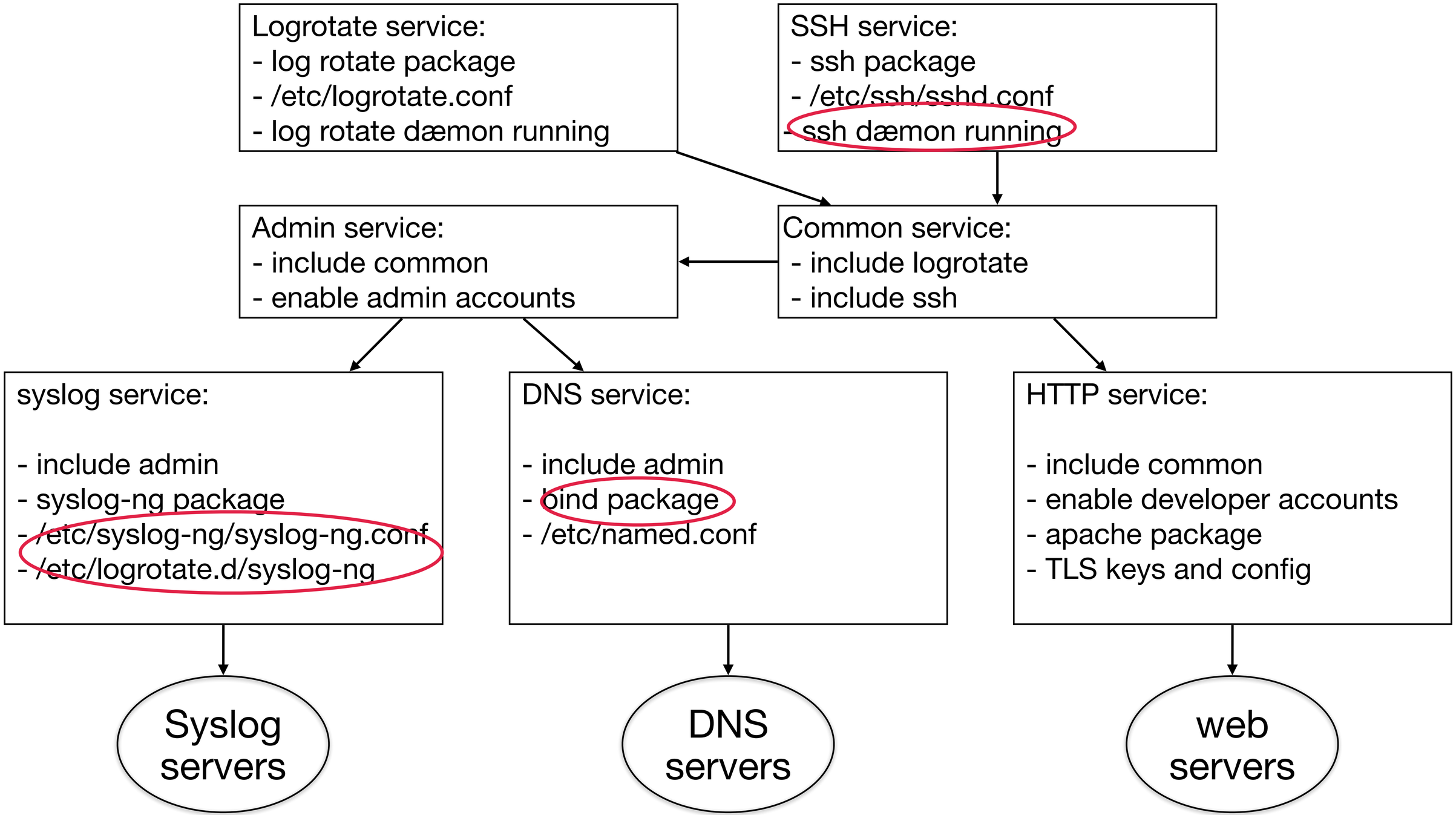
System Administration

Week 10, Segment 2
Configuration Management, Part II

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/615/

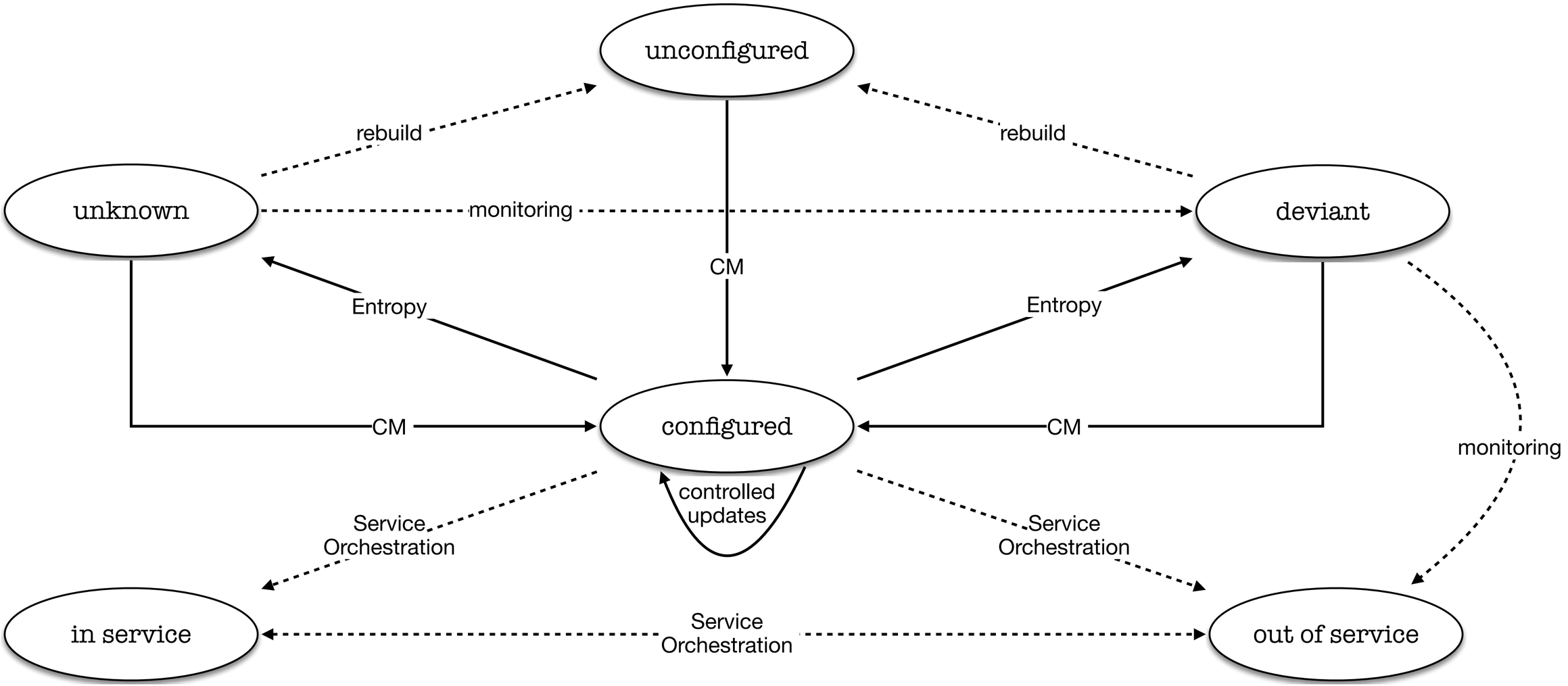

Logrotate service:
- log rotate package
- /etc/logrotate.conf
- log rotate dæmon running
SSH service:
- ssh package
- /etc/ssh/sshd.conf
- ssh dæmon running
Admin service:
- include common
- enable admin accounts
Common service:
- include logrotate
- include ssh
syslog service:
- include admin
- syslog-ng package
- /etc/syslog-ng/syslog-ng.conf
- /etc/logrotate.d/syslog-ng
DNS service:
- include admin
- bind package
- /etc/named.conf
HTTP service:
- include common
- enable developer accounts
- apache package
- TLS keys and config
Syslog servers
DNS servers
web servers

One more layer of abstraction...

The objective of a CM system is not
to *make changes* on a system.

The objective of a CM system is to *assert state*.

CM States


unconfigured
rebuild
rebuild
unknown
monitoring
deviant
CM
Entropy
Entropy
CM
configured
CM
monitoring
controlled
updates
Service
Orchestration
Service
Orchestration
in service
Service
Orchestration
out of service

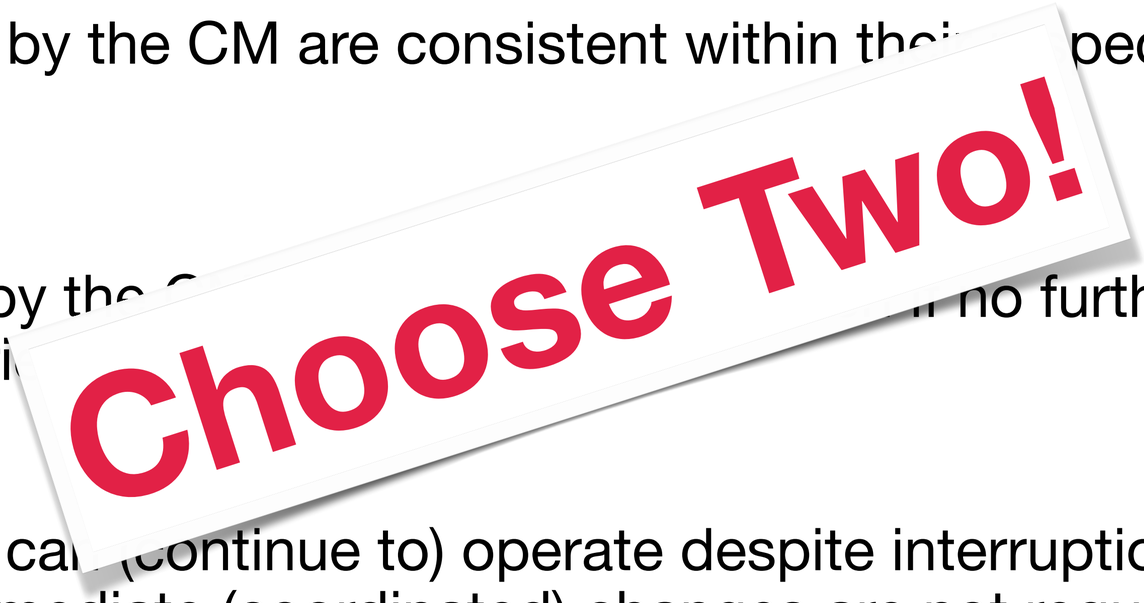
Distributed Systems

CM systems are distributed systems. As such, they are subject to the CAP Theorem:

Consistency: all systems managed by the CM are consistent within their respective service definition.

Availability: the services managed by the CM are available if no further updates or change sets can be retrieved.

Partition tolerance: the CM system can (continue to) operate despite interruptions between its components; e.g. intermediate (coordinated) changes are not required.



Idempotence

CM systems *assert state*. For this, all operations must be *idempotent*.

$$f(f(x)) \equiv f(x)$$

$$||-1|| \equiv |-1|$$

`$ rm resolv.conf`	✅
`$ echo "nameserver 192.168.0.1" > resolv.conf`	✅
`$ echo "nameserver 192.168.0.2" >> resolv.conf`	❌
`$ chown root:wheel resolv.conf`	✅
`$ chmod 0644 resolv.conf`	✅
`$ yum install frozzle`	❌
`$ yum install frozzle-1.2.3`	❓

Convergence and Eventual Consistency

Note: while idempotence enables self-healing and may allow you to not keep state, it does not guarantee efficiency!

CM systems should ensure changes are:

1. idempotent (well, that part's on you)
2. only applied if needed
3. eventually consistent

This often requires complexity, coordination with and awareness of other systems.

Service Orchestration has developed as a separate, related discipline to help address this.

CMs configure complex systems

- CM systems are complex themselves.
- CM systems are inherently trusted.
- CM systems can break everything. To the degree that you can’t unbreak things afterwards.

Consider:

- staged rollout of change sets
- automated error detection and rollback
- self-healing properties
- authentication and privilege

CM Requirements

- software installation
- service management / supervising
- file permissions / ownership
- static files
- host-specific data

- command-execution
- data collection

Configuration Management Overlap

Your configuration management system provides or enables:

- a remote command execution agent
- a reporting agent
- a reporting infrastructure
- role-based actions and visibility

Overlapping information security related tasks:

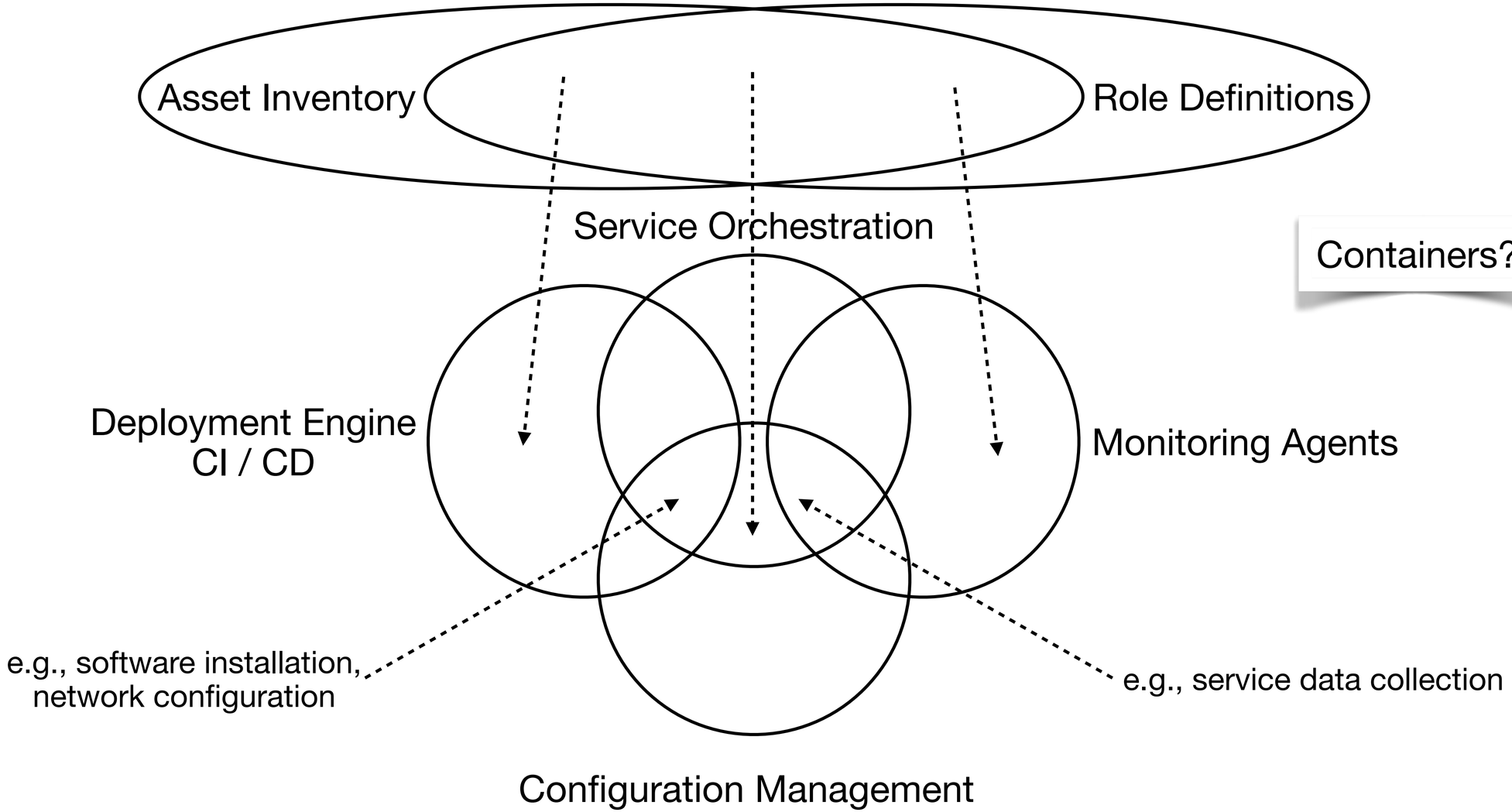
- detection of deviation of known state
- integrity checks and intrusion detection
- patch management
- automated quarantine

Configuration Management Overlap

Configuration Management overlaps with numerous other areas:

- software deployment (base OS, application packages, …)
- monitoring (central reporting and ad-hoc data collection, …)
- revision control and audit logs (CM changes are code changes!)
- compliance enforcement (e.g., baseline configurations)
- ...

Configuration Management Overlap



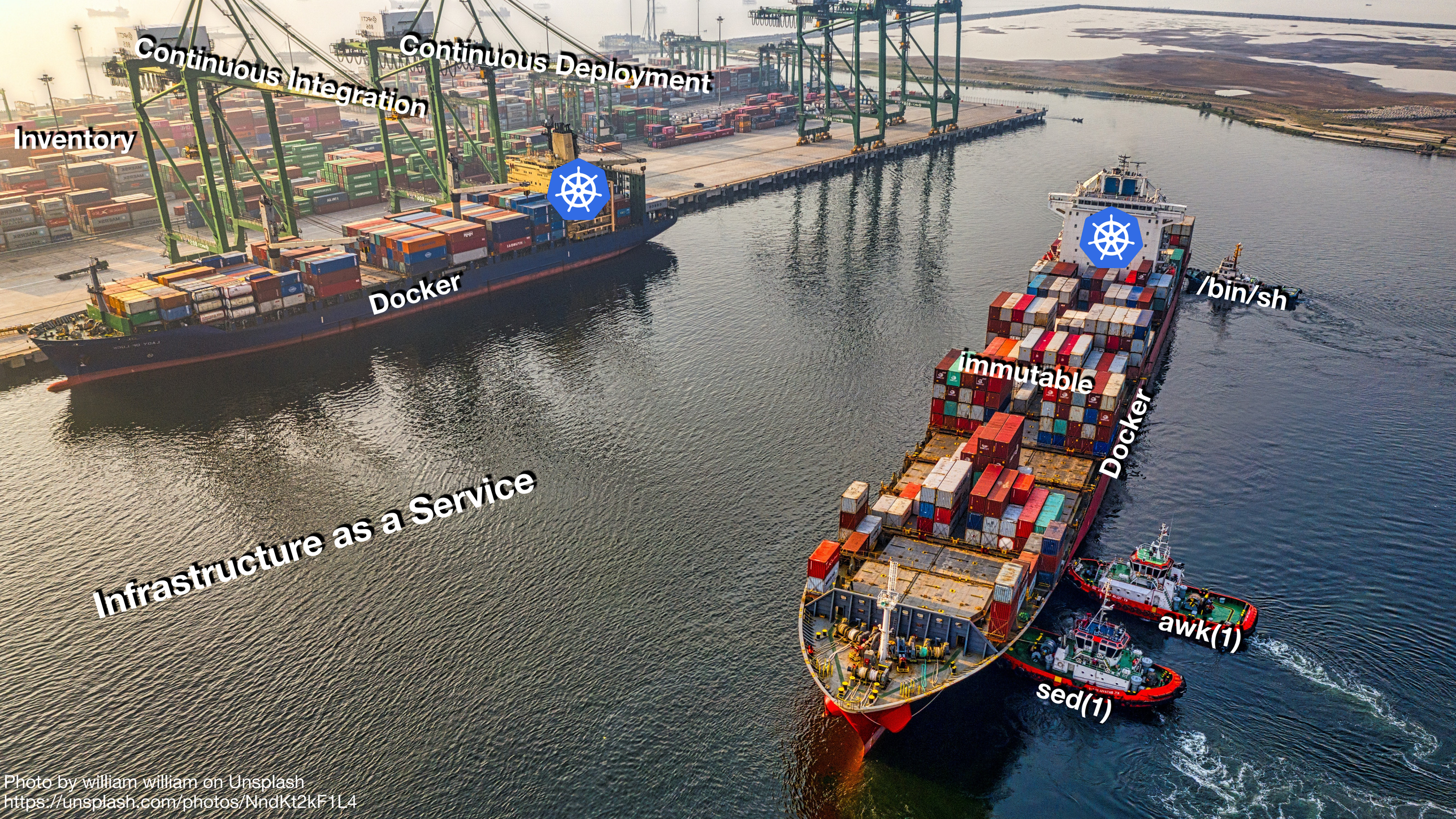


Photo by william william on Unsplash
https://unsplash.com/photos/NndKt2kF1L4

More than just servers...

Configuration Management is not just for servers. You also need to manage configurations for:

- desktops
- mobile clients
- network equipment
- storage devices
- load balancers
- ...

Core concepts underlying Configuration Management:

- abstract services from the hosts or systems they run on
- move away from fragile systems managed by hand to reproducible, exchangeable instantiations of a service definition
- focus not on applying changes, but on state assertion
- as distributed systems, CM systems are subject to the CAP theorem
- CM systems require idempotence, state convergence with eventual consistence
- overlap with CI/CD, Deployment Engines, Service Orchestration, enforcement and monitoring agents
- CM underlies and enables immutable containers, Infrastructure as Code, IaaS

Links

- http://markburgess.org/blog_cap.html
- http://markburgess.org/blog_cap2.html
- https://blog.engineyard.com/pets-vs-cattle
- https://en.wikipedia.org/wiki/CAP_theorem
- https://en.wikipedia.org/wiki/Idempotence
- https://en.wikipedia.org/wiki/Eventual_consistency
- https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing