- Convert ASP.NET (difficult)

# Immediate improvement

- major cost savings: Windows -> Linux
- major cost savings: leveraging spot VMs (~50-90% discount)

- faster autoscaling - from hours to 5 minutes
- 3x-5x faster deployments, and deployments are more reliable

## More improvements

- Better monitoring and alerts through Prometheus + AlertManager
- Vulnerability scanning - blessing and curse
- Major cost savings, after gaining confidence in scaling metrics: scale-to-zero

## Not all sunshine and rainbows

🚫🚫 we have had to bootstrap all Kubernetes ecosystem knowledge
🚫 yearly deprecation cycle (Kubernetes-specific)
🚫 early mistakes haunt us still (Persistent Volumes)
🚫 thread starvation is suddenly a problem

## Outages

🚫 new types of outages

- KEDA
- Prometheus
- Certmanager
- Kubernetes APIServer
- linkerd
- “why are there 200 Pending pods” (Azure VM allocation failures (invisible))
- “why has calico restarted 300 times”

## Ongoing tweaks

- container CPU/RAM right-sizing
- Autoscaling - CPU to message-based scaling to complex PromQL-based scaling, to ???

## Future benefits?

- large ecosystem with a bright future
- leverage savings of ARM over x64 (roughly ~20-30% savings right now?)
- ratcheting up the security many ways
    - read-only filesystem, limited user permissions, ephemeral temp storage, CPU and RAM limits (preventing noisy neighbor problems), clamp down the network

## Security vulnerability scanners: our new dystopian Big Brother

- Reduce attack surface - dotnet/runtime-deps instead of ubuntu
- Microsoft-maintained base images are always improving
- ChainGuard base images
- 🚫 Sweep it under the rug: compile self-contained binary

In retrospect: worth it

## Your journey?

## Takeaways

🚫🚫🚫 Windows containers 🚫🚫🚫

## Takeaways

- **We survived the journey!** Saved a bunch of money, better equipped for the future
- **Migrating from .NET Framework to .NET Core is the bulk of the work**, and after migrating, containerizing is trivial by comparison
- **Dotnet SDK container tooling is good** - use it going forward
- But **avoid "[Visual Studio] Container development tools"**
- **Docker Desktop** may cost you money, but it's the best option for local development. Rancher Desktop is also good, and you can also be successful with Podman Desktop. Mac-only: OrbStack
- **Default to stateless services running in containers** going forward - you will be in the best position to move anywhere in the future
- **In 2023, Azure Container Apps** seem to be the best way to host containerized apps, and Azure App Services is still perfect for Windows/IIS hosting
- **Avoid Kubernetes** unless you need it - maintenance alone will require roughly 1 FTE to maintain
- **Threading might be a new problem**, because containers run in cgroups and linux's Completely Fair Scheduler

## Kubernetes-specific takeaways

- Learning path: raw yaml → Helm → GitOps. Most apps can get by with 5% of Kubernetes' features
- **use k9s or OpenLens** to help navigate your many Kubernetes Resources - vastly better experience over kubectl
- For ecosystem benefits, **deploy your code via Helm charts** - kustomize is inferior but ok
- For Helm charts, **practice YAGNI** - make variables only if it varies
- 🚫 Do not ever imitate bitnami Helm charts, not ever (~600 variables)
- The one-year deprecation cycle is brutal