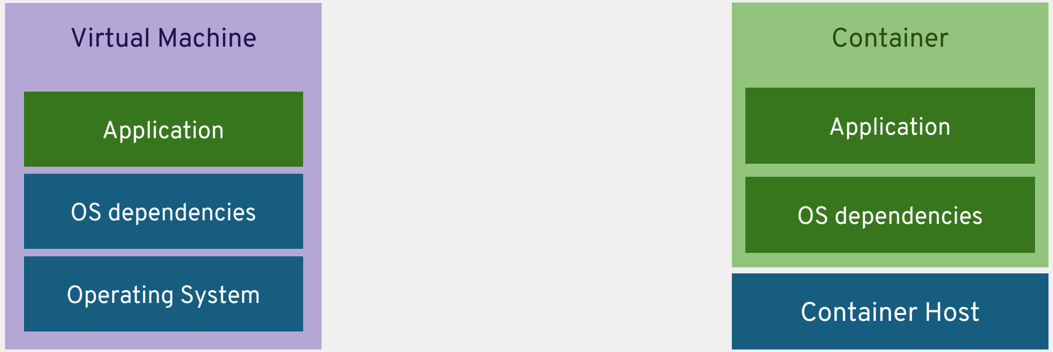
The *X* Steps to Container Excellence

Benjamin Holmes
Solution Architect, Public Sector

Why Should I Bother?

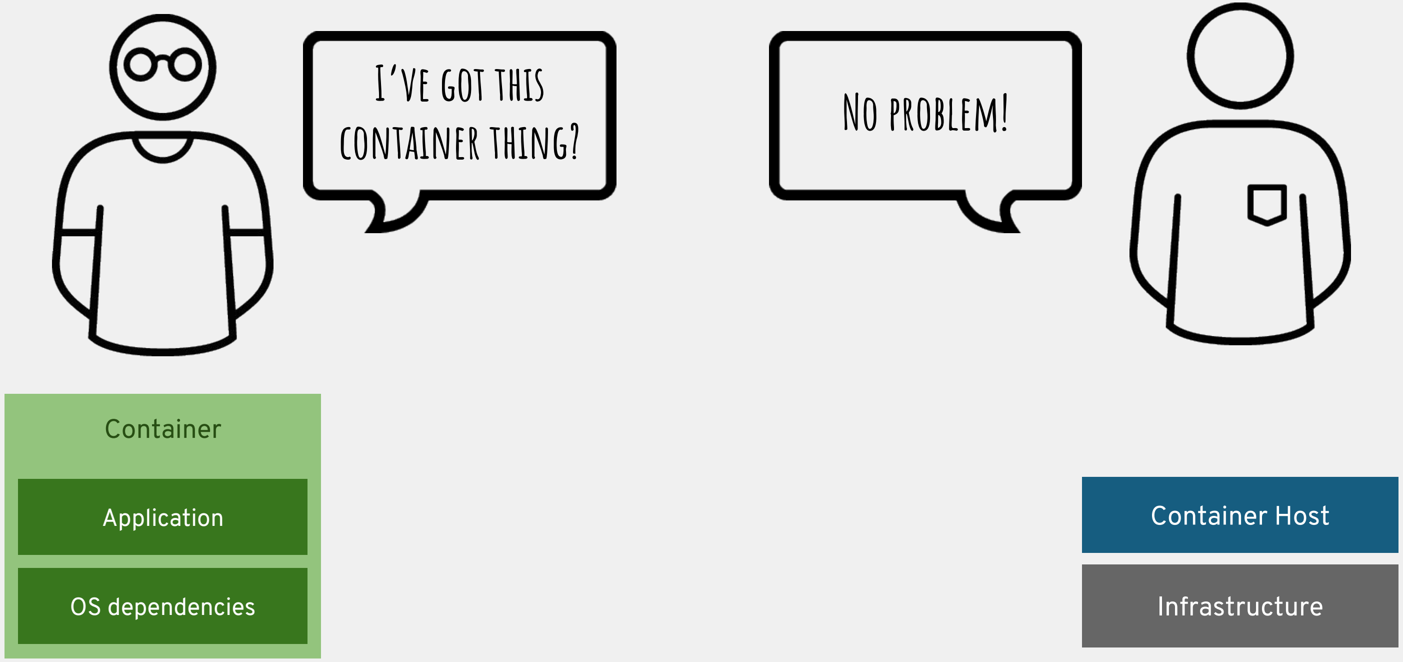
CONTAINERS ARE (STILL) LINUX



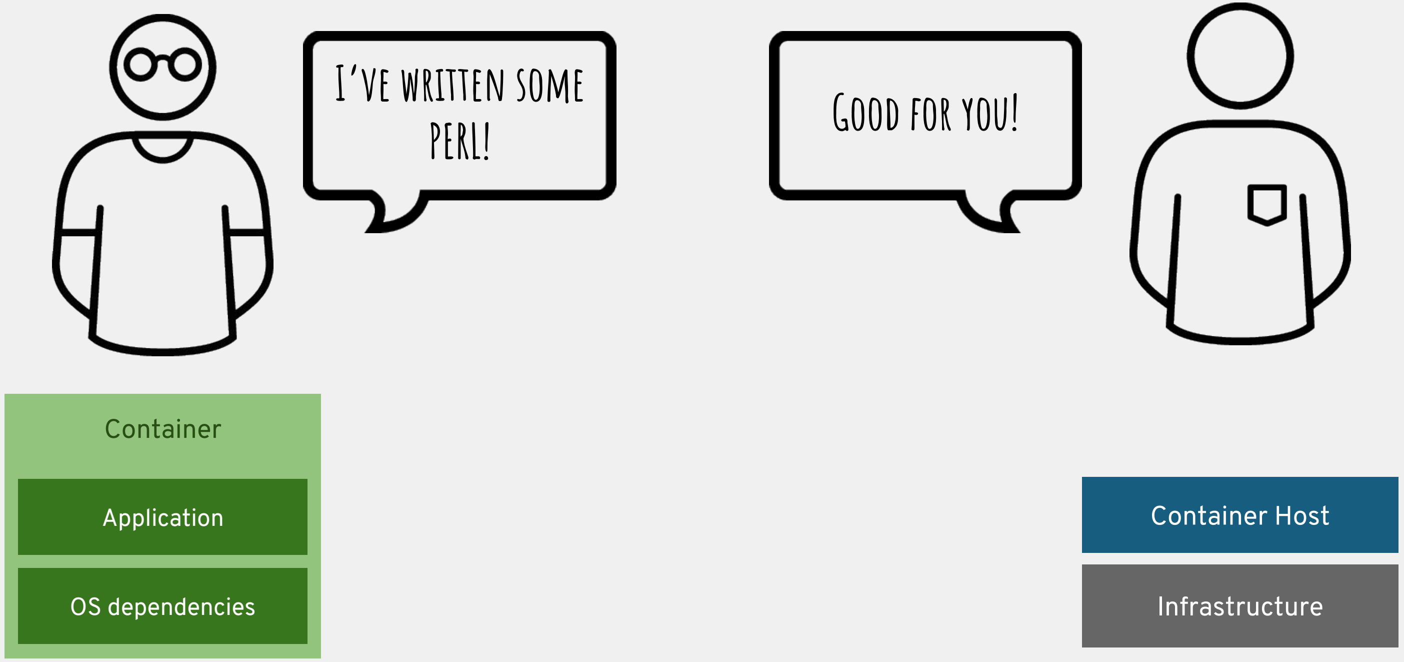
- ➕ VM Isolation
- ➖ Complete OS
- ➖ Static Compute
- ➖ Static Memory
- ➖ High Resource Usage

- ➕ Container Isolation
- ➕ Shared Kernel
- ➕ Burstable Compute
- ➕ Burstable Memory
- ➕ Low Resource Usage

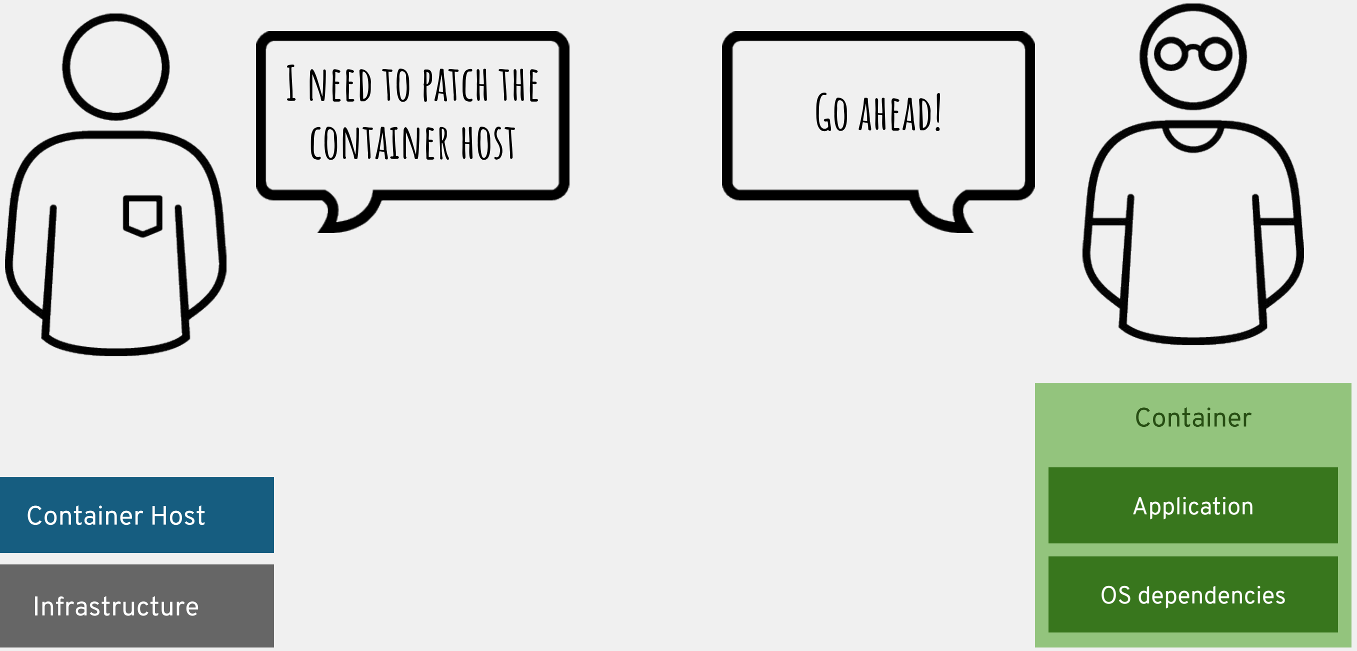
SIMPLER SOFTWARE DELIVERY



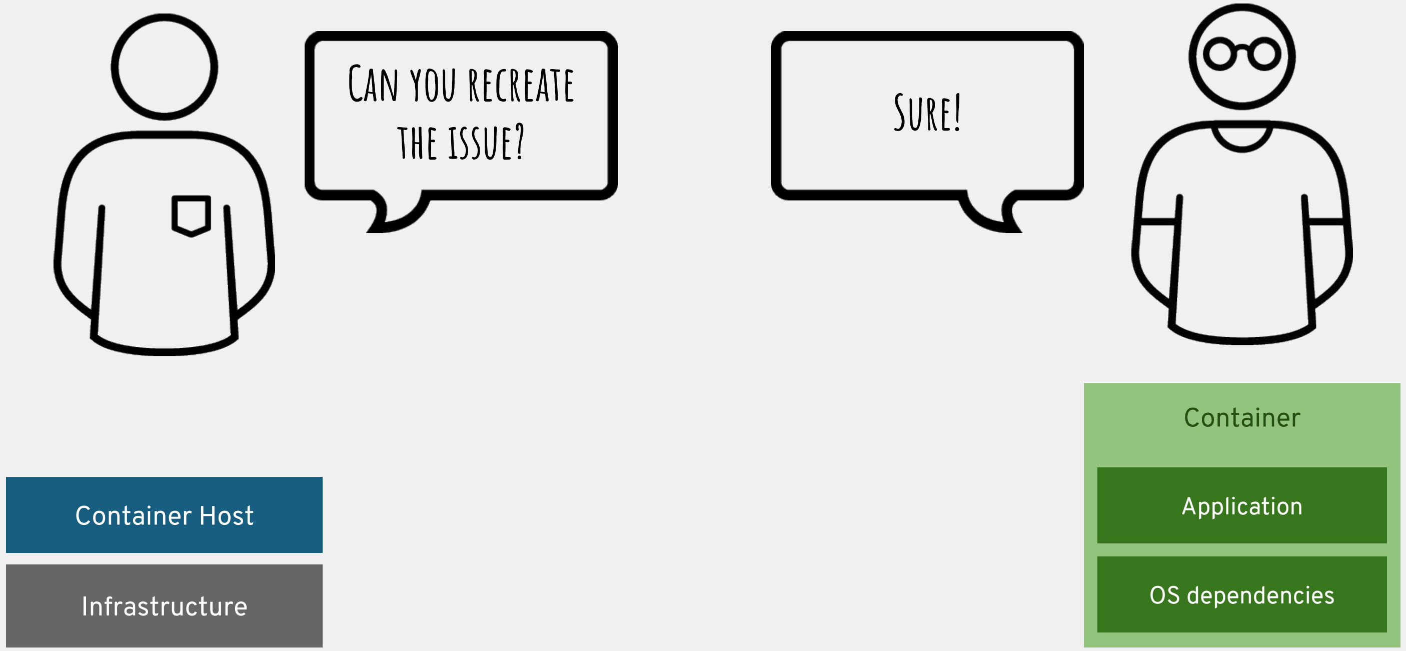
POLYGLOT


I'VE WRITTEN SOME PERL!
GOOD FOR YOU!
Container
Application
OS dependencies
Container Host
Infrastructure

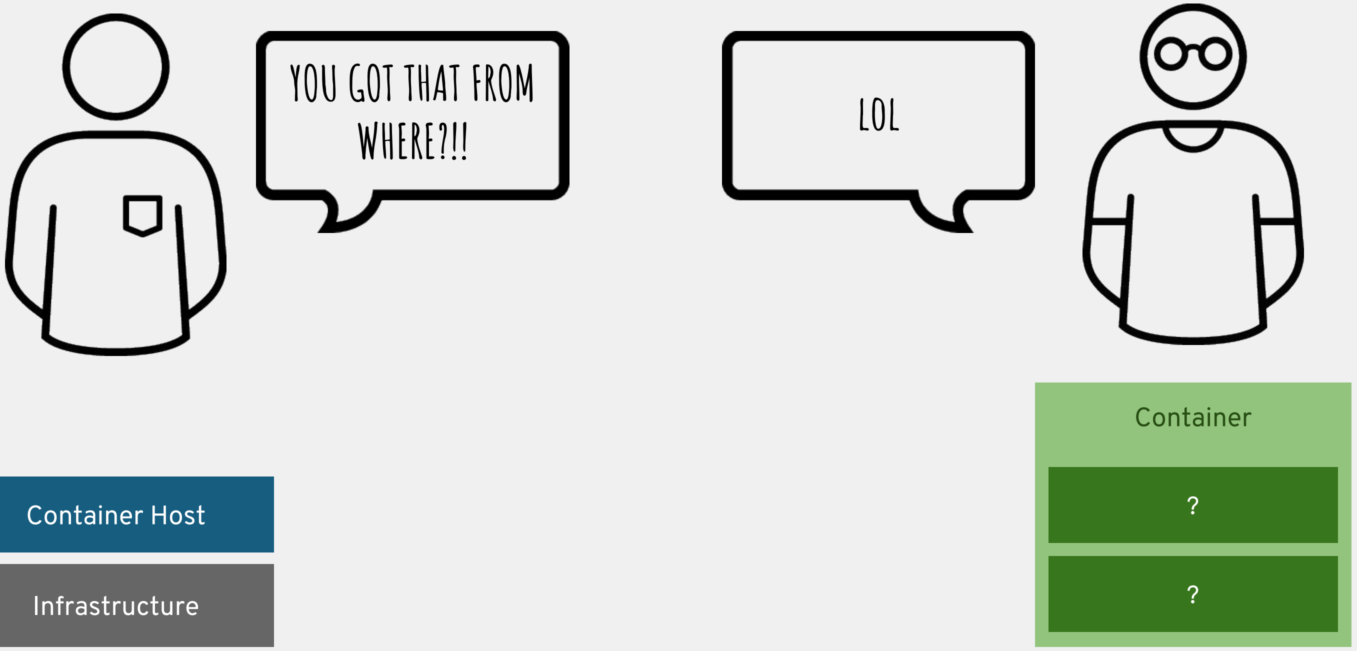
ENCAPSULATION



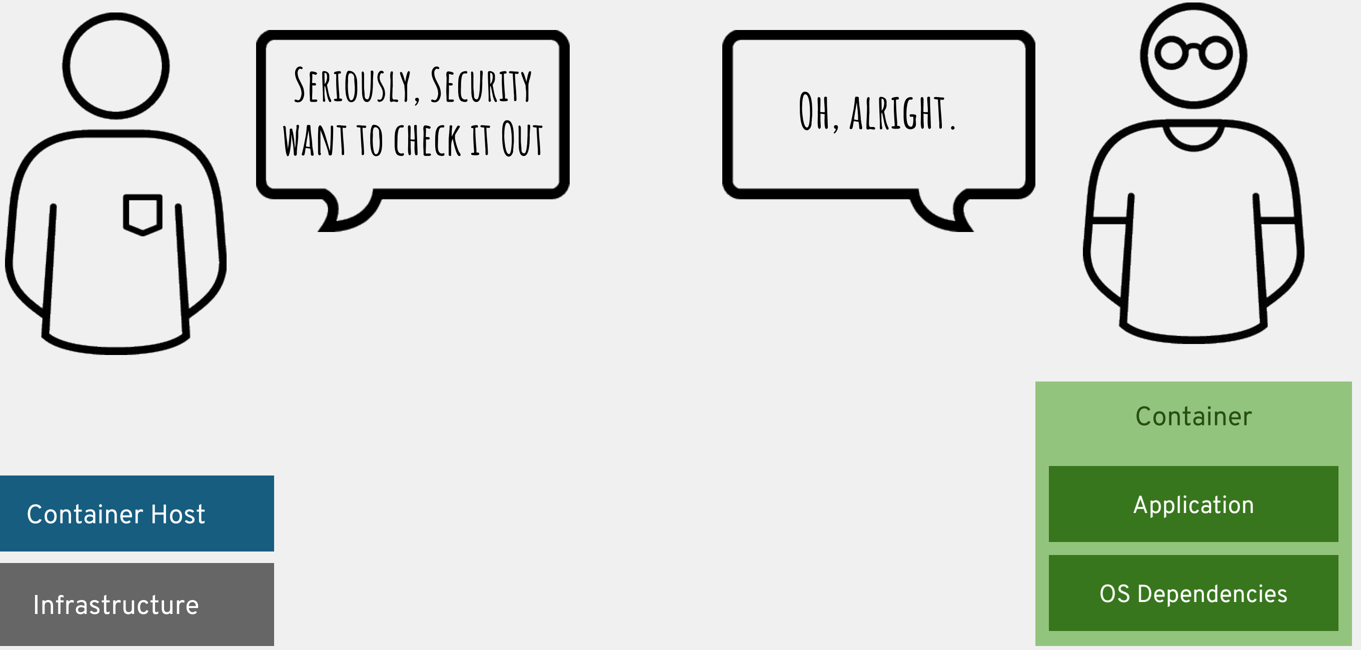
REPEATABILITY



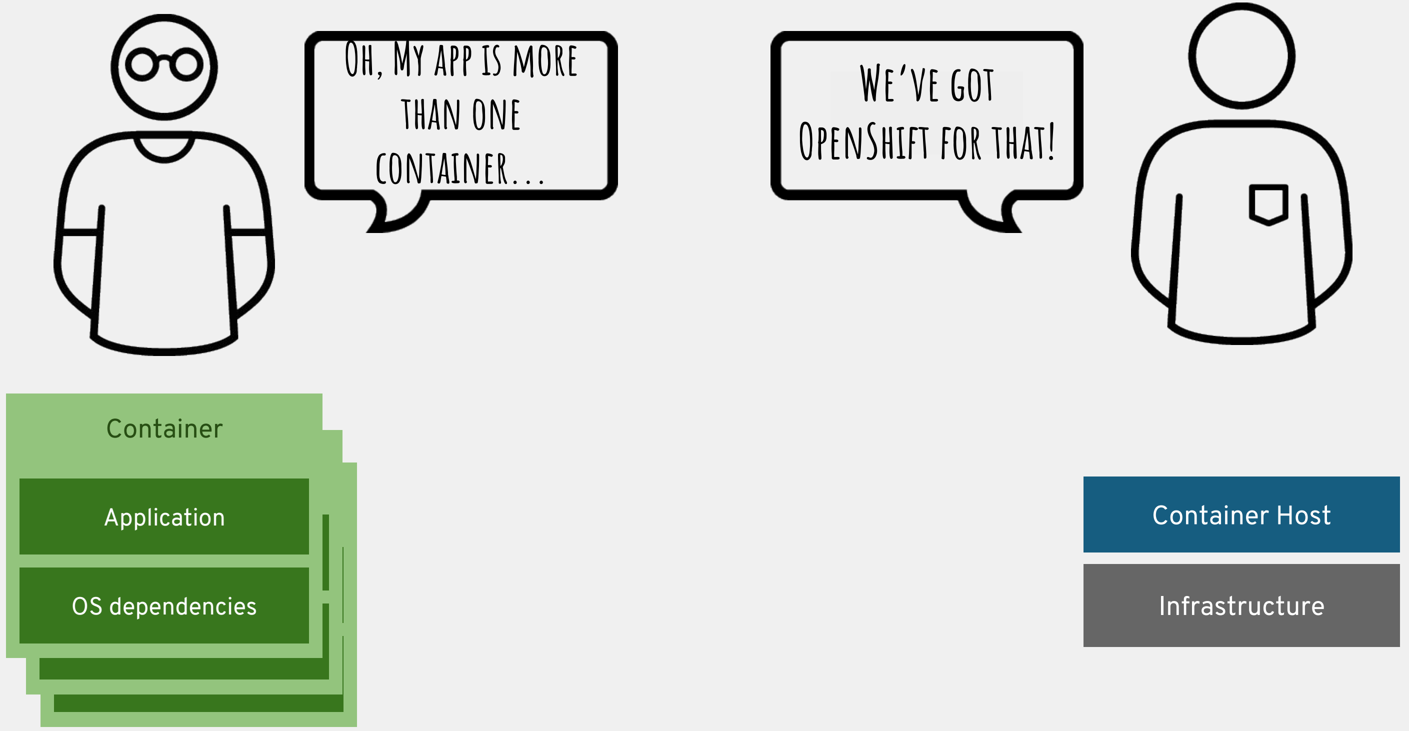
PROVENANCE


YOU GOT THAT FROM WHERE?!!
LOL
Container Host
Infrastructure
Container
?
?

PROVENANCE


Seriously, Security want to check it Out
Oh, alright.
Container Host
Infrastructure
Container
Application
OS Dependencies

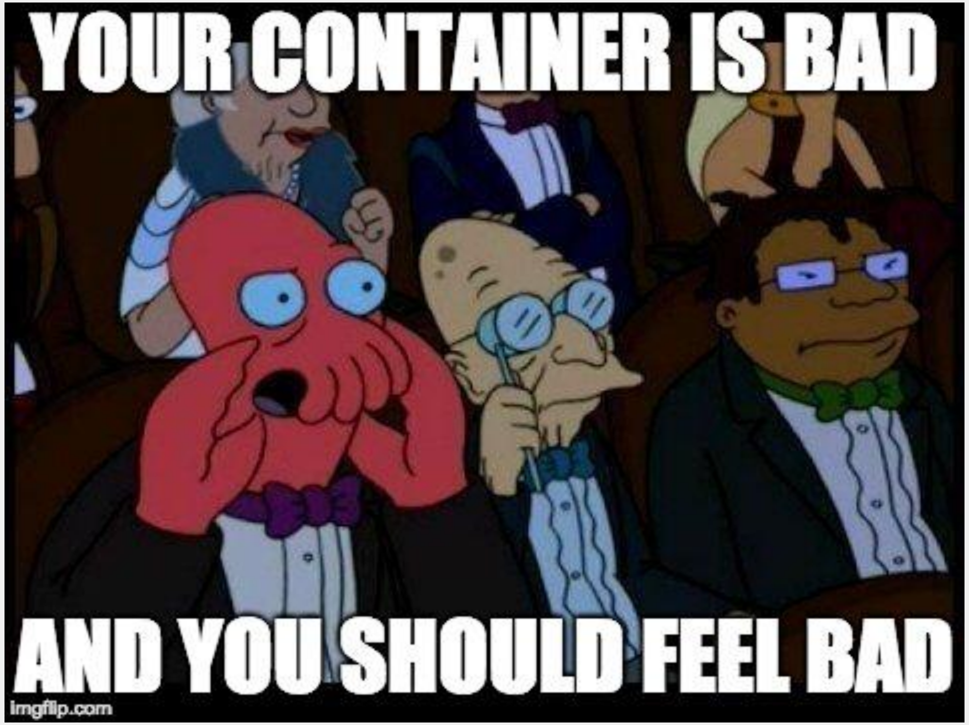
OPENSHIFT CONTAINER PLATFORM


Oh, my app is more than one container...
We've got OpenShift for that!
Container
Application
OS dependencies
Container Host
Infrastructure

Nice. So Will My App Containerise?

MOST THINGS WILL CONTAINERISE, BUT...

- Does it run as root?
- Does it have esoteric networking requirements?
- Does it contain more than one process?
- Does it have dependencies on specific hardware or architectures?
- Does it require specific kernel or host capabilities?
- Does it have licence costs or usage constraints?



https://imgflip.com/memegenerator/You-Should-Feel-Bad-Zoidberg

A GENERAL RULE OF THUMB

COTS	Brown Field	Green Field	Green Field
Stateful workloads	Monoliths	Microservices	FaaS
Potentially Suitable	Requires Modernisation	Good Fit	Good Fit

Sounds Good. Now What?

WHAT IS A CONTAINER NATIVE DEVELOPMENT?

- Uses Container Platform features
- Abstracted from Infrastructure
- Resilient
- Consistent

WHAT SHOULD MY PLATFORM PROVIDE?

OBSERVABILITY

SCALABILITY

FLEXIBILITY

MY PLATFORM IS OPENSHIFT. WHAT'S YOURS?

Self-Service

Multi-language

Automation

Collaboration

Multi-tenant

Standards-based

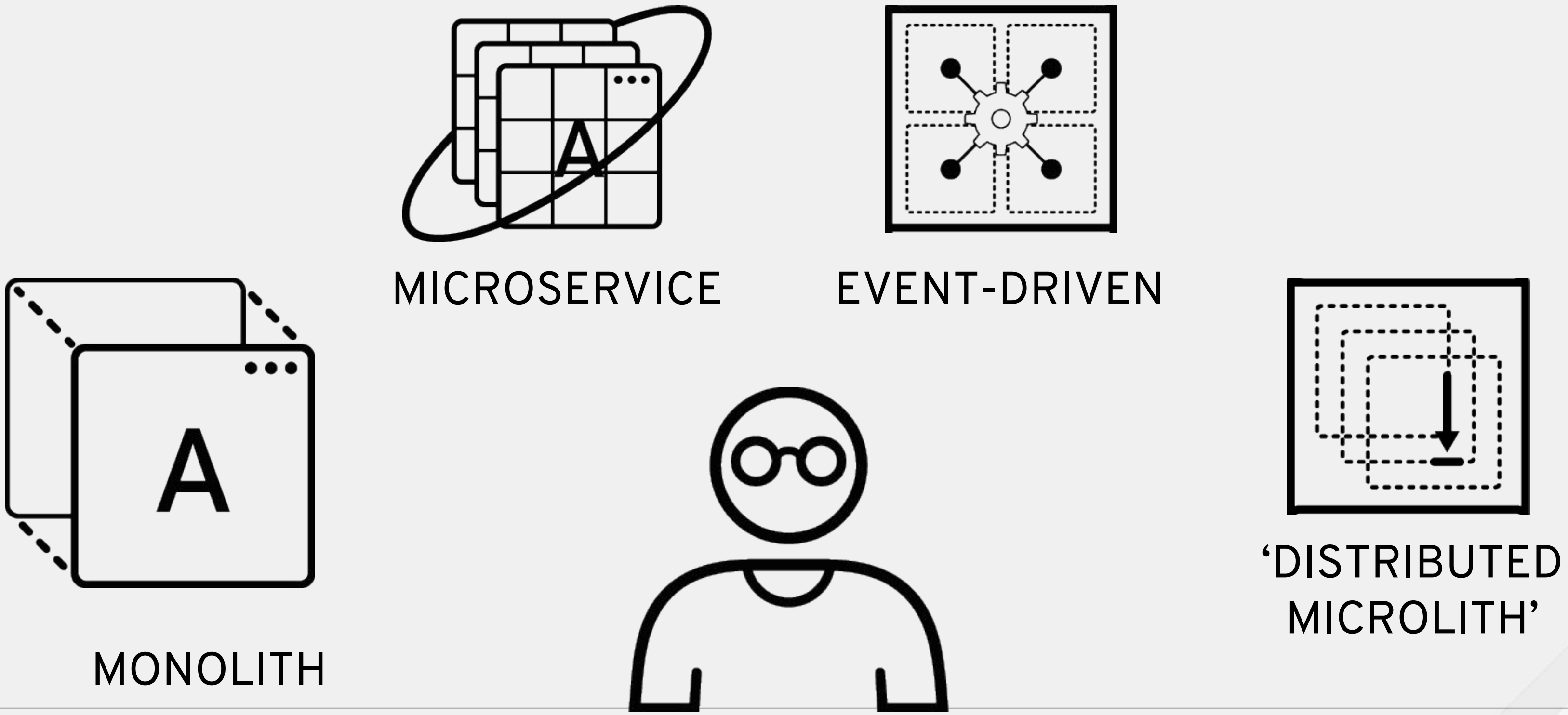
Web-scale

Open Source

Enterprise Grade

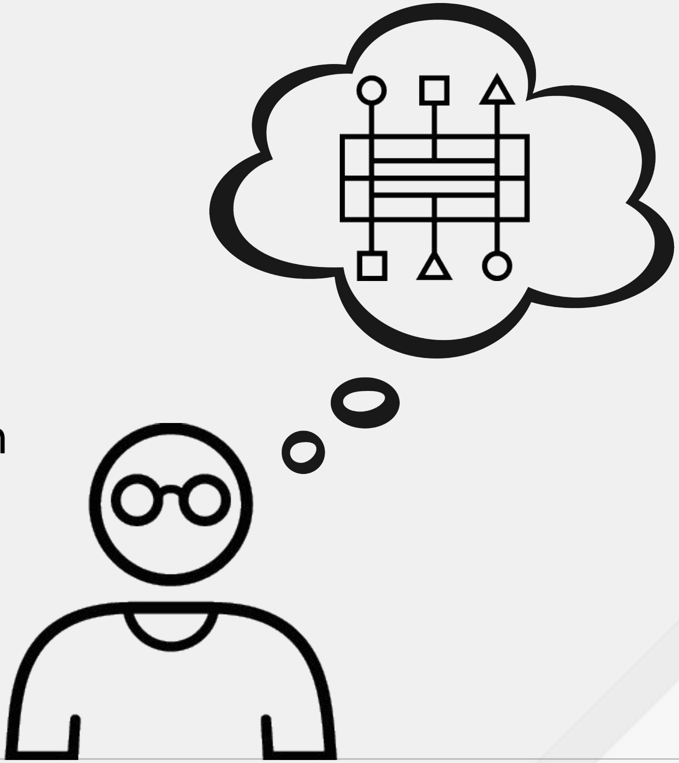
Secure

APPROACHES ARE ARCHITECTURE AGNOSTIC



WHAT ENDPOINTS SHOULD MY APP PROVIDE?

- Health Checks
- Metrics Endpoints
- Thread Dump Generator
- Dynamic Logging Level Switch
- API Contract

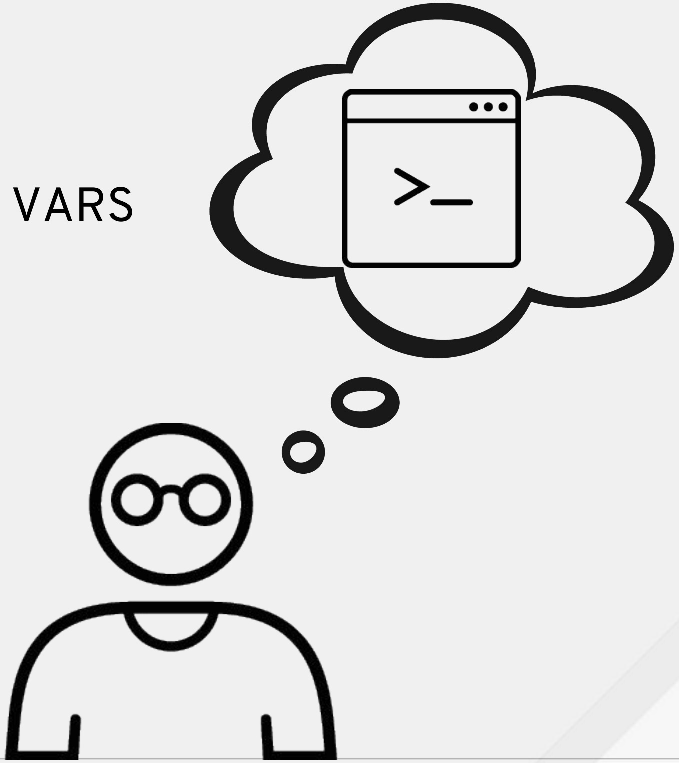


WHAT METRICS SHOULD MY APP PROVIDE?

- Connection Pools
- Last Request Timestamps
- Request / Error / Thread Counts
- Garbage Collection Metrics

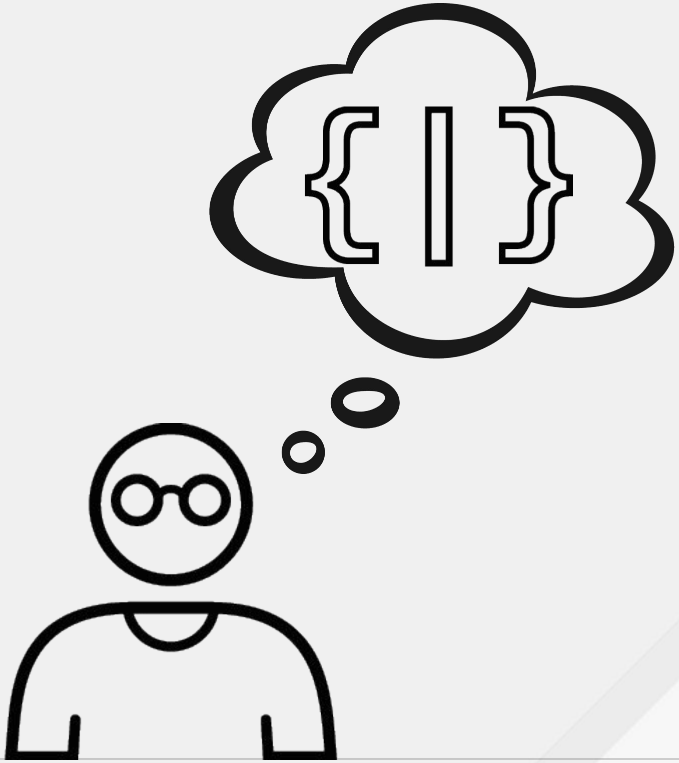
HOW SHOULD MY APP BE CONFIGURED?

- Runtime Flags
- ConfigMaps, Secrets, and ENV VARS
- Service Serving Certificates
- Feature Flags



WHAT SHOULD MY APP BE LOGGING?

- Consistent Formatting
- Correlation IDs
- Default to STDOUT



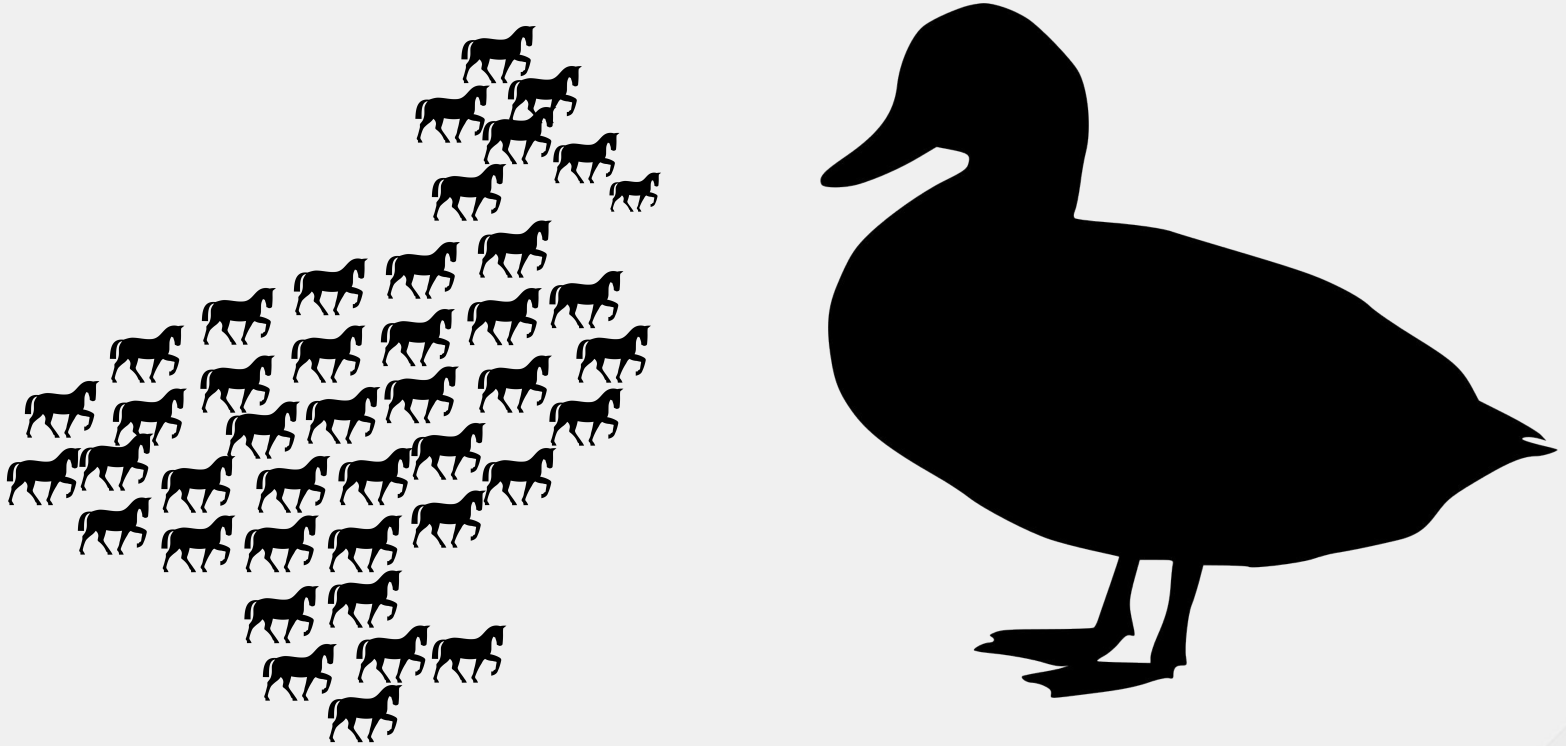
HOW SHOULD I MAKE MY APP MORE RESILIENT?

- N > 1 Replicas
- Lifecycle Hooks
- Wiping the Slate

BUT MY APP IS SPECIAL BECAUSE REASONS...

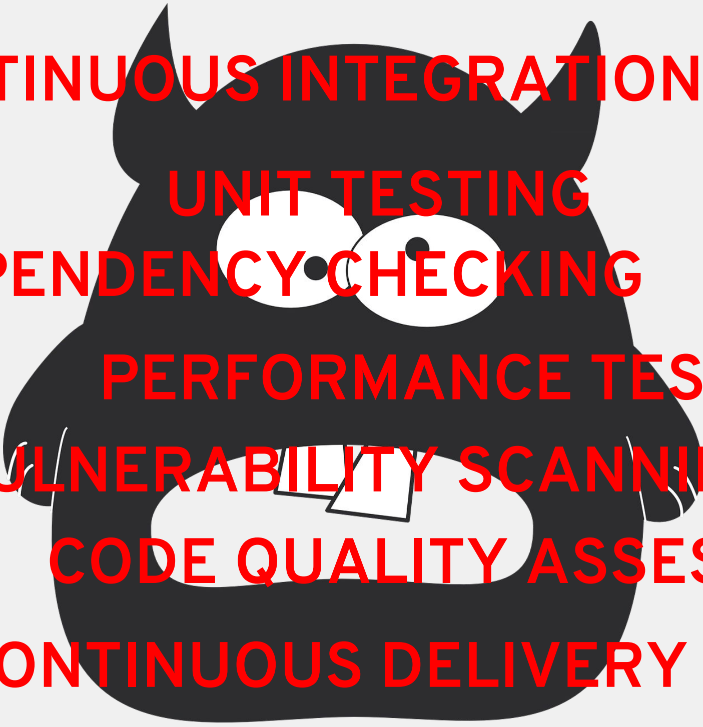
- Common Base Images
- Service Mesh
- Sidecar Containers

SOMETIMES IT CAN FEEL A BIT LIKE...



So How Should I Containerise My App?

THE APPLICATION LIFECYCLE CAN BE A MONSTER…



CONTINUOUS INTEGRATION

UNIT TESTING

DEPENDENCY CHECKING

PERFORMANCE TESTING

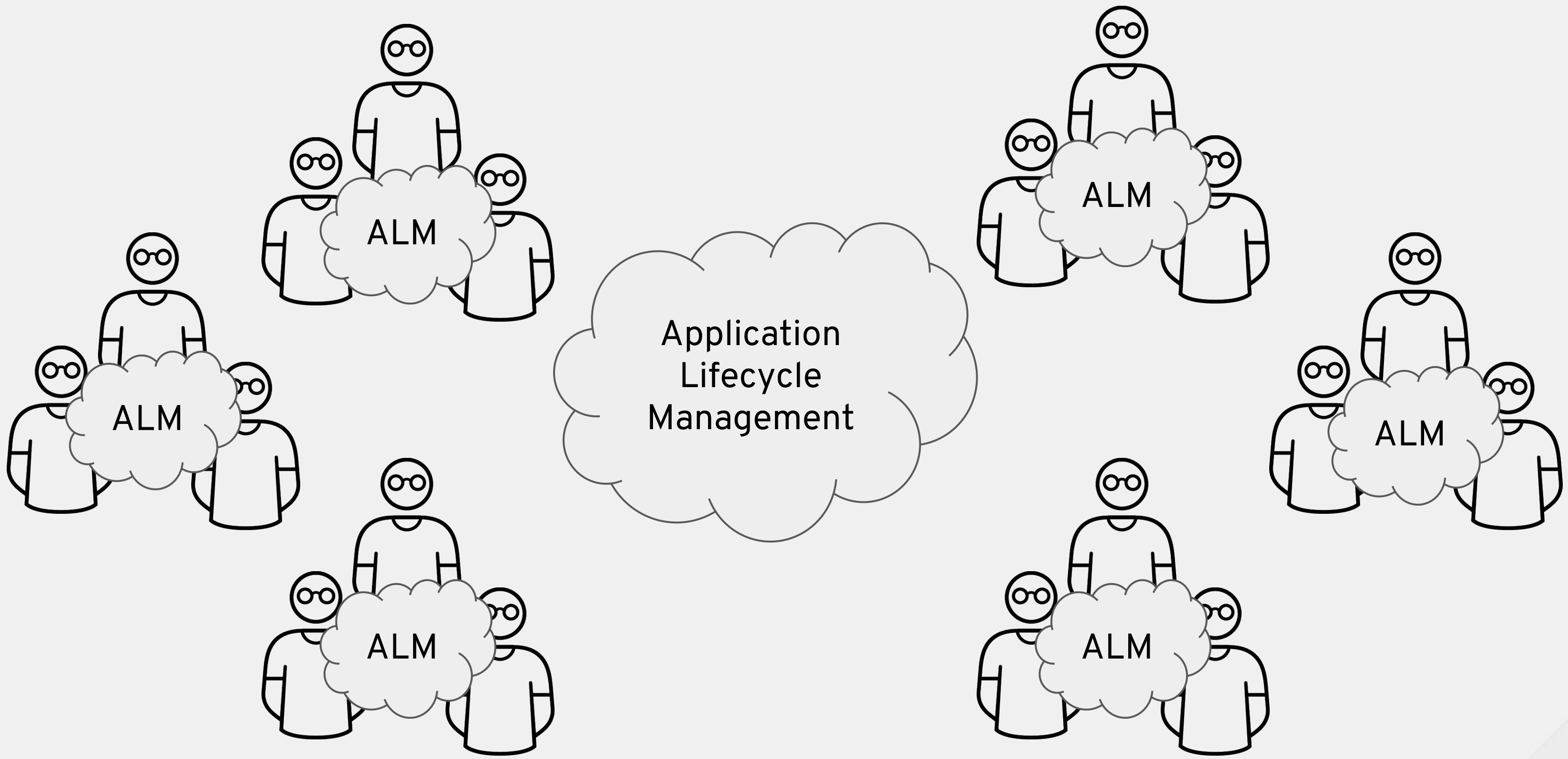
VULNERABILITY SCANNING

CODE QUALITY ASSESSMENTS

CONTINUOUS DELIVERY

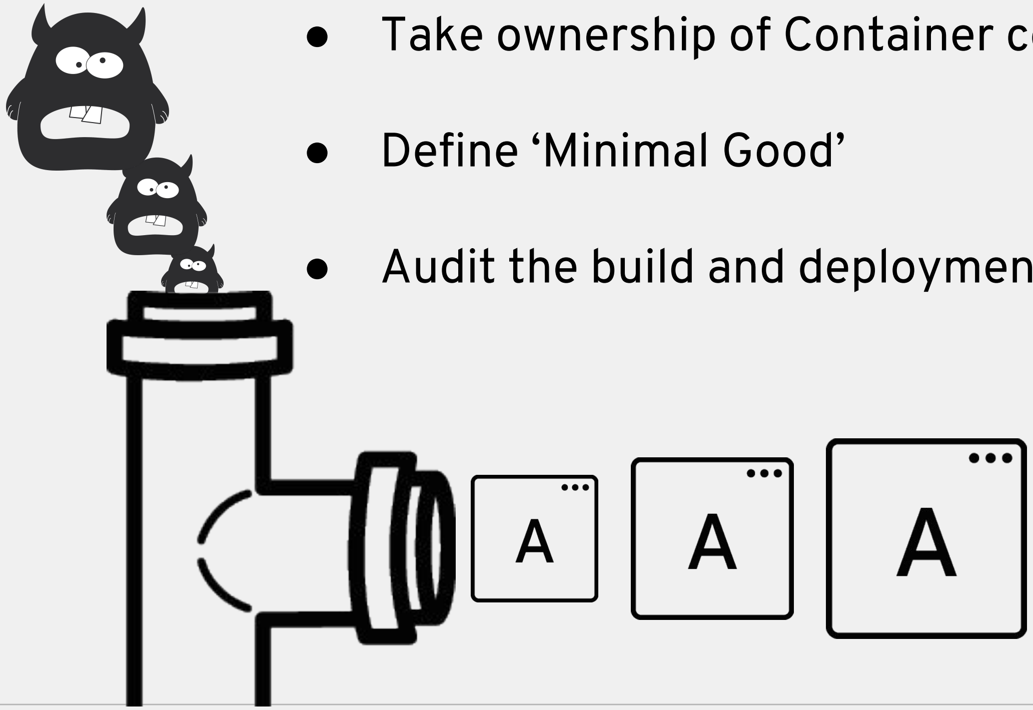
https://pixabay.com/en/monster-nasty-devil-teufelchen-602548/

…SO LET'S EMPOWER DEVS TO FIGHT IT



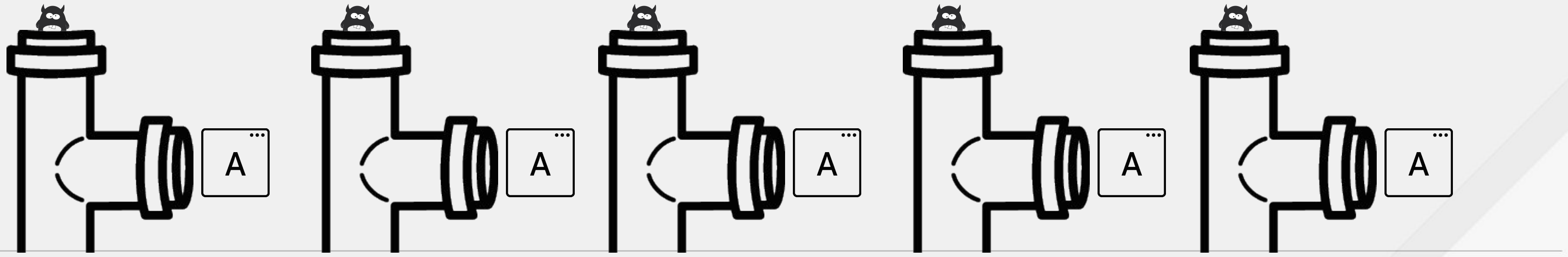
TRUST THE PIPELINE

- Take ownership of Container content
- Define ‘Minimal Good’
- Audit the build and deployment process



INDUSTRIALISE THE PIPELINE

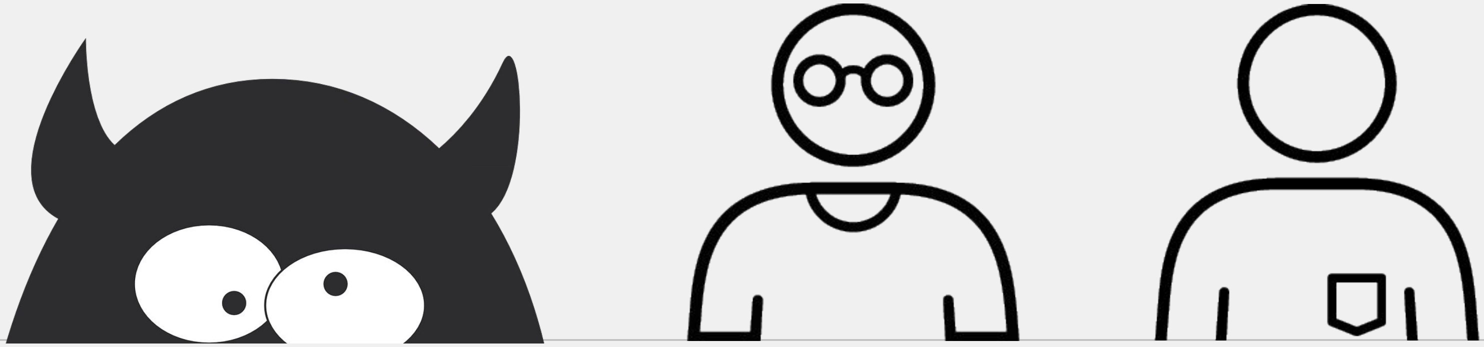
- Path of Least Resistance
- Opinionated, yet Flexible
- Open / Inner Sourced



In Summary

IN SUMMARY

- Better app design - build for the platform, not against it
- Make use the capabilities provided by a Service Mesh
- Learn to love the Pipeline!



Thank You