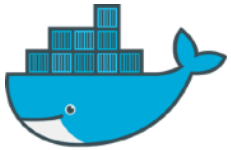
# CS155: Infrastructure Security Building Blocks 🔒

@diogomonica



- Infrastructure Company
- Running on 10% of the internet
- Infra: Go, some python

- Mobile payments company
- Moving $60 billion annually
- Infra: Java & Ruby, some Go

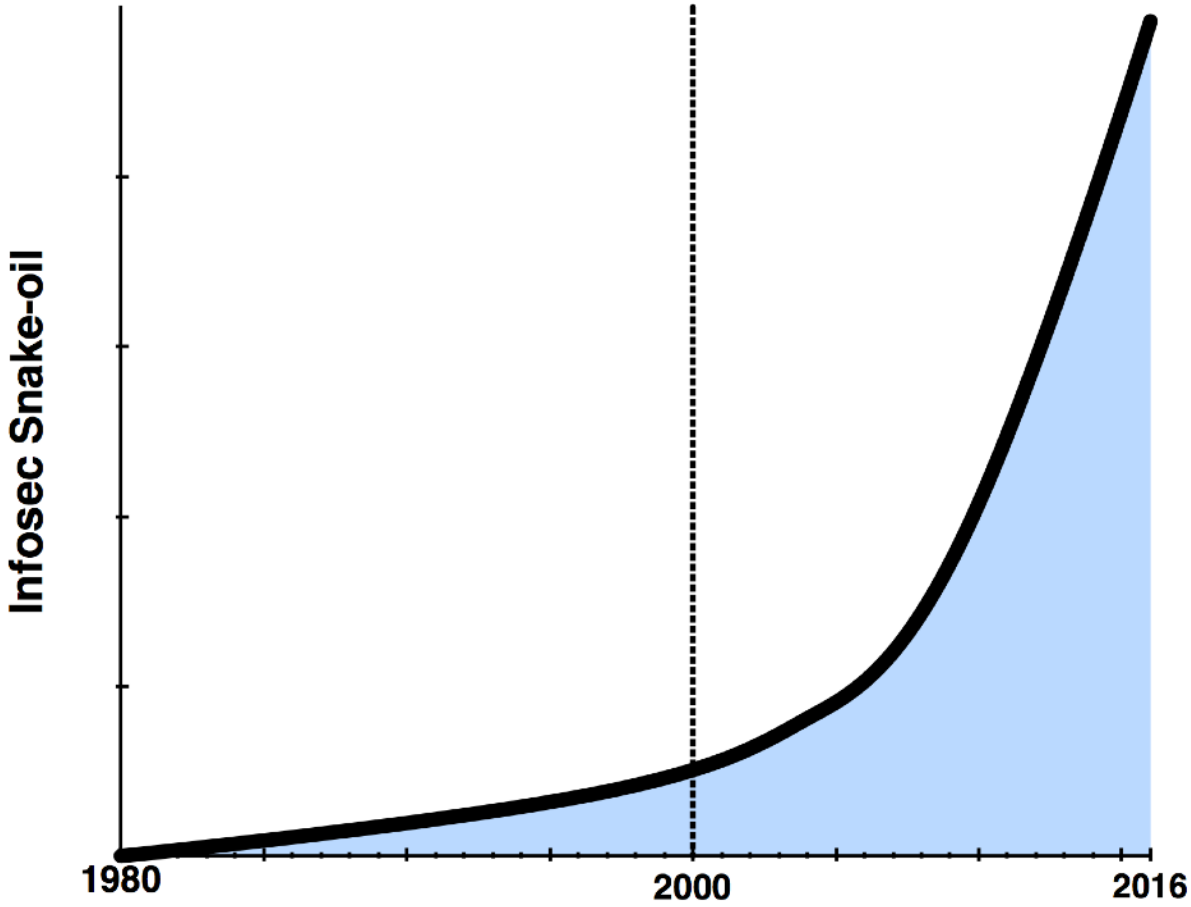
# Agenda

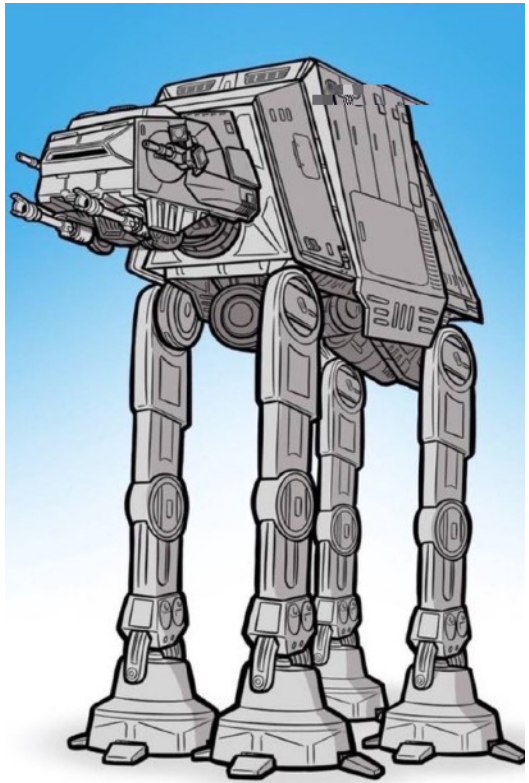
1. Why infrastructure security?
2. Biased timeline of how we got here
3. What is Docker
4. Docker security tetronimos
5. Demo

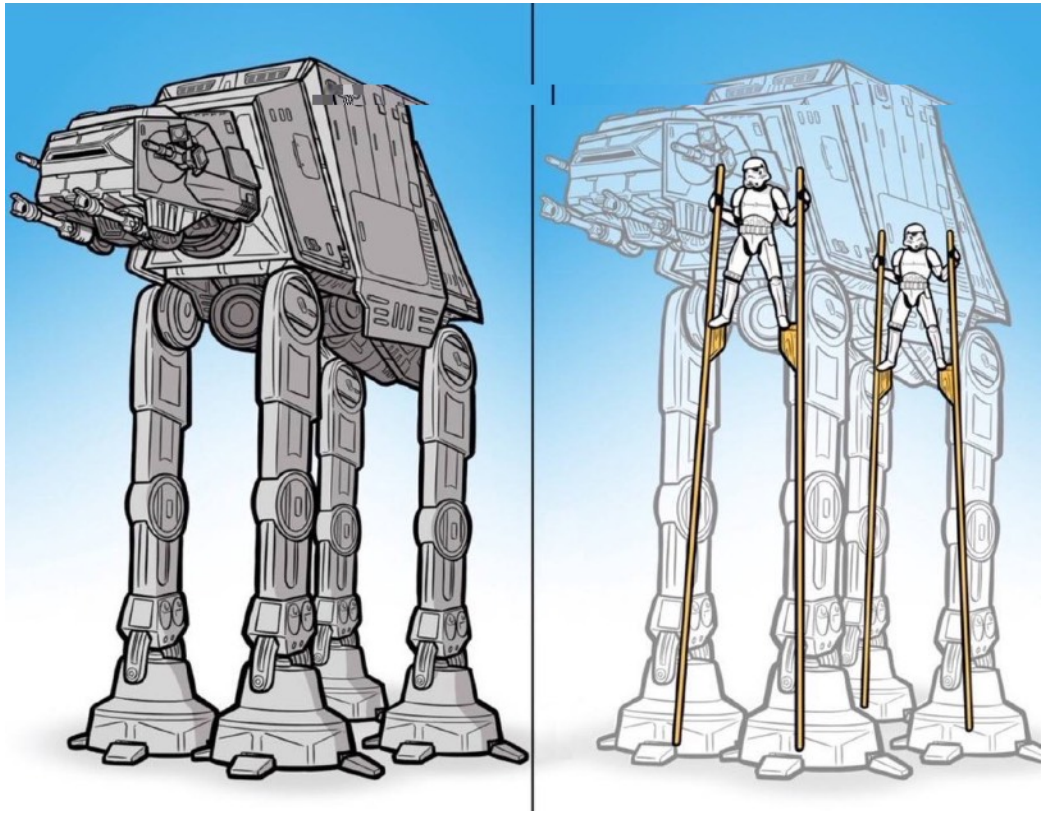
# Why Infrastructure Security?


# The Golden Ages


# The Dark Ages

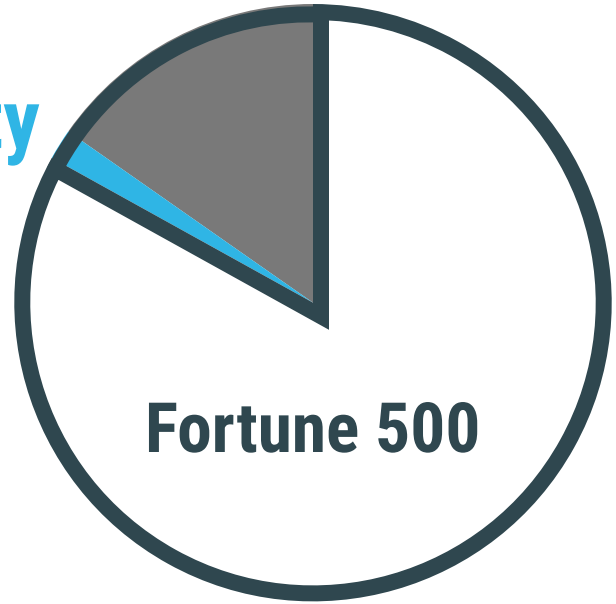

Infosec Snake-oil
1980
2000
2016

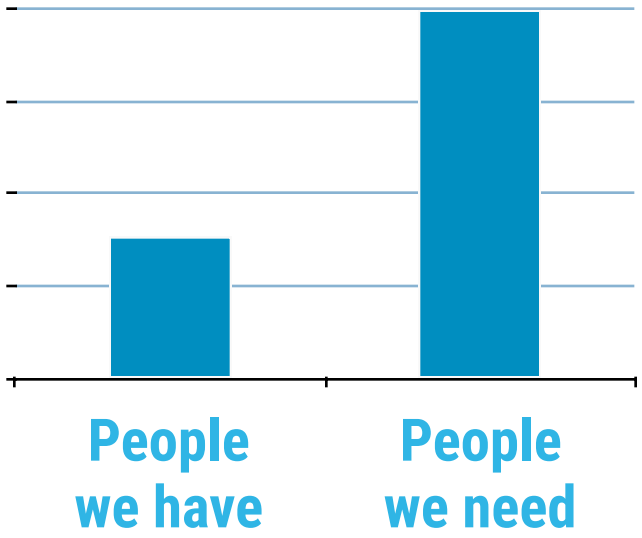




Tech Companies

Good at security

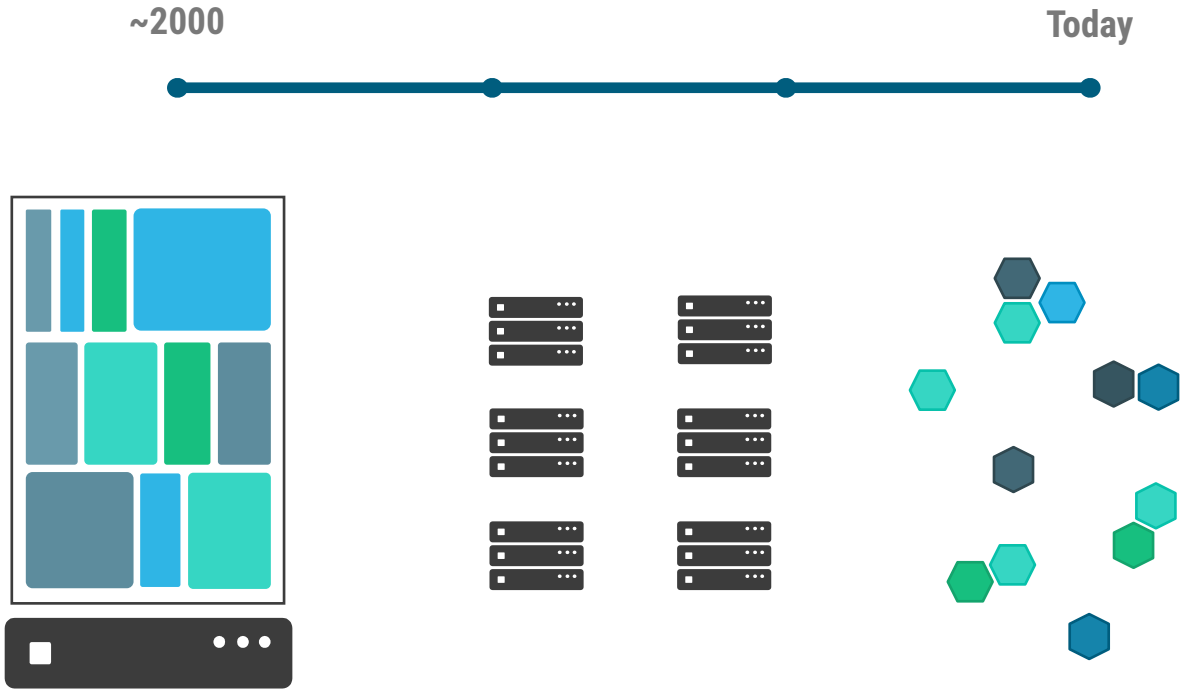



People we have
People we need

# Building not breaking

**If it doesn’t come on by default, no one will use it**

# Biased Timeline of the Evolution of Security

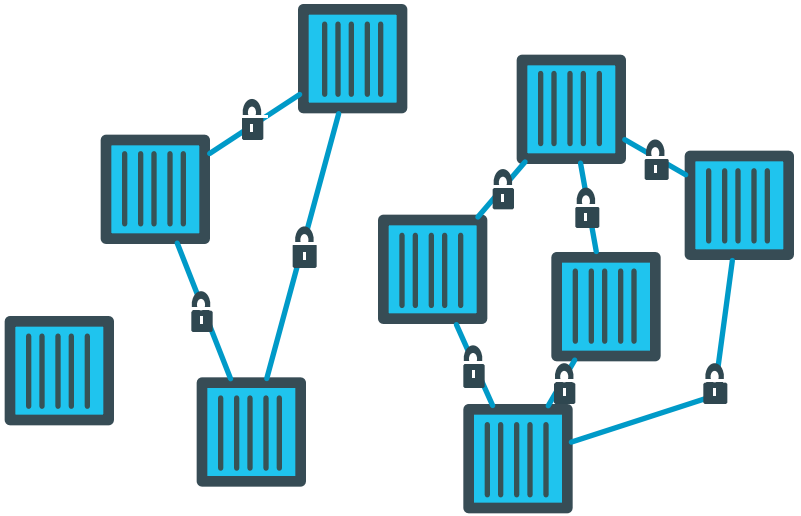

~2000
Today

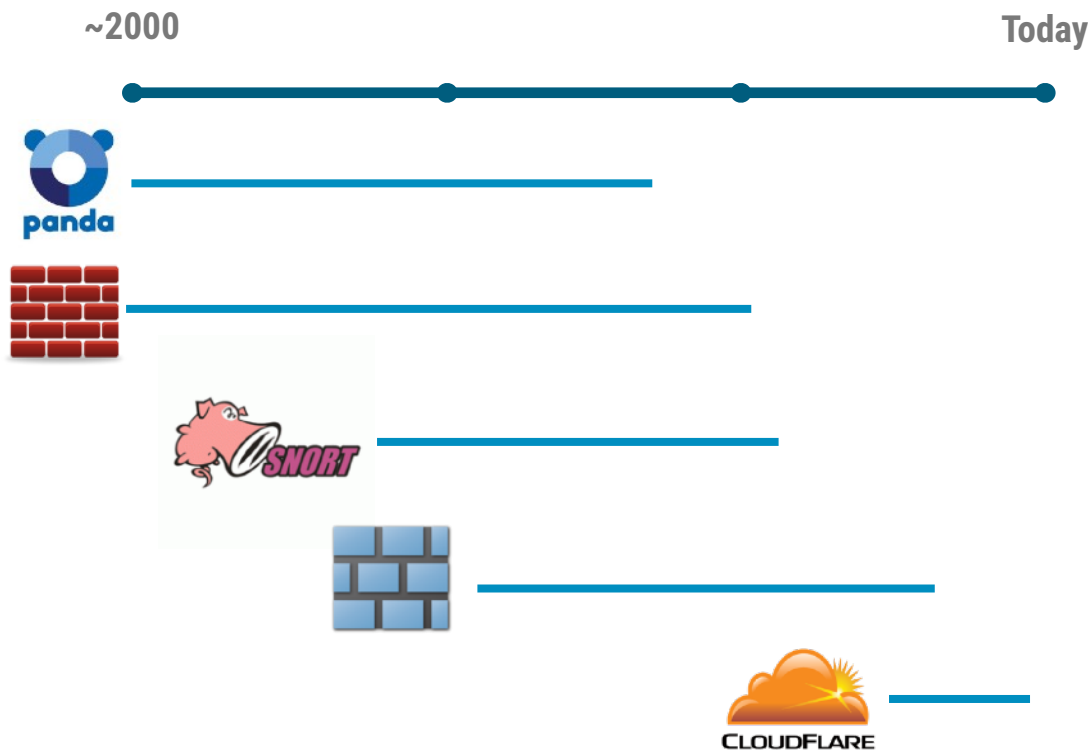
# Immutable Infrastructure

```
# echo "evil!" > /bin/sh
/bin/sh: can't create /bin/sh: Read-only file system
```

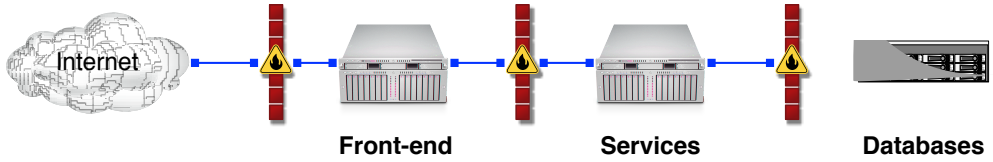
# Application Containers

# Decentralized Authorization




~2000
Today
panda
SNORT
CLOUDFLARE

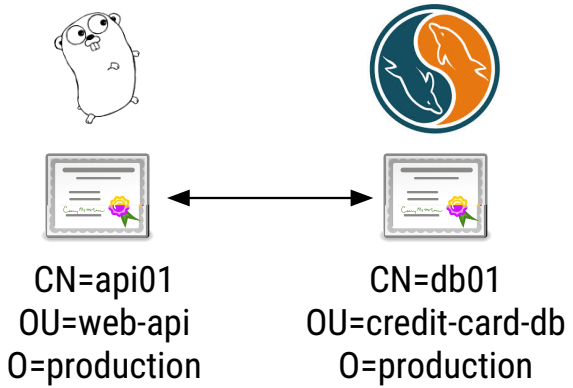
# Security boundary is the service



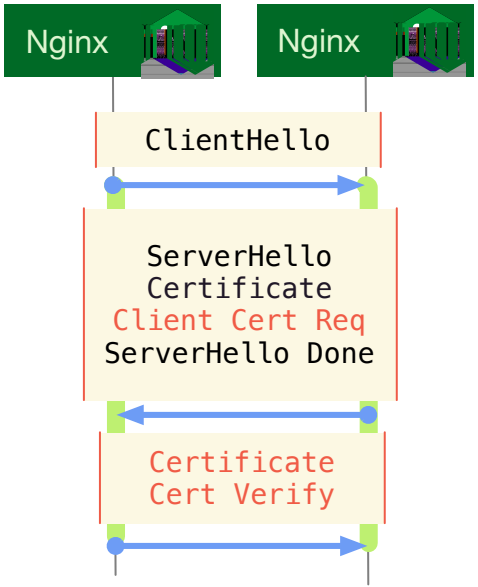
# One App Instance, One ID

# One Node,
# One ID

Every service call is authorized and authenticated



# mTLS


Nginx
Nginx
ClientHello
ServerHello
Certificate
Client Cert Req
ServerHello Done
Certificate
Cert Verify

# mTLS - the good

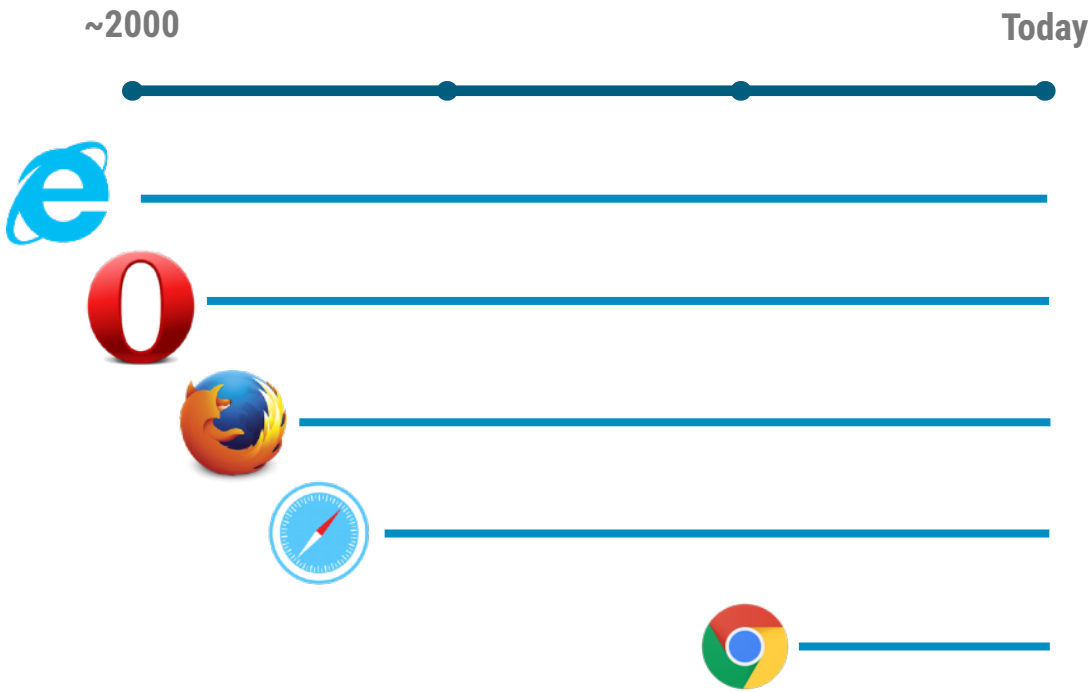
1. **Key-material stays secret**
2. **Supported everywhere**
3. **Authentication and Encryption**

# mTLS - the bad

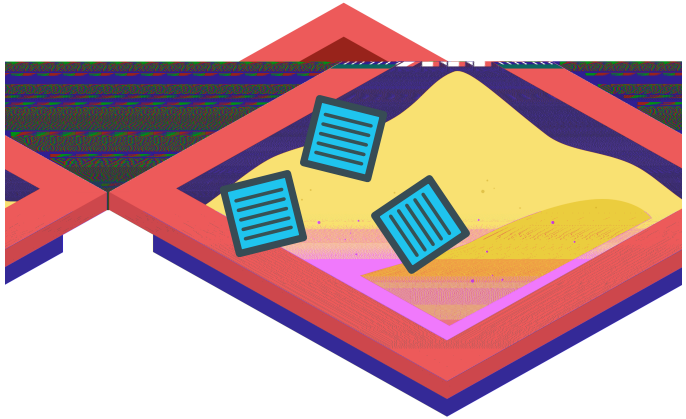
1. A LOT of certs
2. Confusing for developers
3. No good revocation story
4. Running a PKI is hard
5. Unforgiving

# Least-privilege resource access

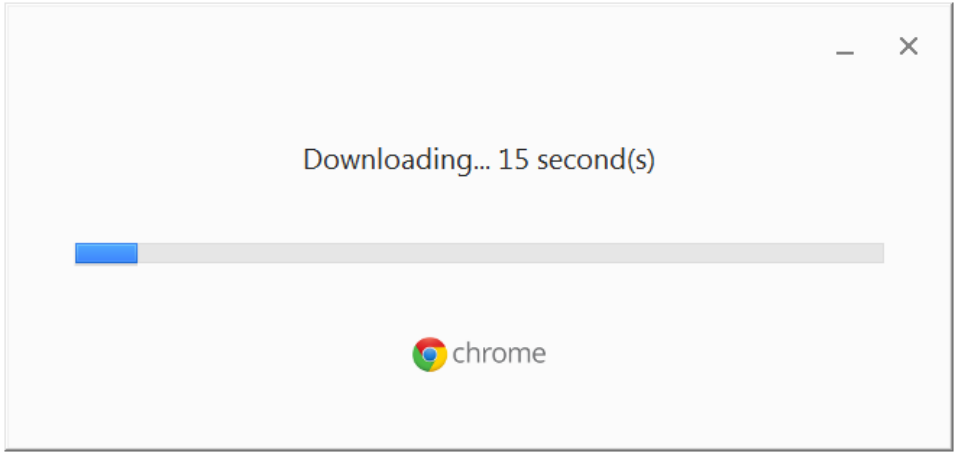
```
[ { "permission":
        { "method": "GET", "resource": "/user" },
    "allow": ["web", "fulfillment", "payments"] },

  { "permission":
        { "method": "POST", "resource": "/user" },
    "allow": ["signup", "web"] },

  { "permission":
        { "method": "DELETE", "resource": "/user/.*" },
    "allow": ["web"]
}]
```


~2000
Today

# Sandboxing

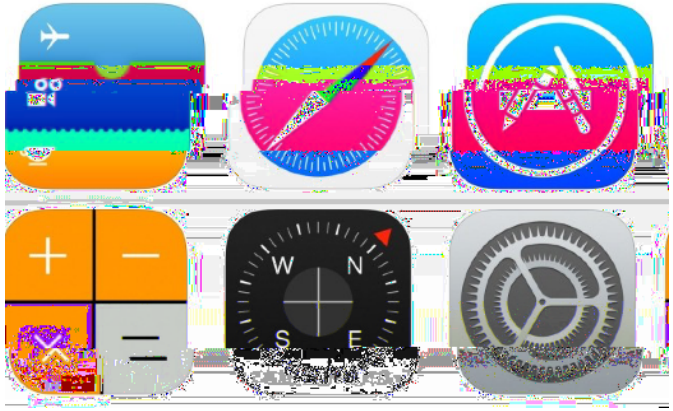


# Transparent Updates

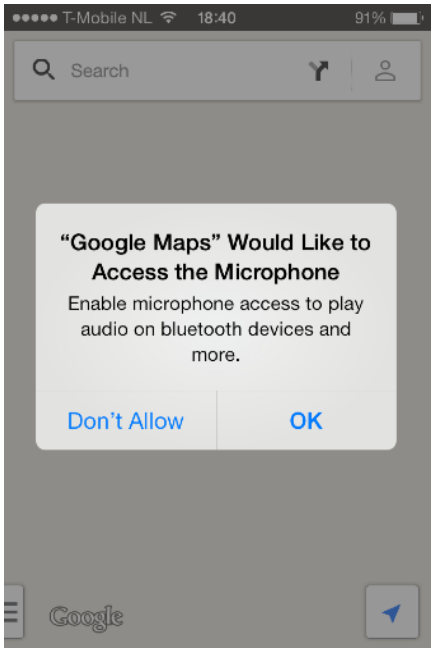



1112223333

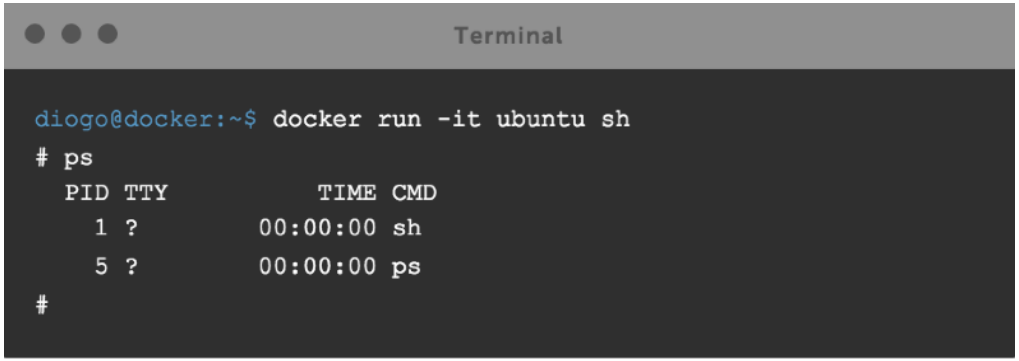
# Application Whitelist



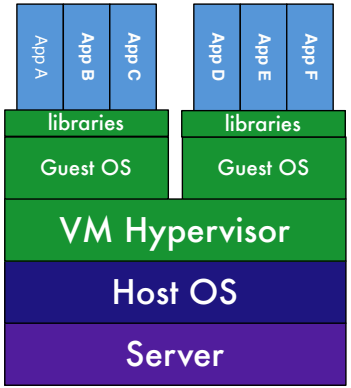
# Usable Security

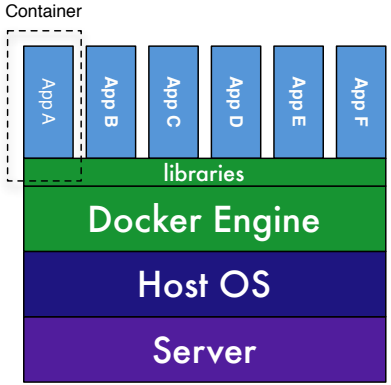


# A container is isolated



# Containers VS VMs

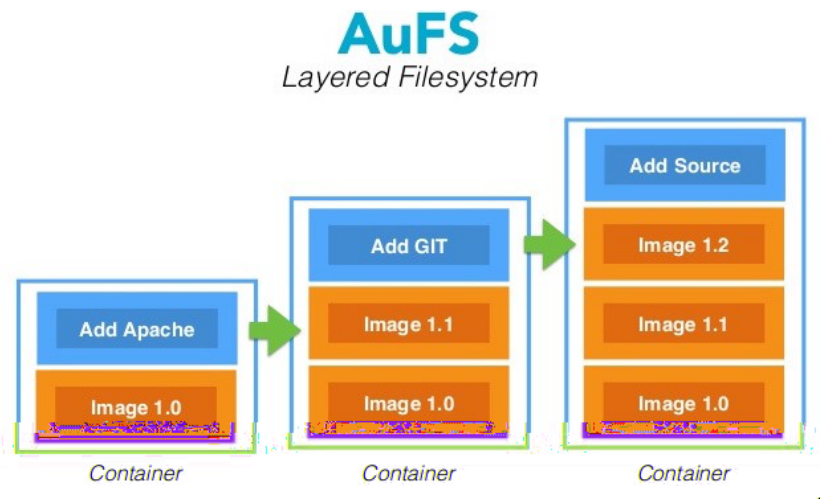


# A container is a process

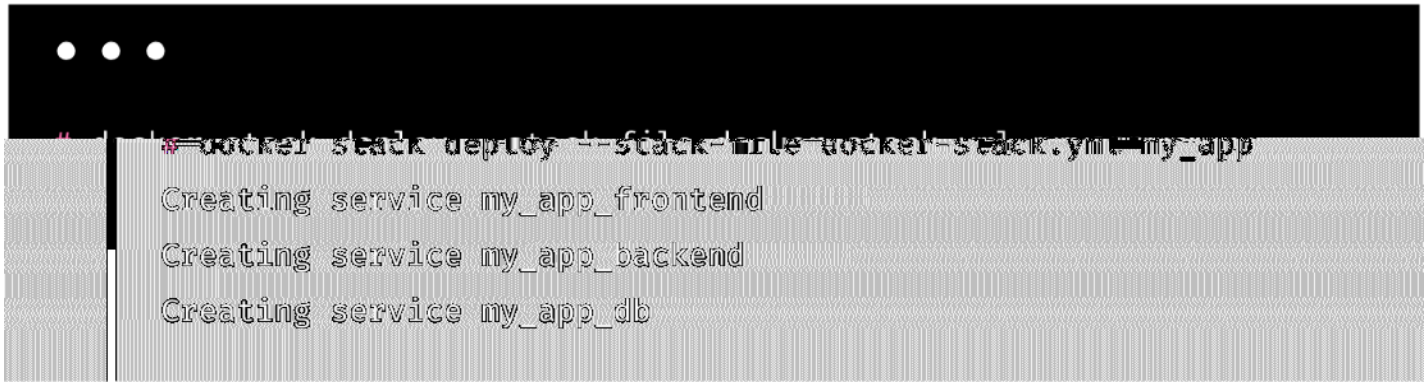
```
root@ubuntunew:/home/diogo# strace -c -f -p 6389
Process 6389 attached
Process 6476 attached
Process 6477 attached
Process 6479 attached
Process 6480 attached
Process 6481 attached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 62.83    0.000747          68        11         6 wait4
  7.91    0.000094           6        17           lstat
  5.05    0.000060          15         4           getdents64
  3.45    0.000041           2        19           open
  3.20    0.000038           0       106           read
  2.61    0.000031          31         1           nanosleep
  2.02    0.000024           2        10           mprotect
  1.85    0.000022           1        38           rt_sigaction
  1.60    0.000019           0        43           ioctl
  1.43    0.000017           1        20           close
  1.26    0.000015           0        83           writev
```

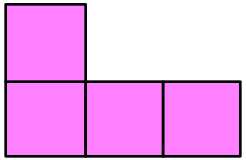
# Copy-on-write FS

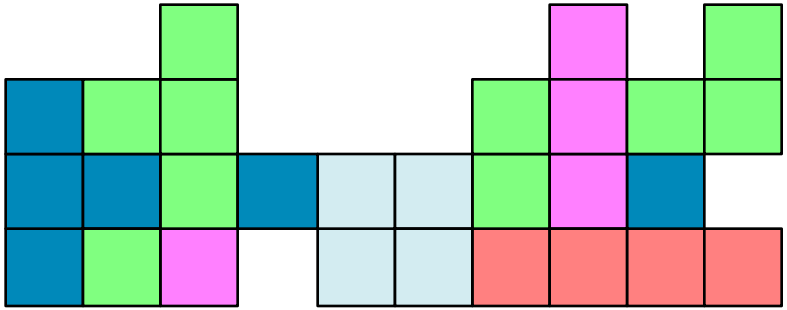


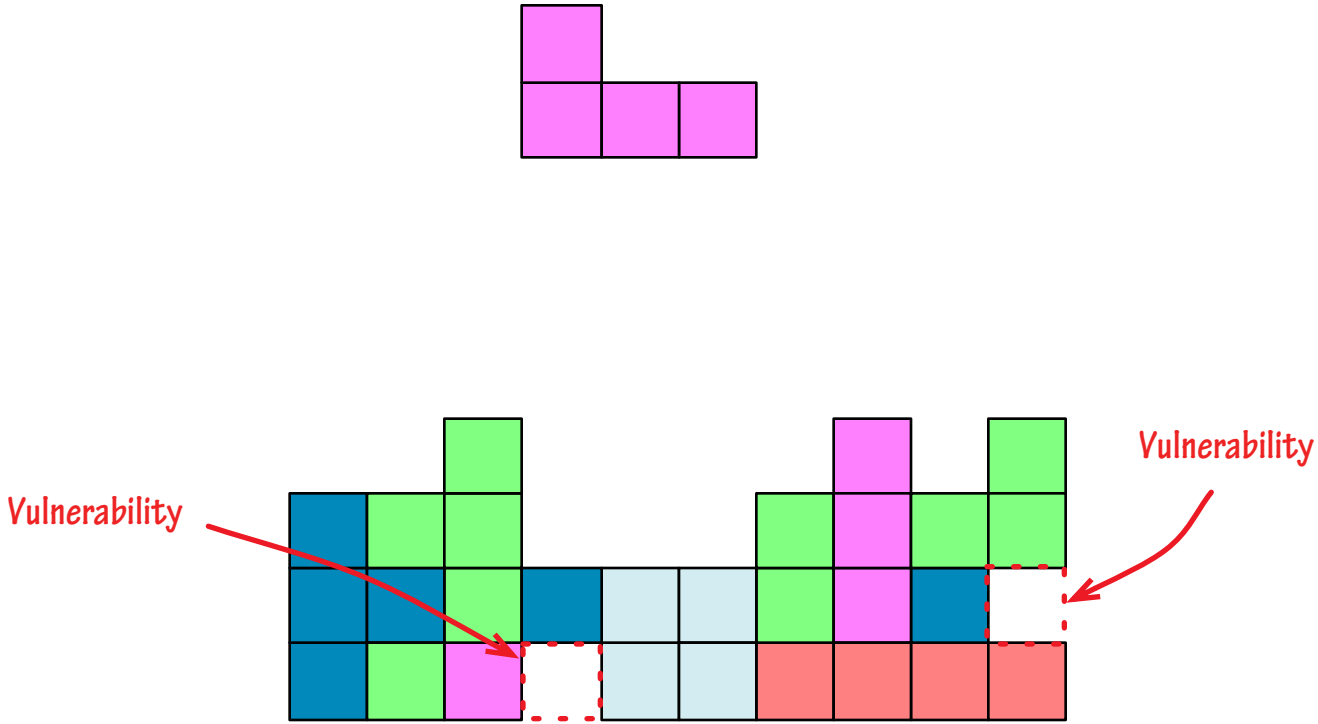
# Key Security advantages

1. **Additional layer of isolation**
2. **Easier patch management, faster deployments, reduced risk**
3. **Better visibility into application behavior**


$ docker stack deploy --stack-file docker-stack.yml my_app
Creating service my_app_frontend
Creating service my_app_backend
Creating service my_app_db

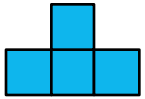




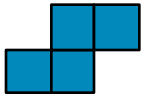

Vulnerability
Vulnerability

# Security Tetrominos

runC

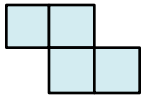


containerD

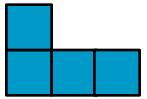


Docker

infraKit

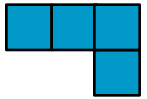


linuxKit



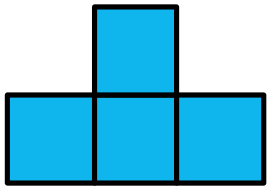
Notary

swarmKit



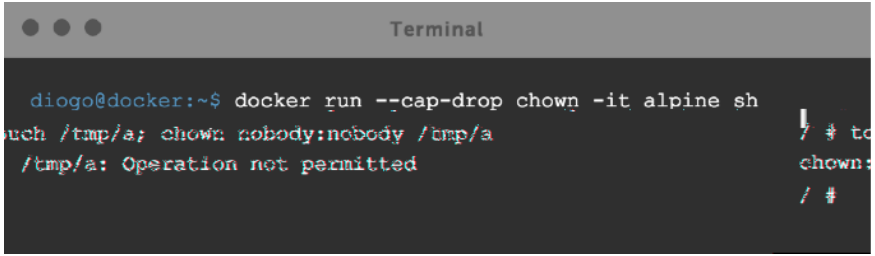
# runC

## Lightweight universal container runtime



# runC

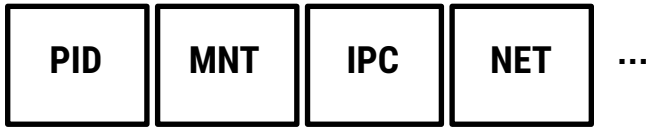
Capabilities



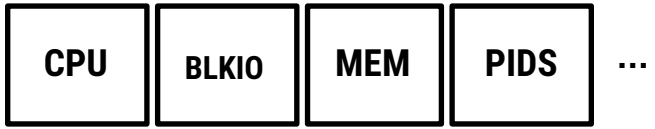
# runC

- Namespace Isolation
- Cgroups

## Namespaces




## Cgroups



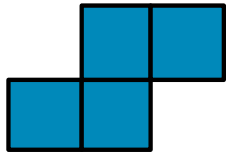
# runC

## Seccomp-bpf

```
{
    "defaultAction": "SCMP_ACT_ERRNO",
    "architectures": [
        "SCMP_ARCH_X86_64",
    ],
    "syscalls": [
        {
            "name": "accept",
            "action": "SCMP_ACT_ALLOW",
            "args": []
        },
        {
            "name": "accept4",
            "action": "SCMP_ACT_ALLOW",
            "args": []
        },
        ...
    ]
}
```

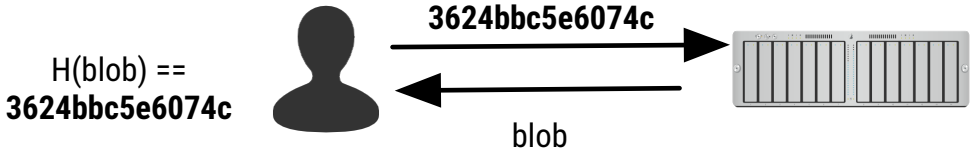
# containerD

## Container runtime supervisor



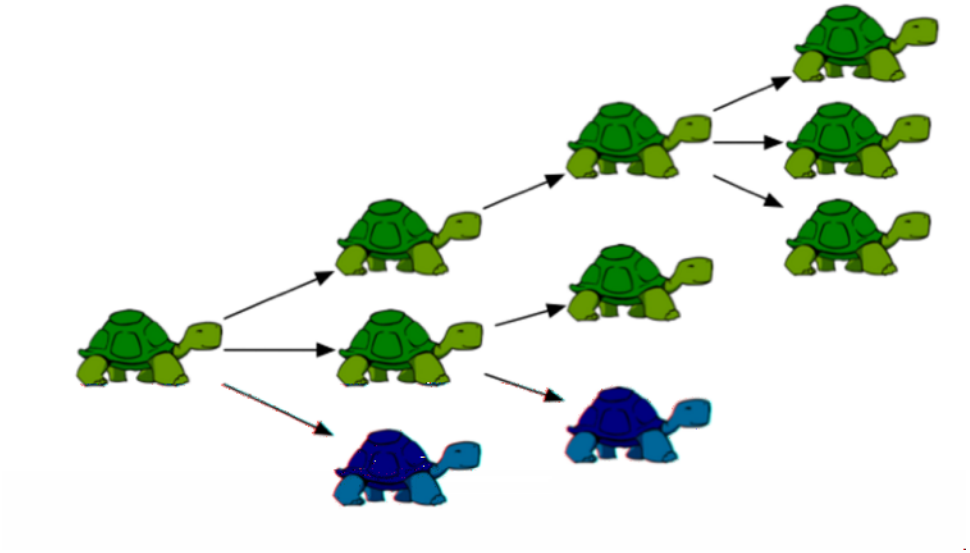
# containerD

Content addressable images



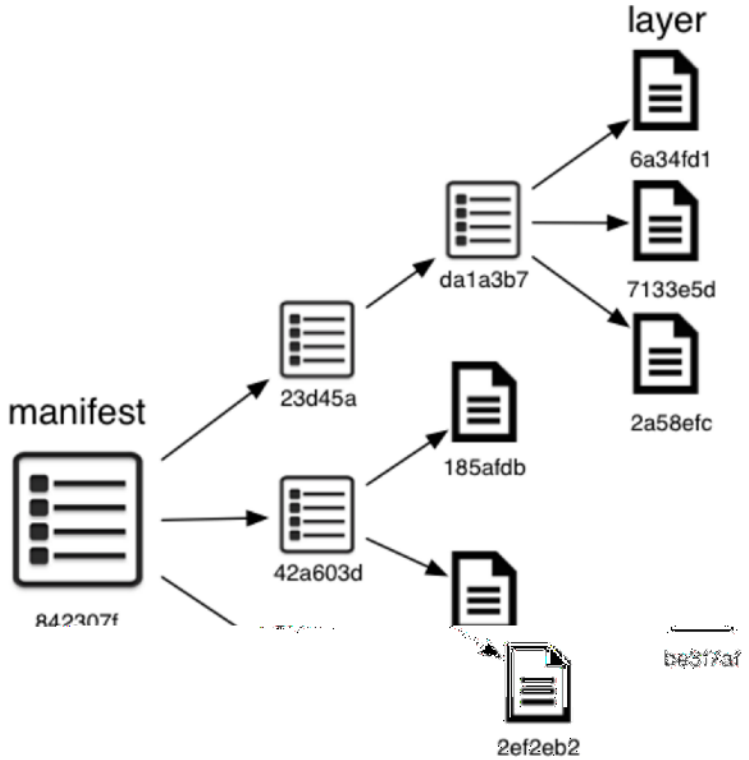
# containerD

Turtles all the way down



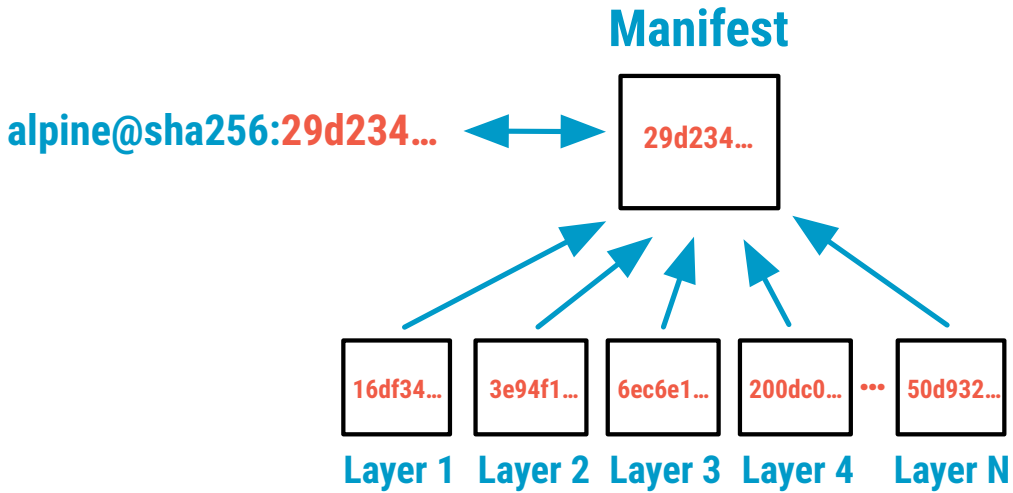
# containerD

Images are Merkle DAGs of layers



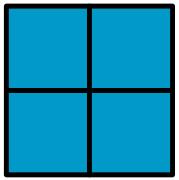
# containerD

Content Addressable Image Pulls



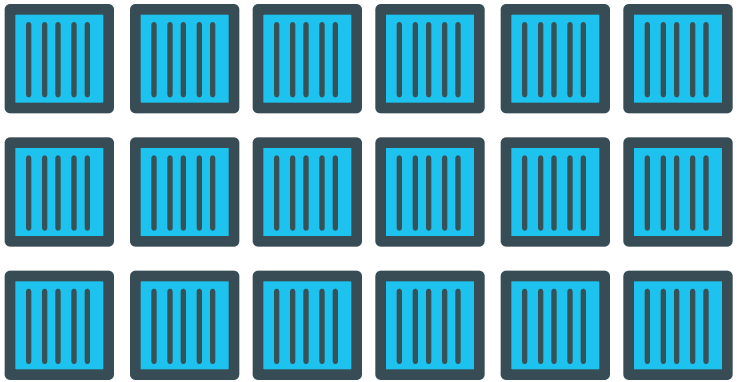
# Docker

## Secure-by-default software container platform



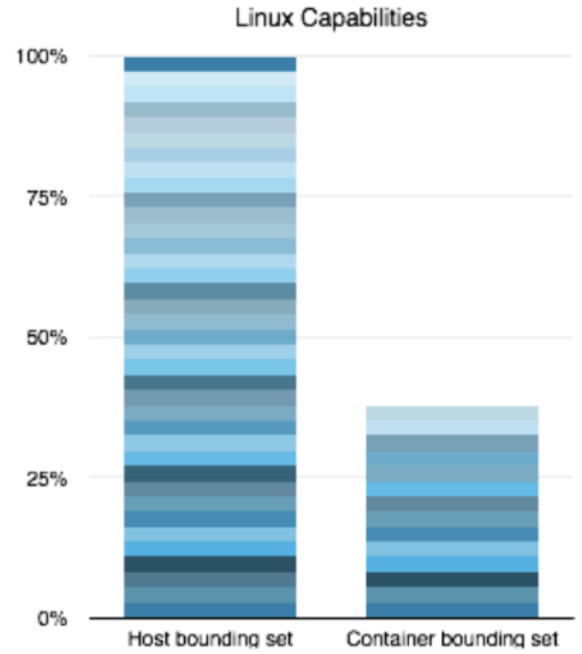
# Docker

- SELinux & AppArmor
- Capability Whitelist
- Syscall Whitelist
- Copy-on-write
- No device access



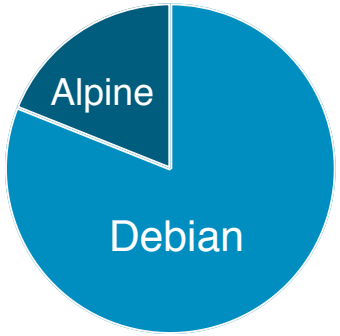
# Docker

Less than half the default capabilities



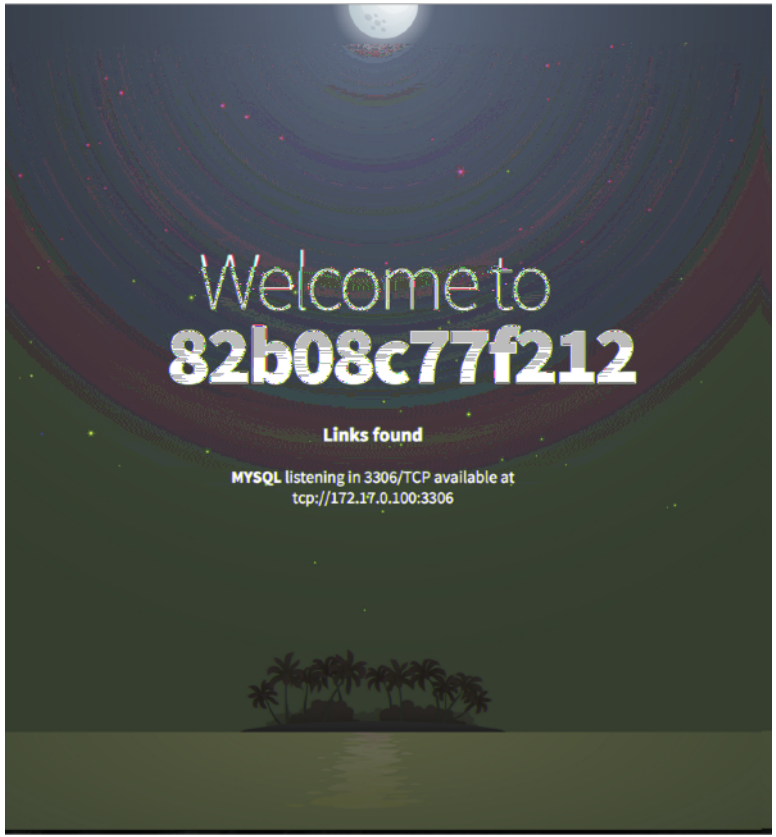
# Docker

Minimal Distros



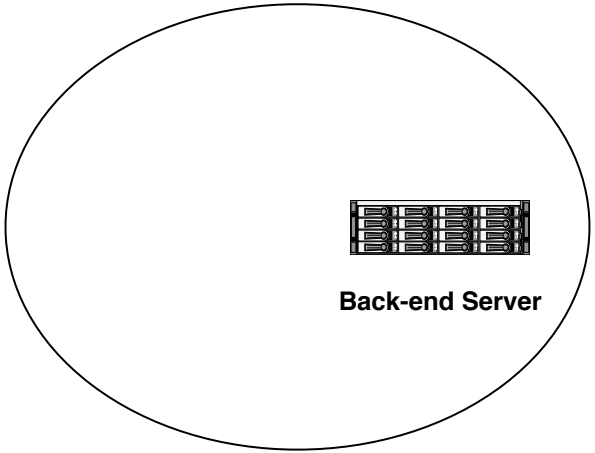
# Demo

Our new website



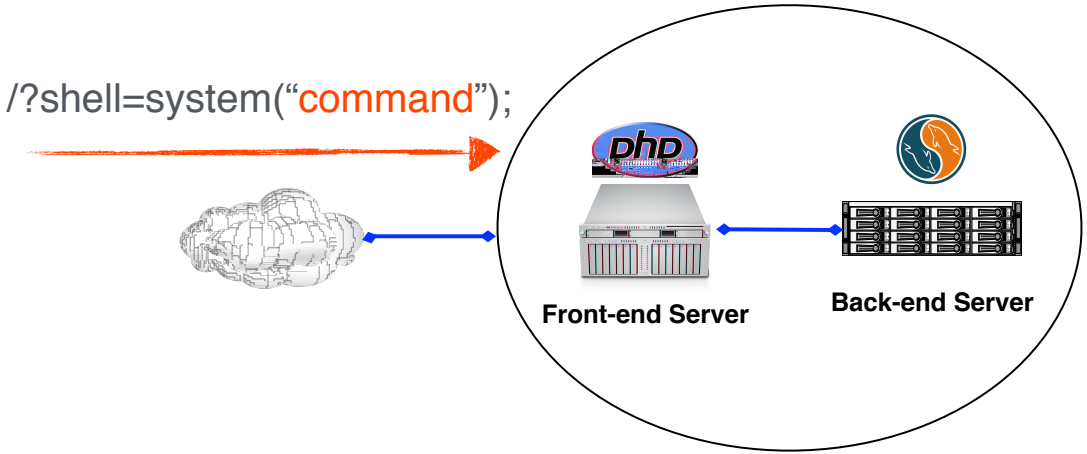
# Demo

Our new architecture



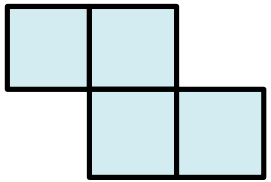
# Demo

Our new backdoor



# infraKit

## Infrastructure independent machine management

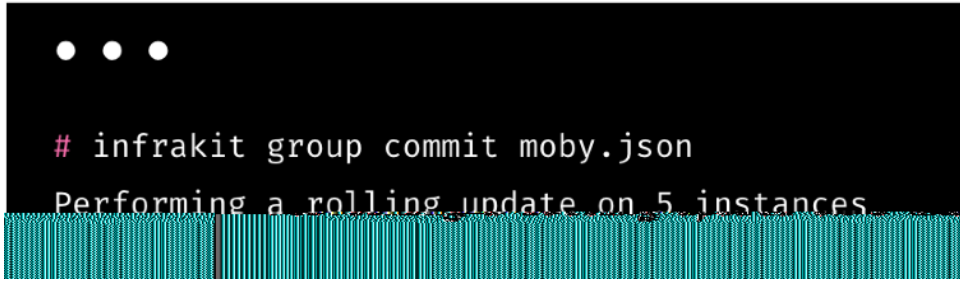


# infraKit

Platform Agnostic

# infraKit

## Declarative Updates



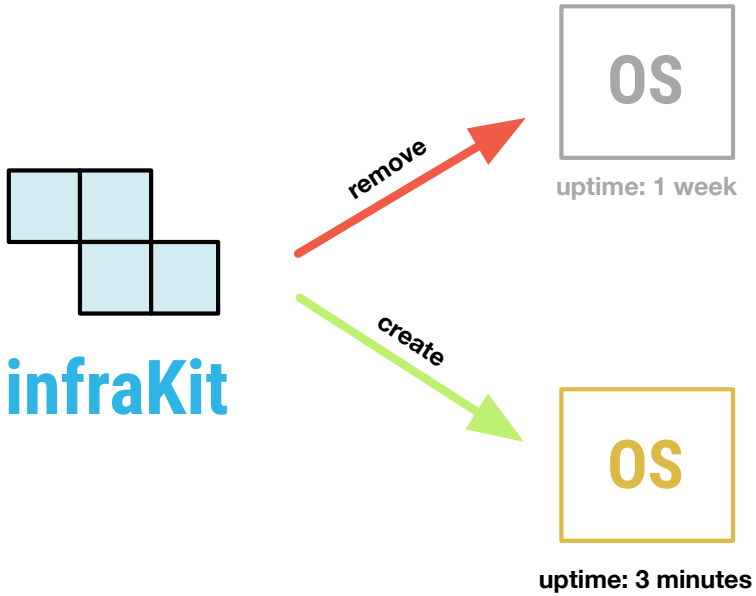
# infraKit

Reverse Uptime

10:00

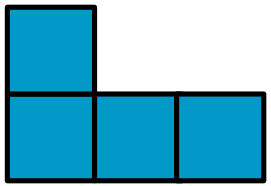
# infraKit

Rolling Deploys



# linuxKit

## A secure OS builder for your containers



# linuxKit

Immutable Linux
OS builder

```
kernel:
  image: "linuxkit/kernel:4.9.x"
  cmdline: "console=ttyS0 page_poison=1"
init:
  - linuxkit/init:1b8a7e394d…
onboot:
  - name: dhcpcd
    image: “linuxkit/dhcpcd:7d2b8aaaf…
    command: ["/sbin/dhcpcd", “--nobackground" ]
trust:
  org:
    - linuxkit
```

# linuxKit

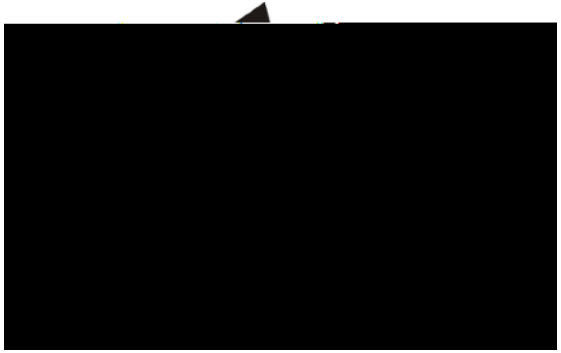
Minimal Base

# linuxKit

Type-safe
System
Daemons

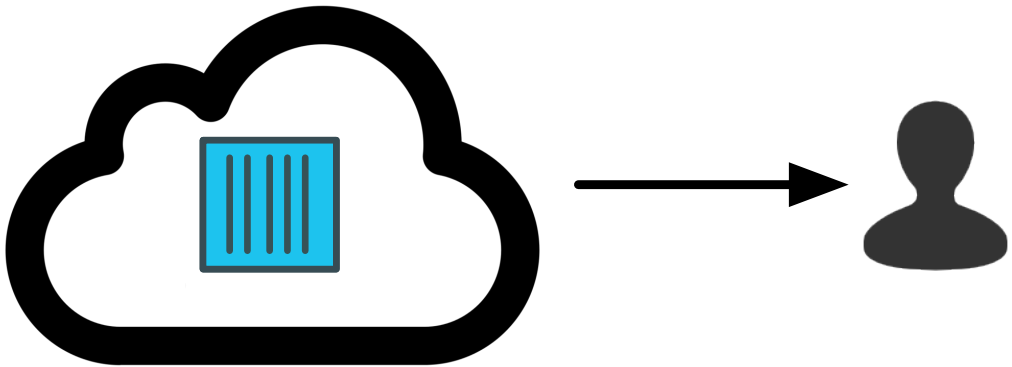
# linuxKit

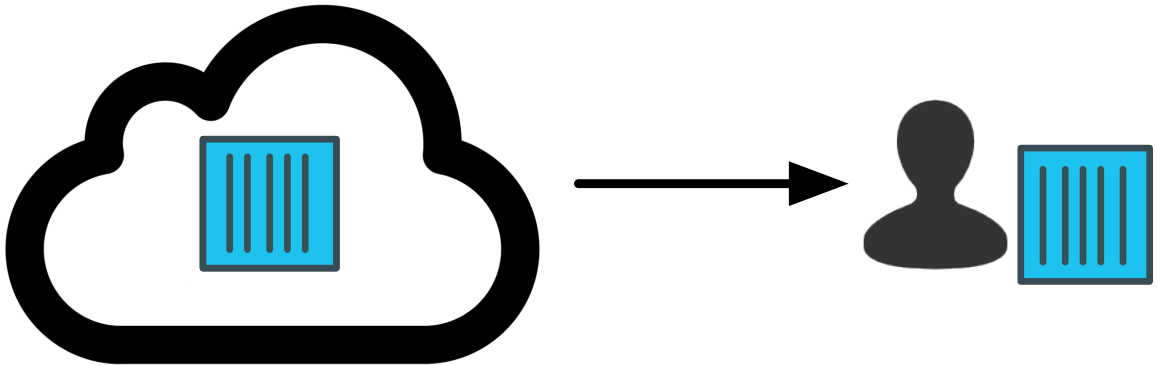
Hardening the kernel

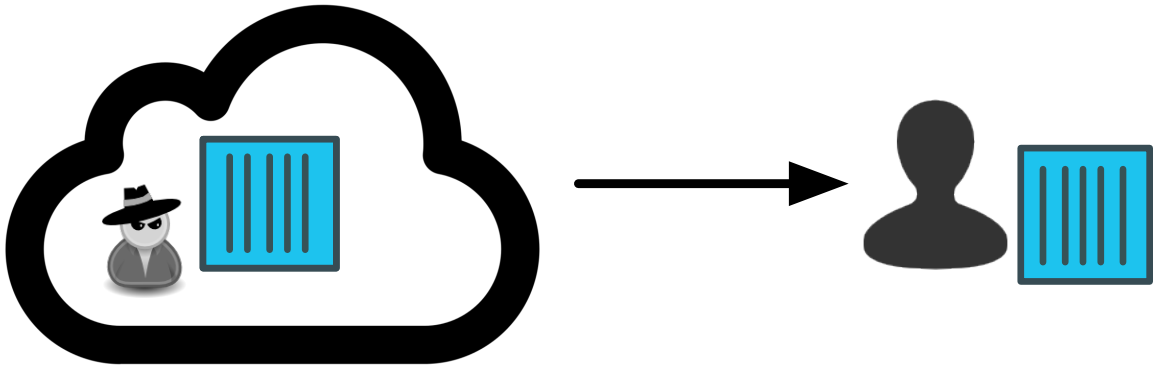


# Notary

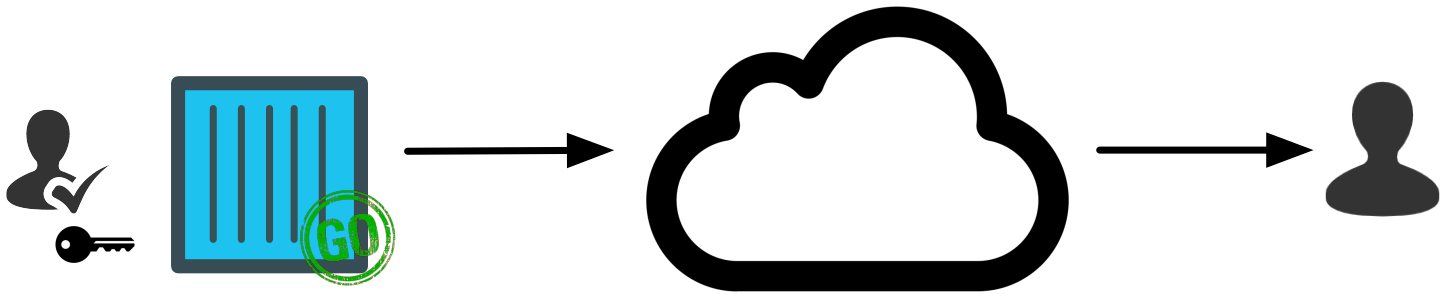
## Trusted software delivery

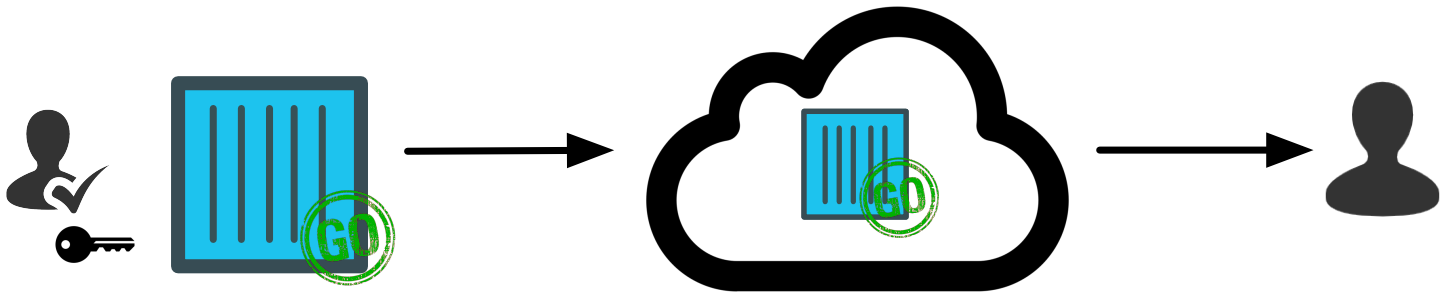




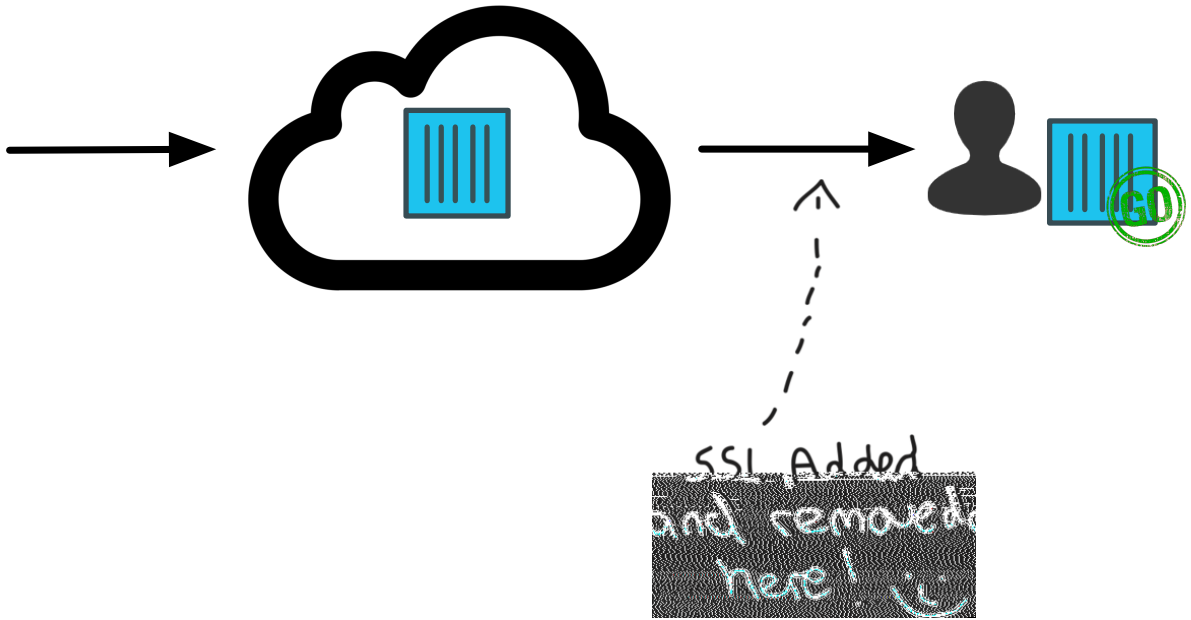


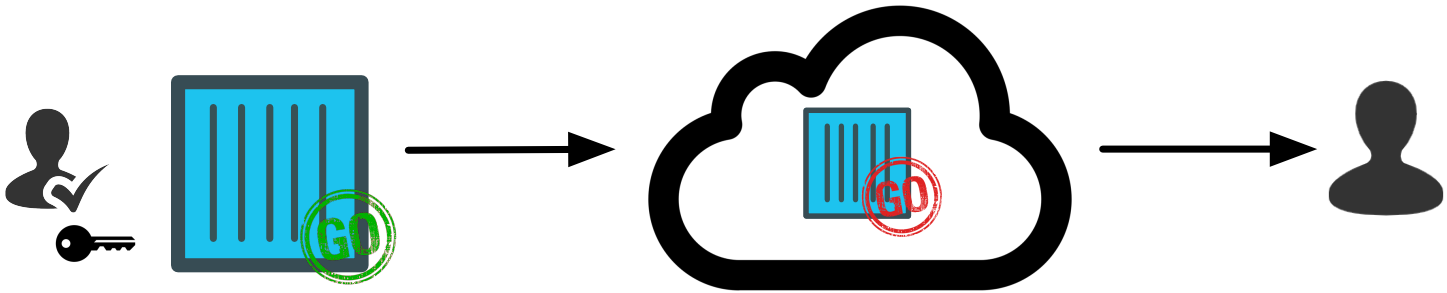
# What about GPG?

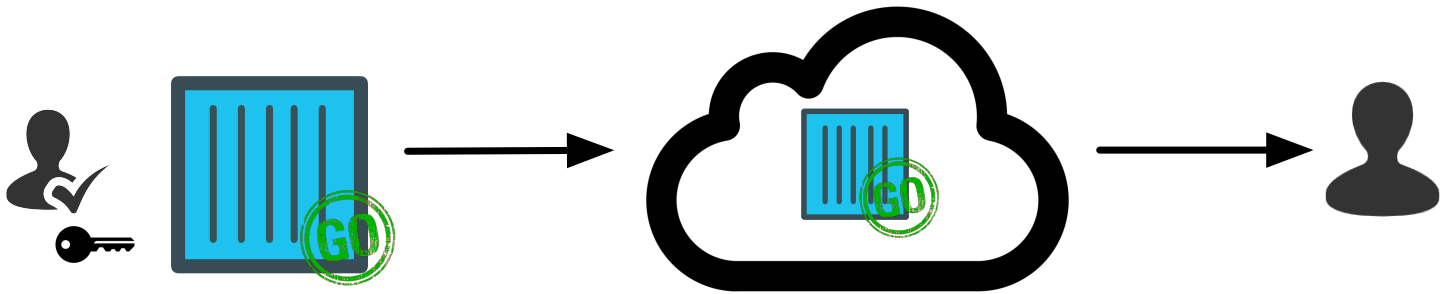

GO

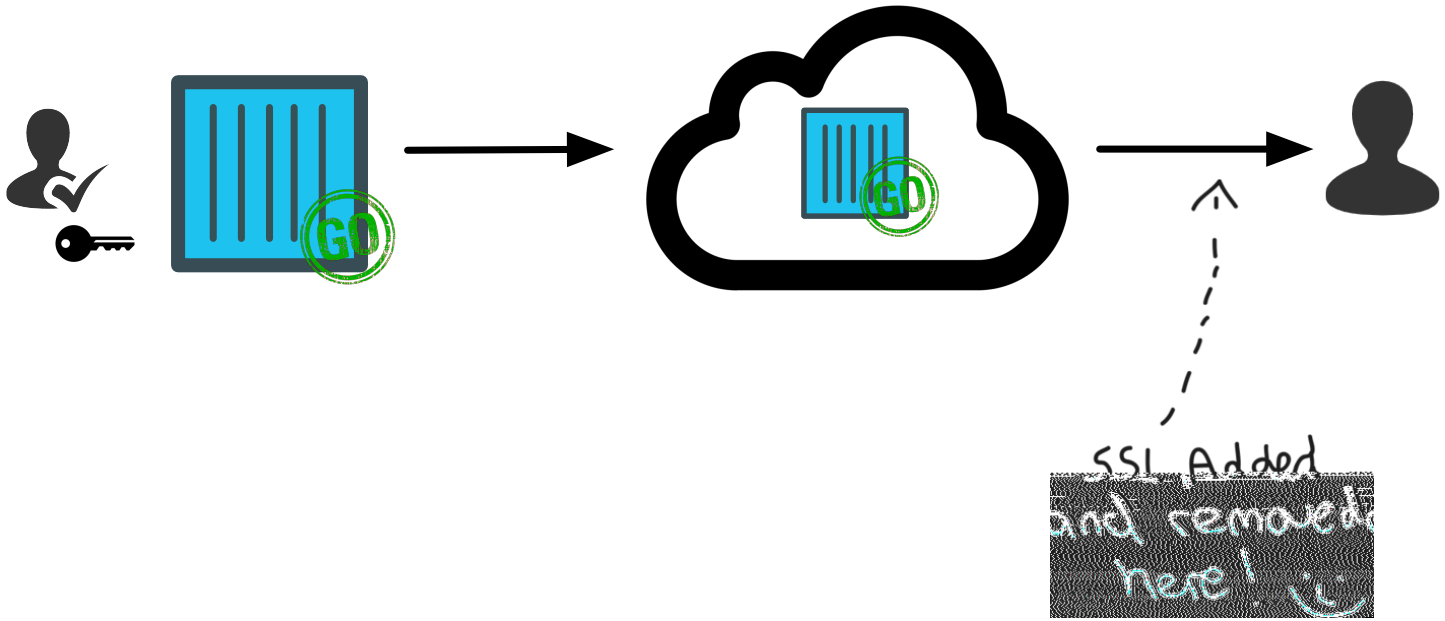

GO
GO

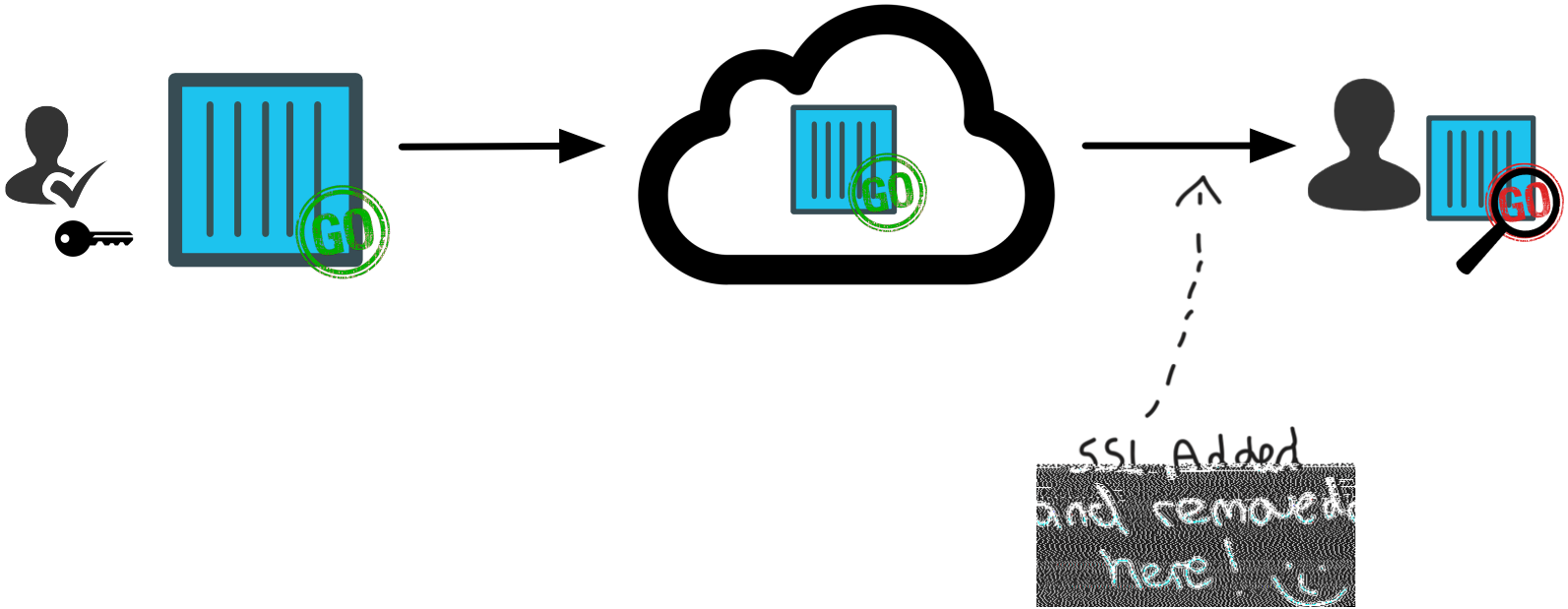

GO
GO
GO


SSL Added
and removed
here!


GO
GO


GO
GO


GO
GO
SSL Added
and removed
here!


GO
GO
GO
SSL Added
and removed
here!

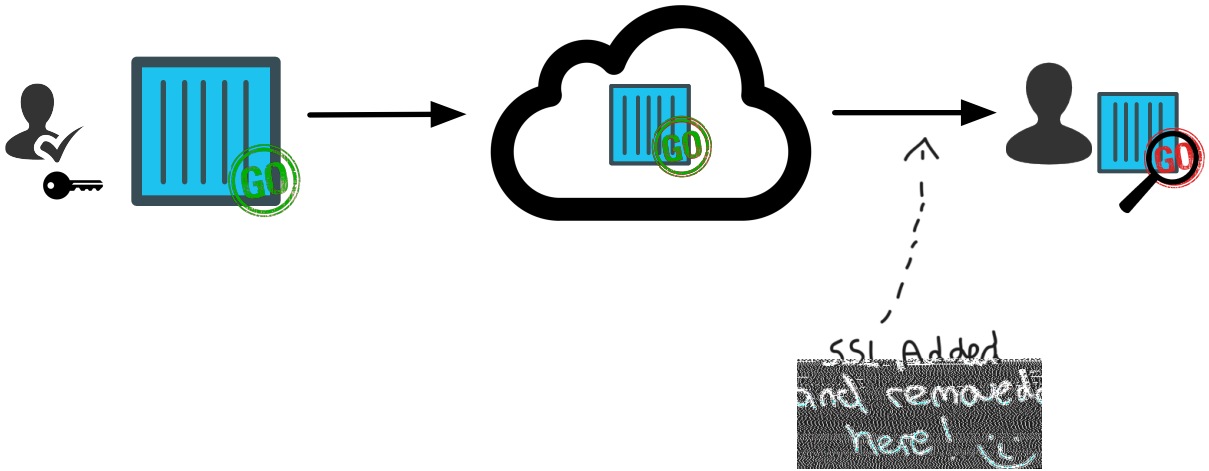

Secure Software Distribution is Hard!
arbitrary software
slow retrieval
extra dependencies
indefinite freeze
wrong software
malicious mirrors
rollback
mix-and-match
endless data
key compromise

“A software update system is secure if it can be sure that it knows about the latest available updates in a timely manner, any files it downloads are the correct files, and no harm results from checking or downloading files.”

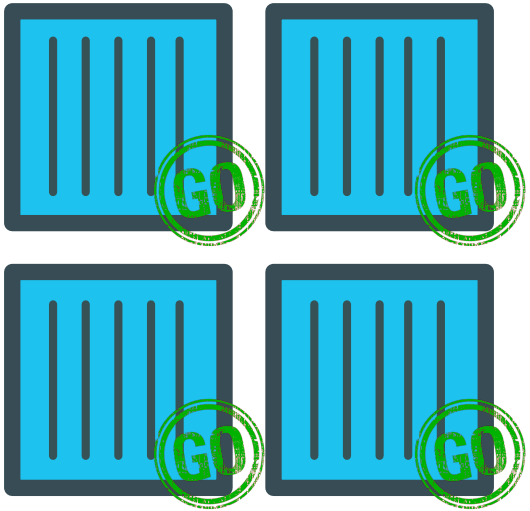
- The Update Framework

1. Freshness
2. Signed Collections
3. Key Hierarchy
4. Transparent Key Rotation
5. Threshold Signing

# Freshness



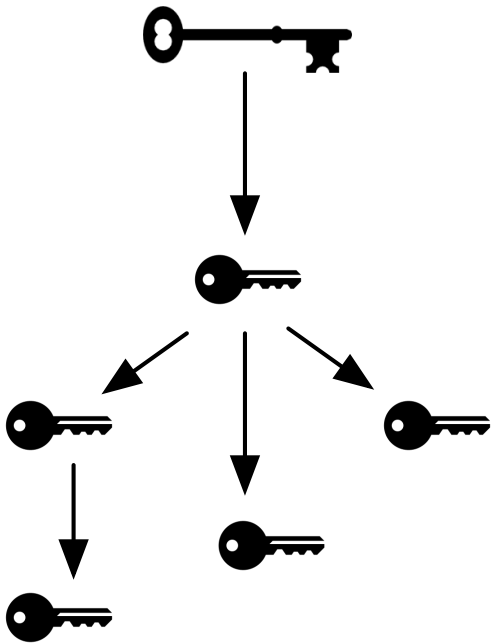
# Signed Collections



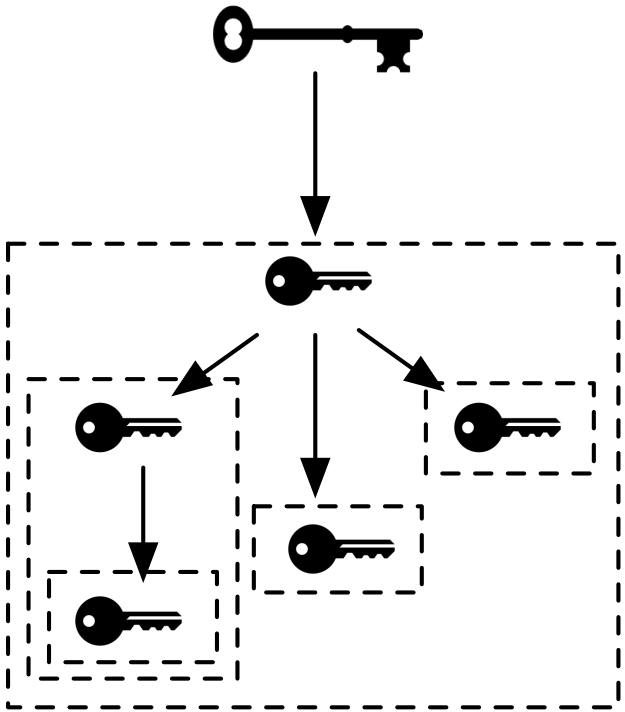

Signed Packages


Signed Collection

# Key Hierarchy



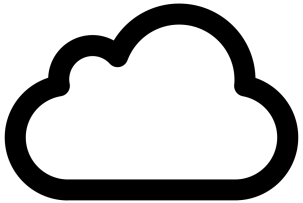
# Key Hierarchy

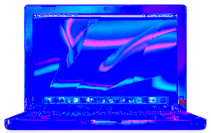


# Key Hierarchy

Short Expiry

Long Expiry

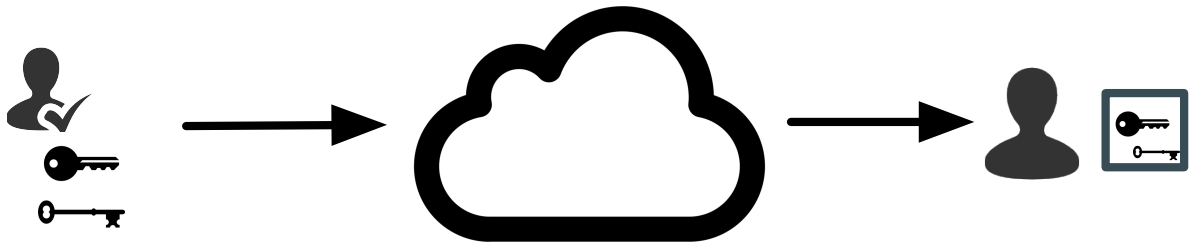




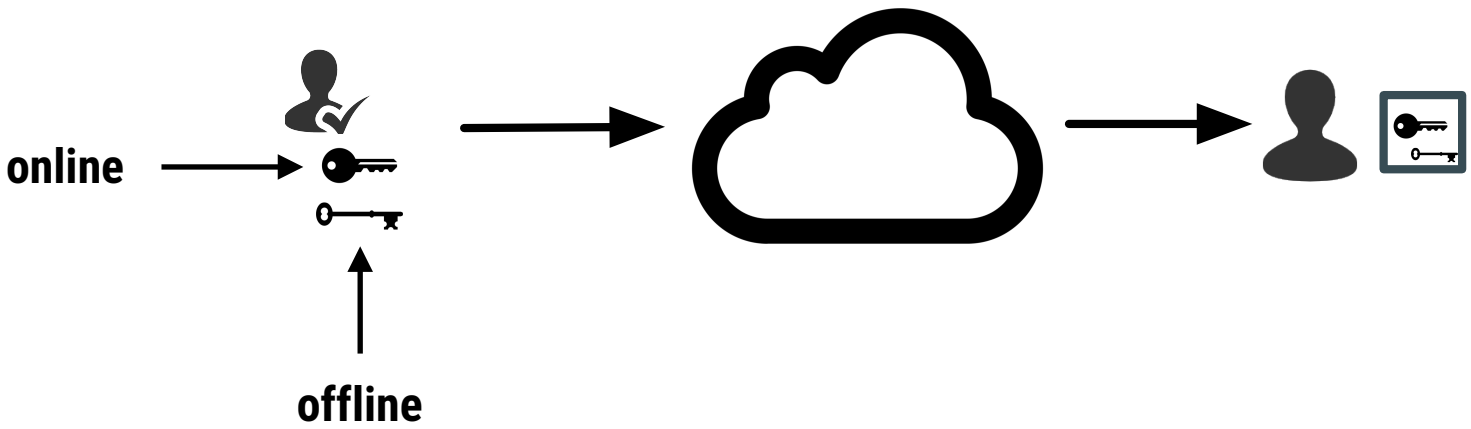
Less Sensitive

More Sensitive

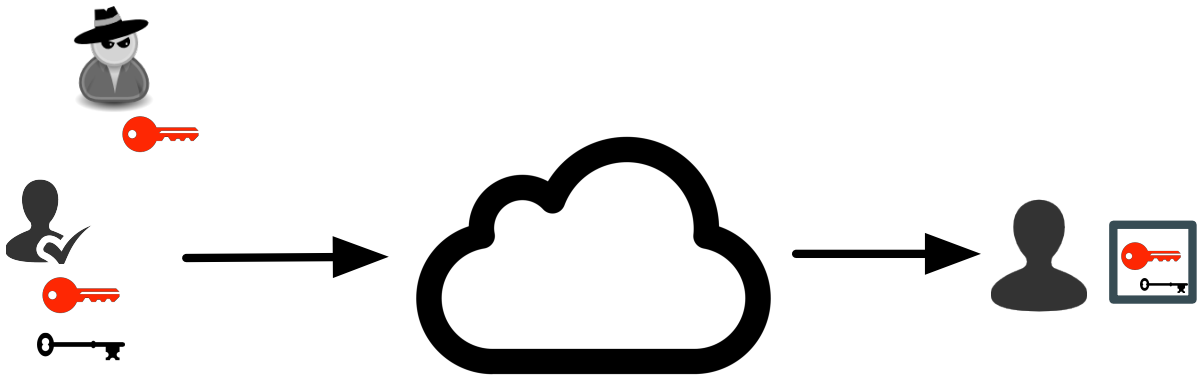
# Transparent Key Rotation



# Transparent Key Rotation

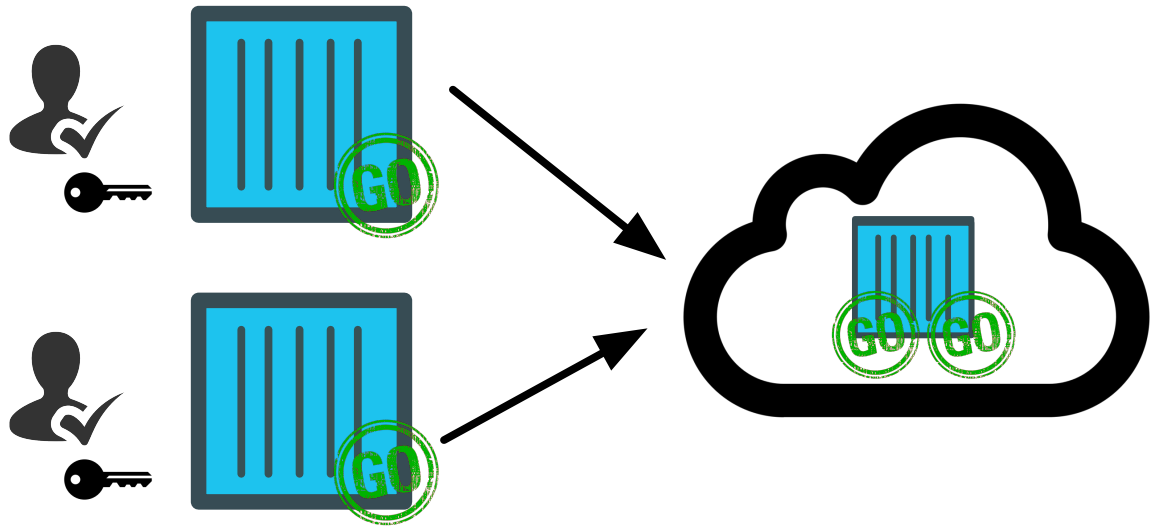


# Transparent Key Rotation

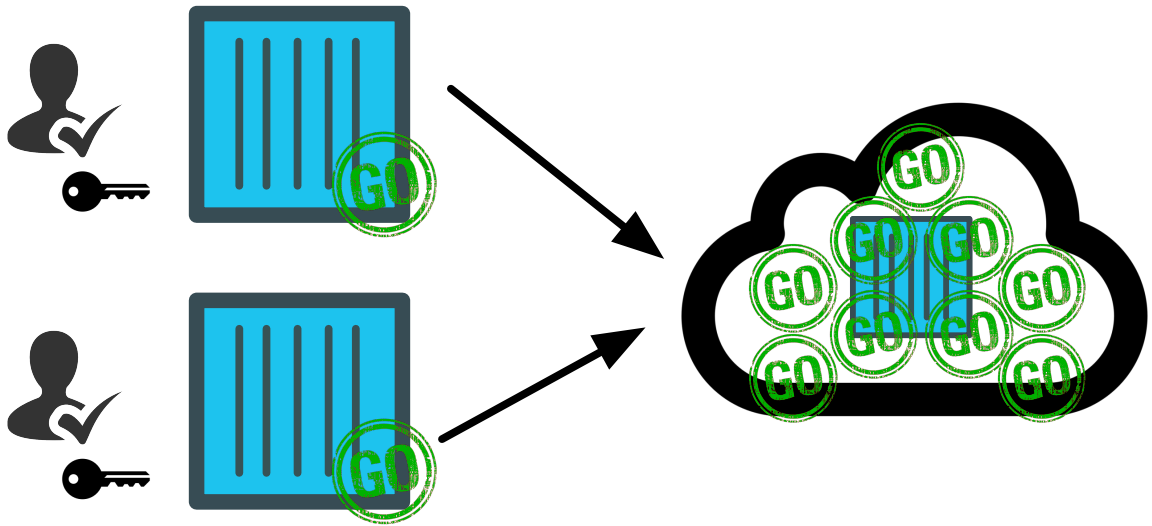


# Transparent Key Rotation

# Threshold Signing



# Threshold Signing



# The Update Framework (TUF)

theupdateframework.com

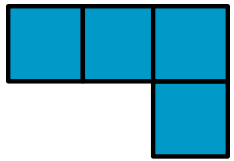
# Notary

Cryptographic
Name Resolution

# swarmKit

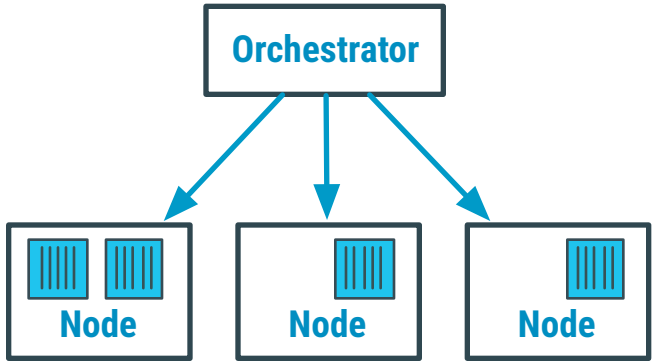
## Least-privilege

## container orchestrator



# swarmKit

Multiple
nodes

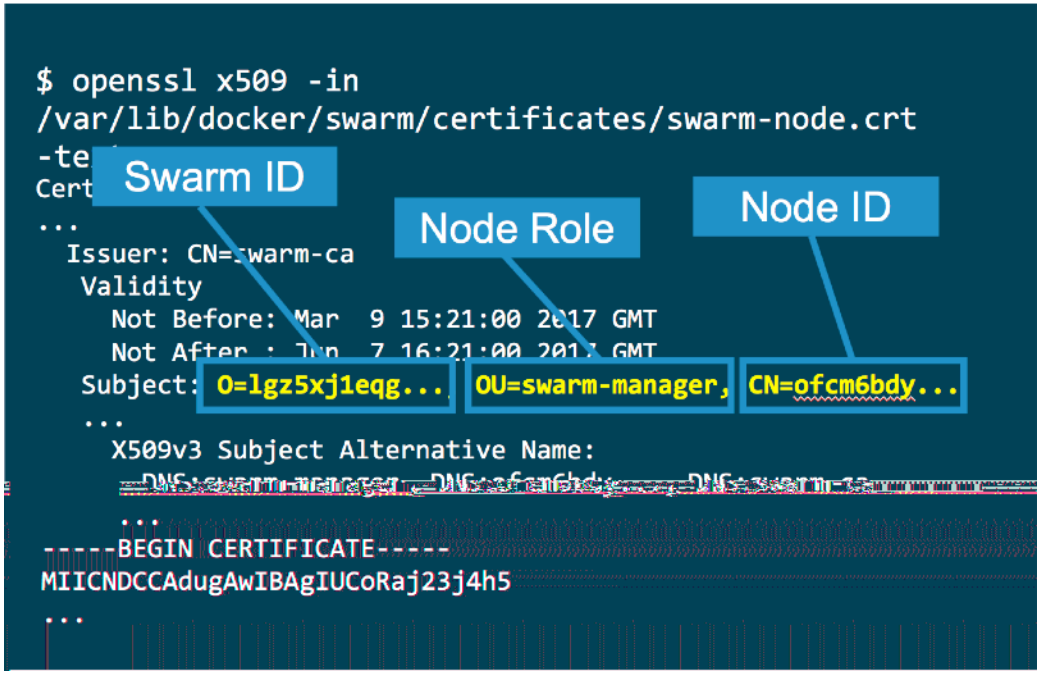


# swarmKit

## Secure Node Introduction

# swarmKit

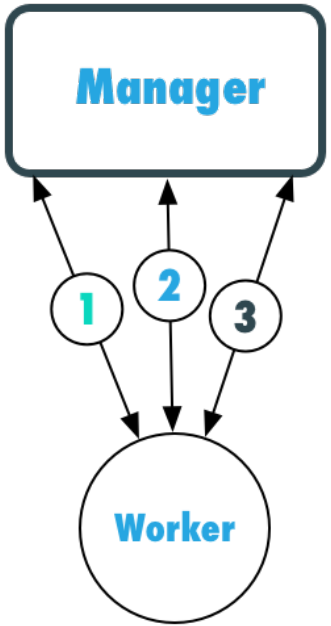
Cryptographic
Node Identity



# swarmKit

Bootstrap

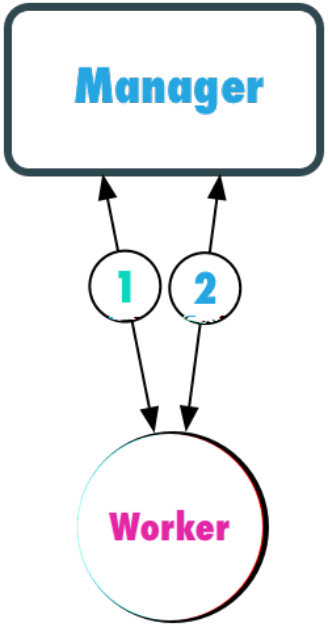
1. Retrieve and validate Root CA Public key material.
2. Submit new CSR along with secret token.
3. Retrieve the signed certificate.



# swarmKit

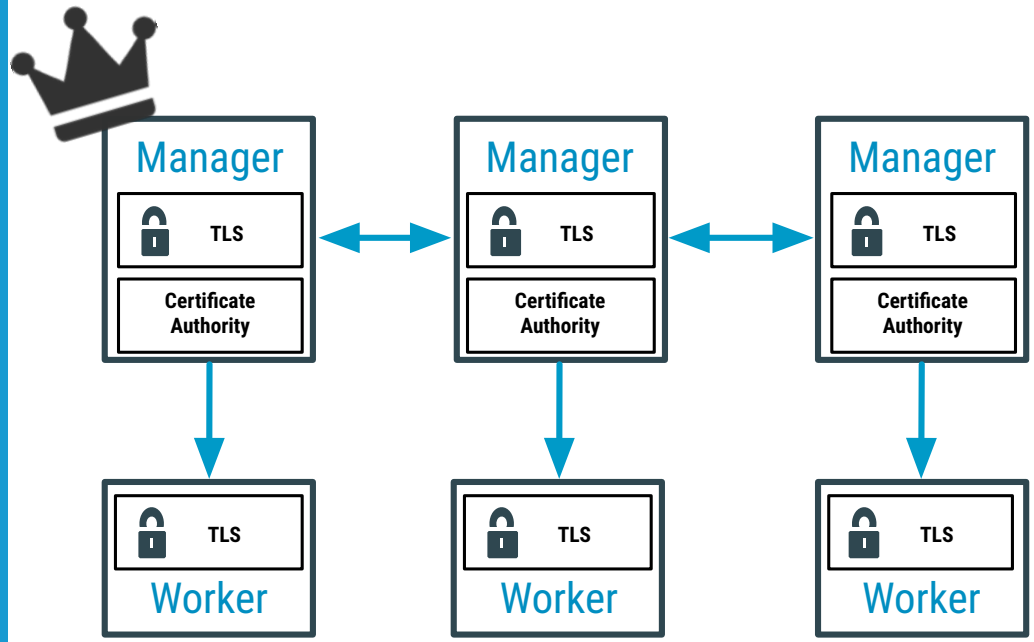
Automatic Certificate Rotation

1. Submit new CSR using old key-pair.
2. Retrieve the new signed certificate.



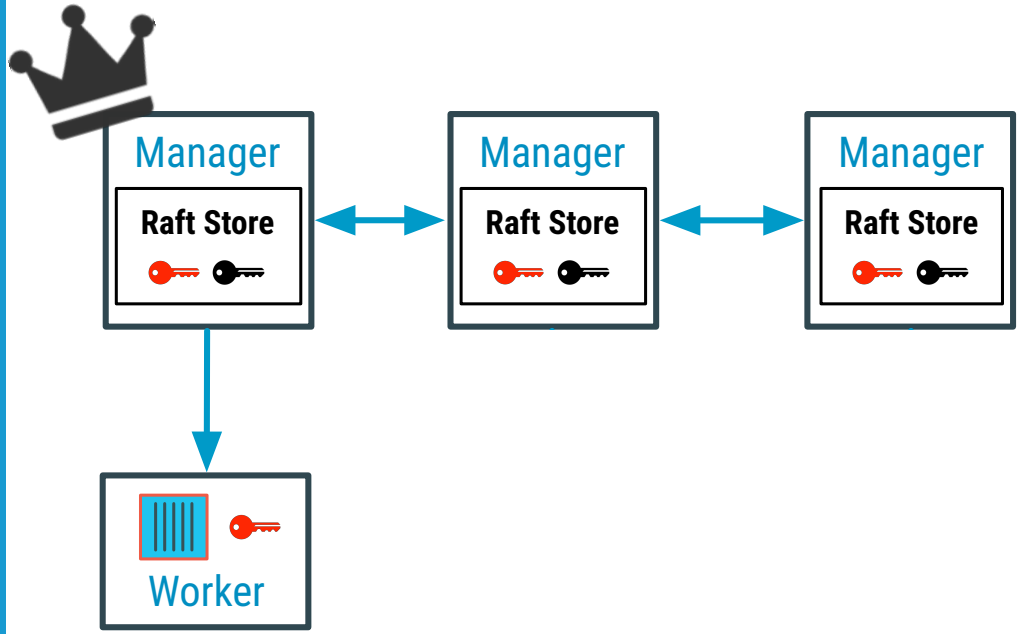
# swarmKit

MTLS Between
All Nodes



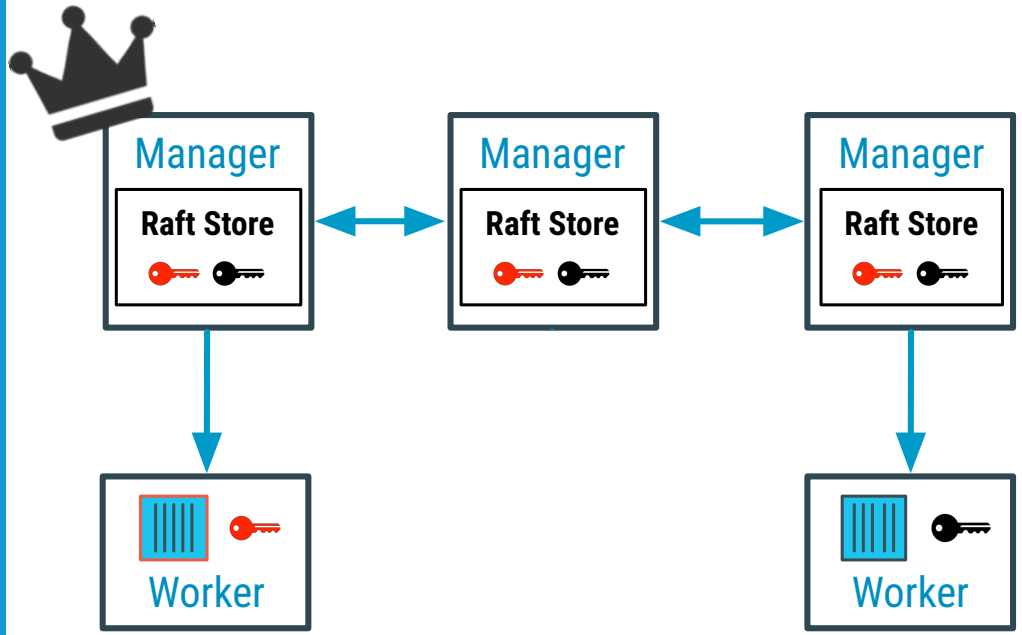
# swarmKit

Least-privilege
Secret
Distribution



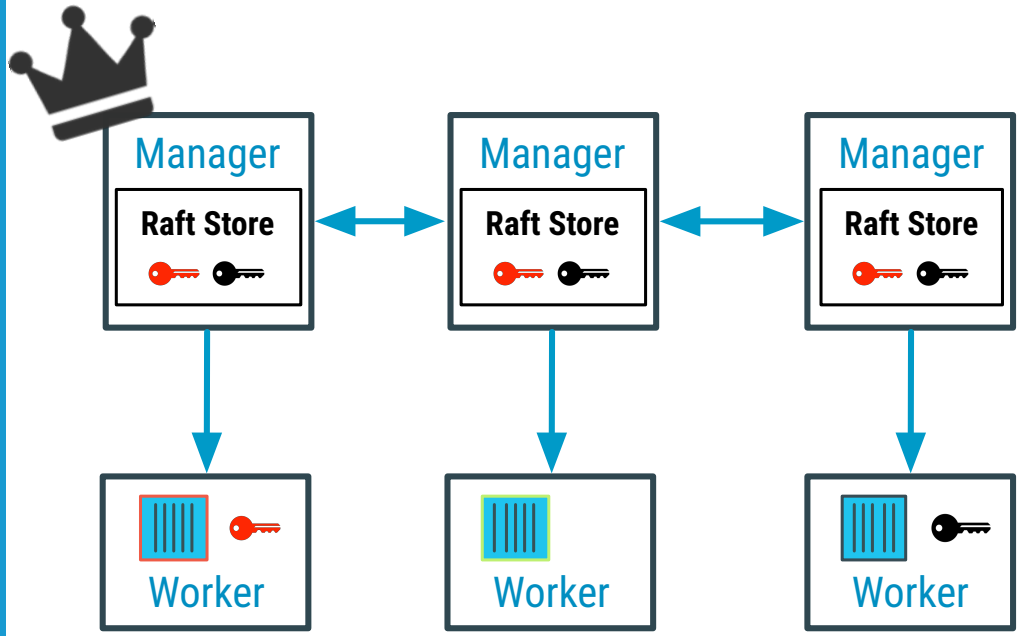
# swarmKit

Least-privilege
Secret
Distribution



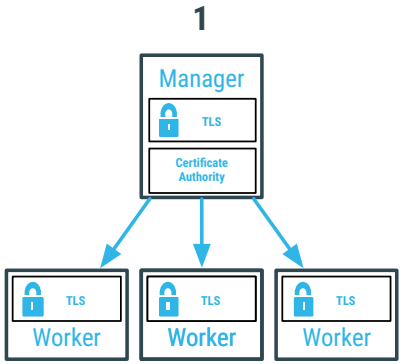
# swarmKit

Least-privilege
Secret
Distribution



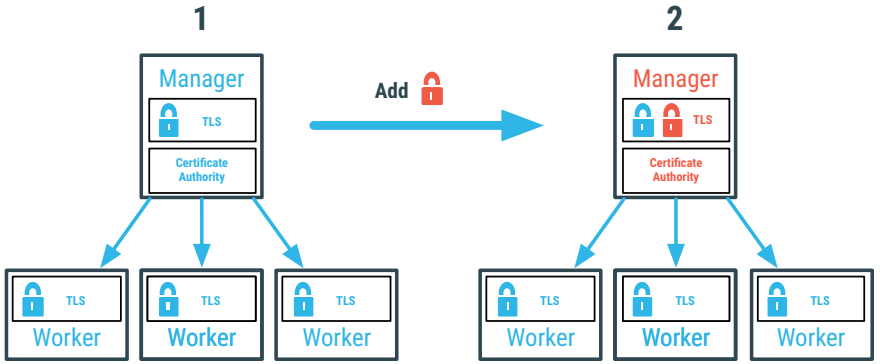
# swarmKit

Transparent
Root Rotation



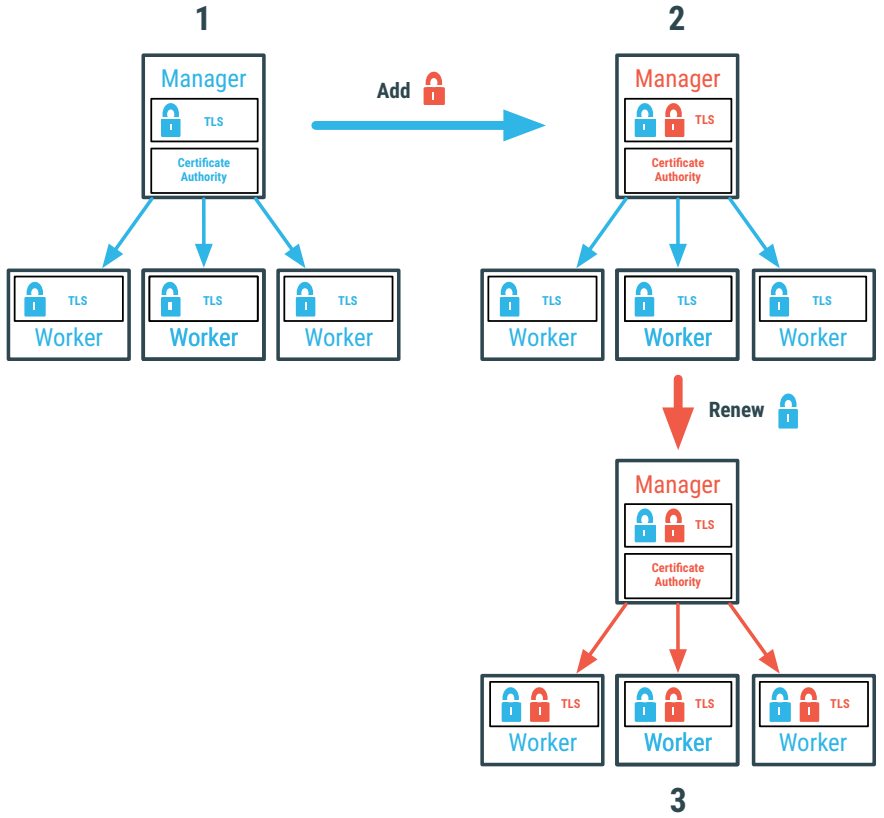
# swarmKit

## Transparent Root Rotation



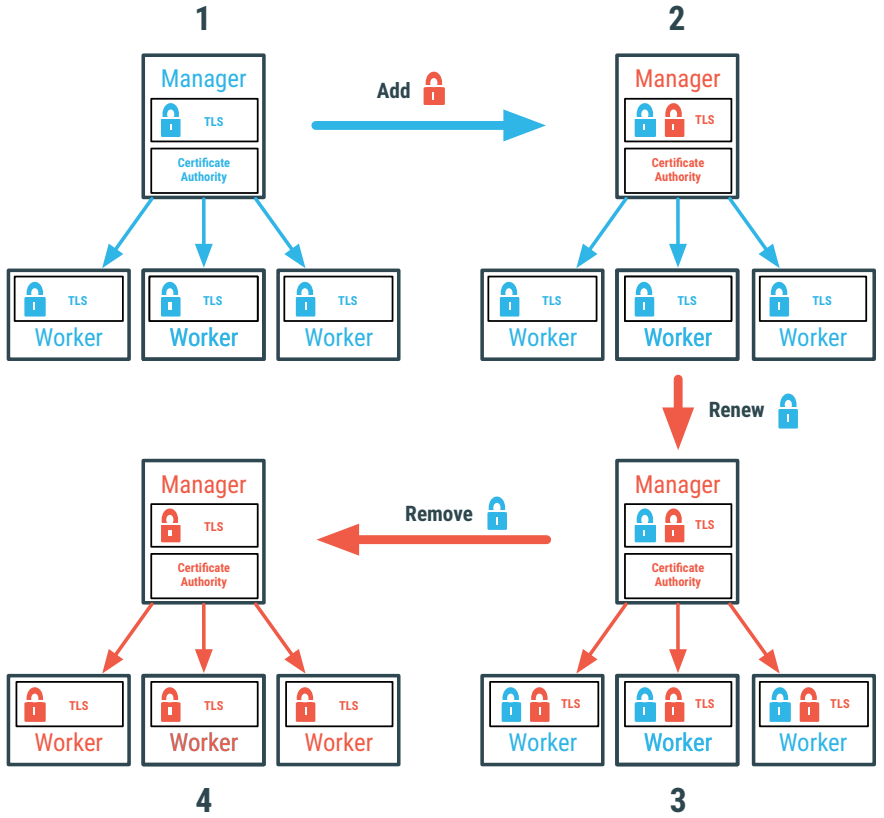
# swarmKit

## Transparent Root Rotation



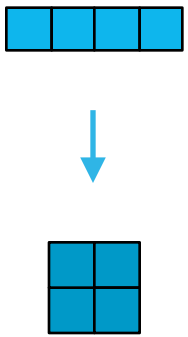
# swarmKit

## Transparent Root Rotation

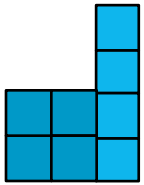


# Bringing it all together

# Notary for Docker image name resolution

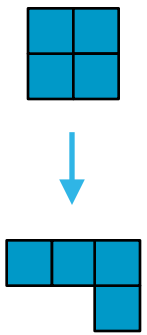


# Notary for Docker image name resolution

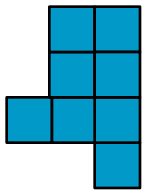


# Cryptographically Verified Pulls

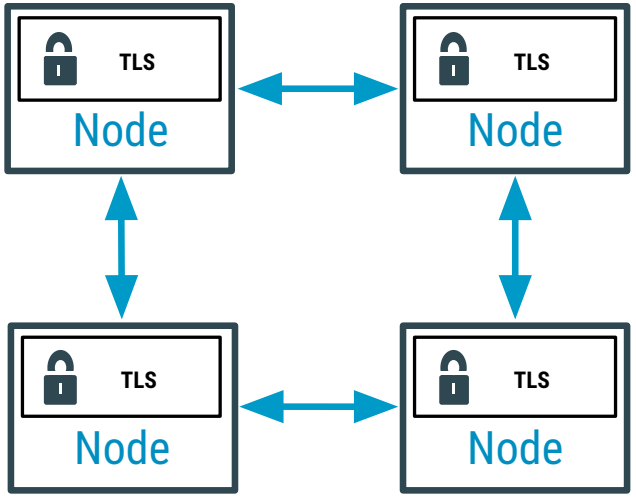
# swarmKit delivered Docker containers



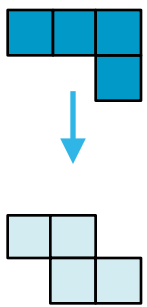
# swarmKit delivered Docker containers



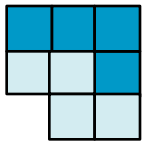
Authorized,
Authenticated,
Encrypted delivery
of Resources



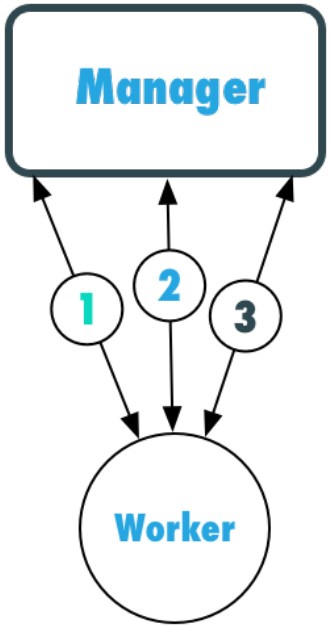
# infraKit for swarmKit Bootstrap



# infraKit for swarmKit Bootstrap

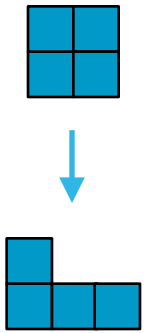


# Secure Node Cluster Introduction

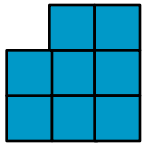



1. Retrieve and validate Root CA Public key material.
2. Submit new CSR along with secret token.
3. Retrieve the signed certificate.

# linuxKit as the base OS builder

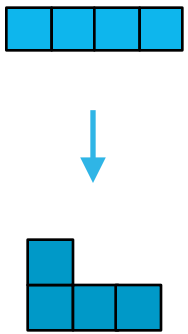


# linuxKit as the base OS builder

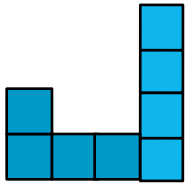


# Hardened Configuration

# Notary for secure dependency resolution

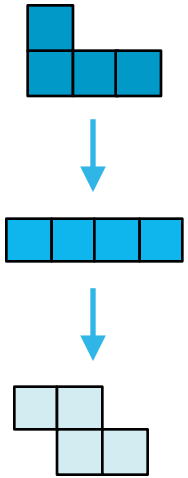


# Notary for secure dependency resolution

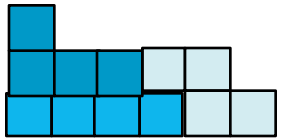


# Cryptographically Verified Build

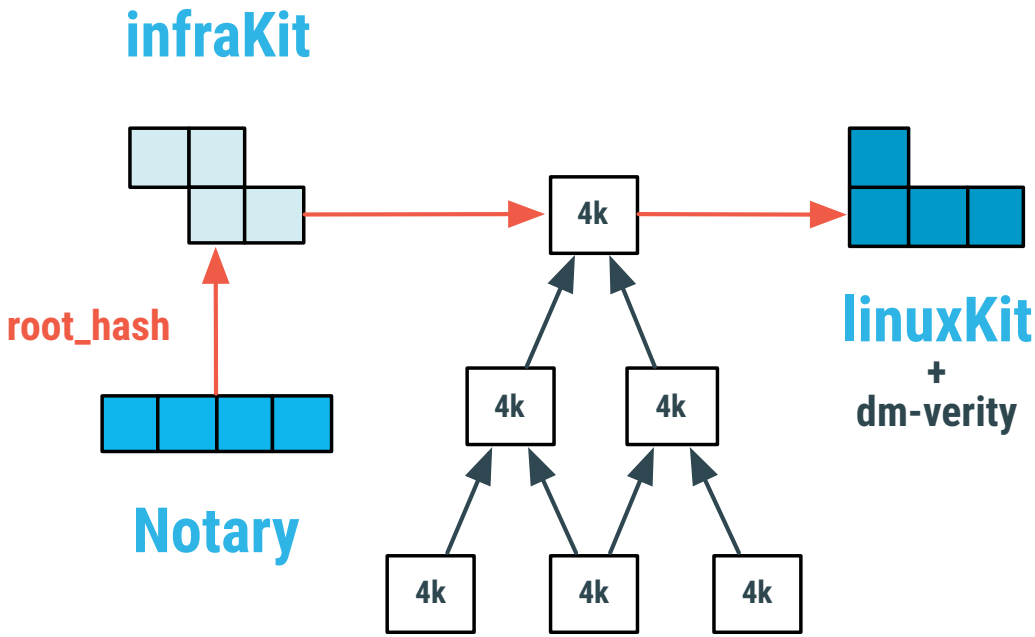
# infraKit plus Notary for trusted OS Provisioning

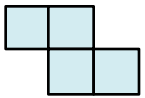


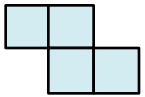
# infraKit plus Notary for trusted OS Provisioning

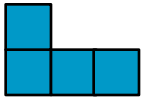


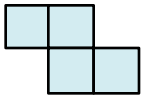
# Cryptographically Verified Boot

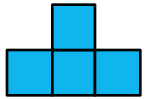


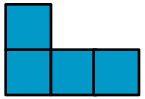


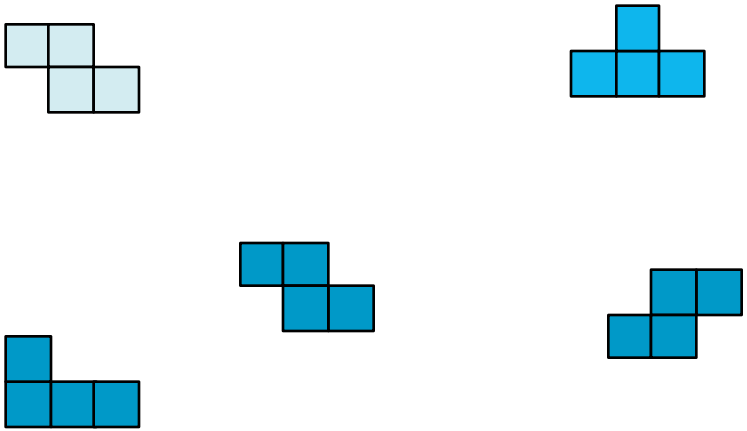


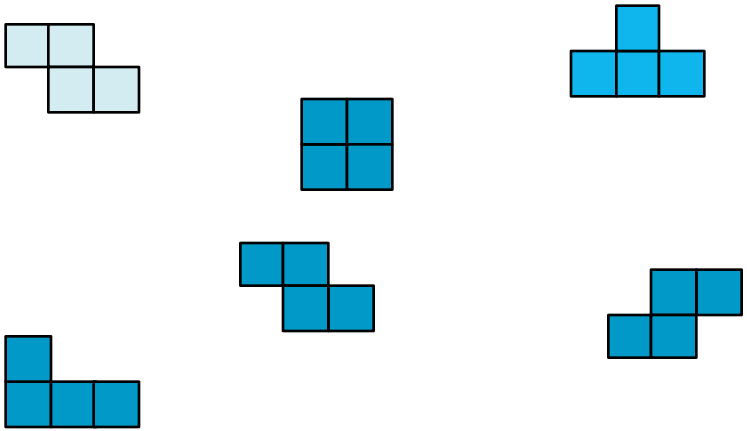


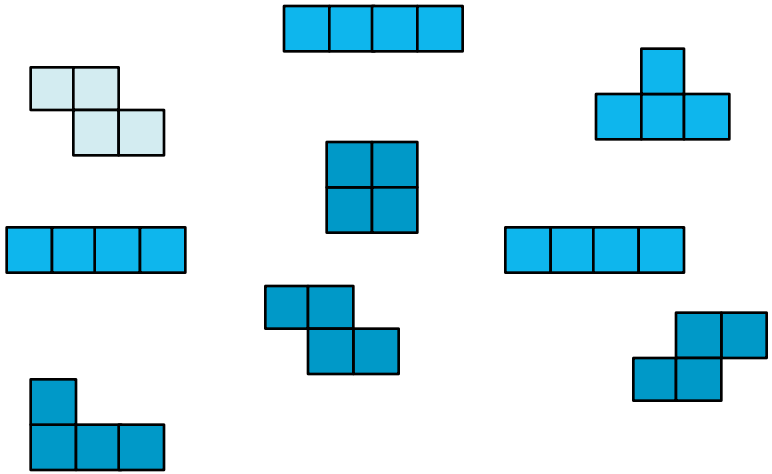


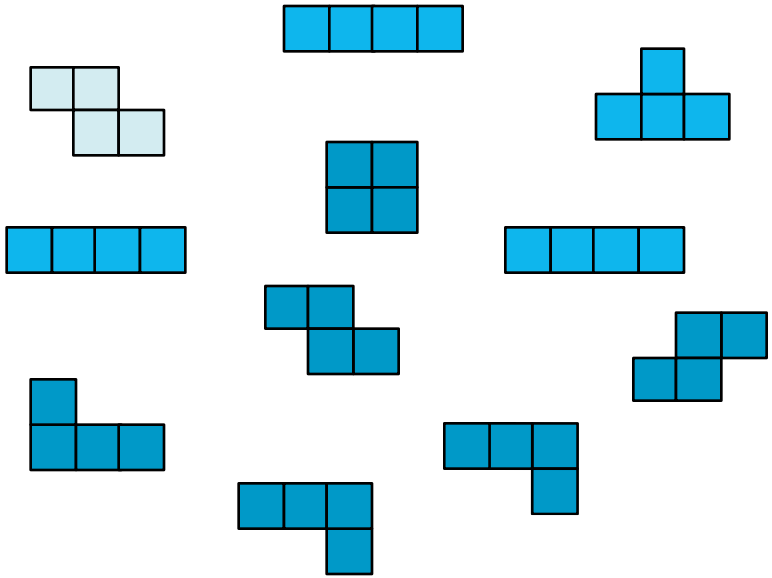


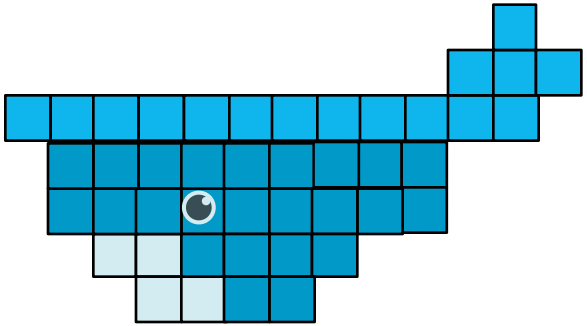


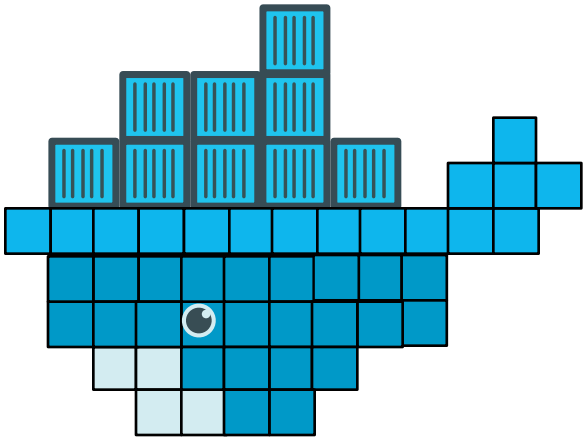












# Thank you!