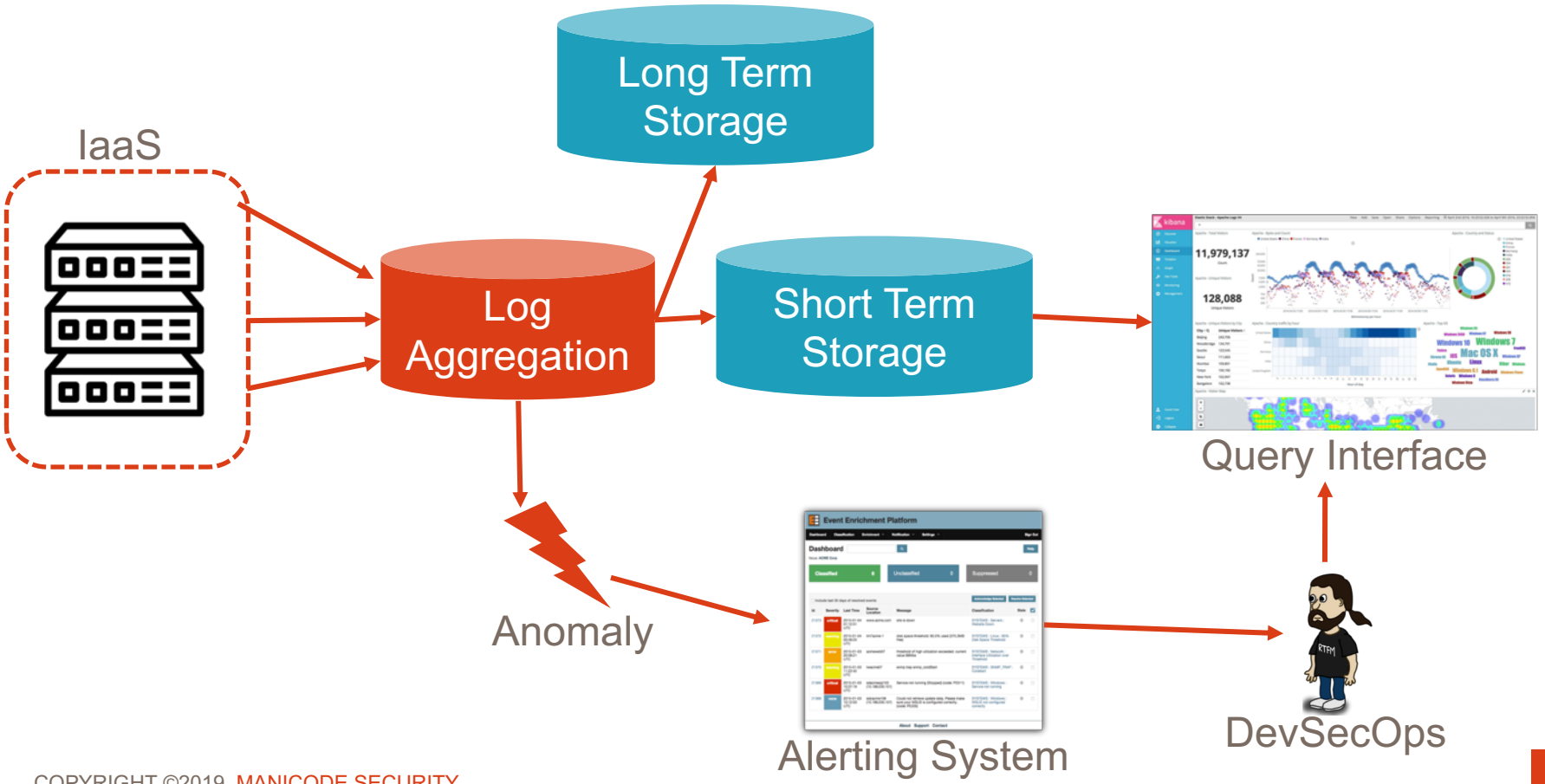
Logging, Monitoring, and Alerting

- Logs are a part of daily life in the DevOps world
- In security, we focus on particular logs to detect security anomalies and for forensic capabilities
- A basic logging pipeline can be shared between Developers, Operations, and Security teams:
  - **Log Aggregation:** Used to ingest logs from systems, applications, network components, etc.
  - **Long Term Storage:** Filesystem which retains logs for an extended period of time. Good for forensics or breach investigation.
  - **Short Term Storage**: Filesystem or DB which stores logs to be queried quickly and easily.
  - **Alerting:** Anomaly detection system which is responsible for sending alerts to teams when a deviation occurs

Logging and Monitoring Pipeline



Infrastructure as Code

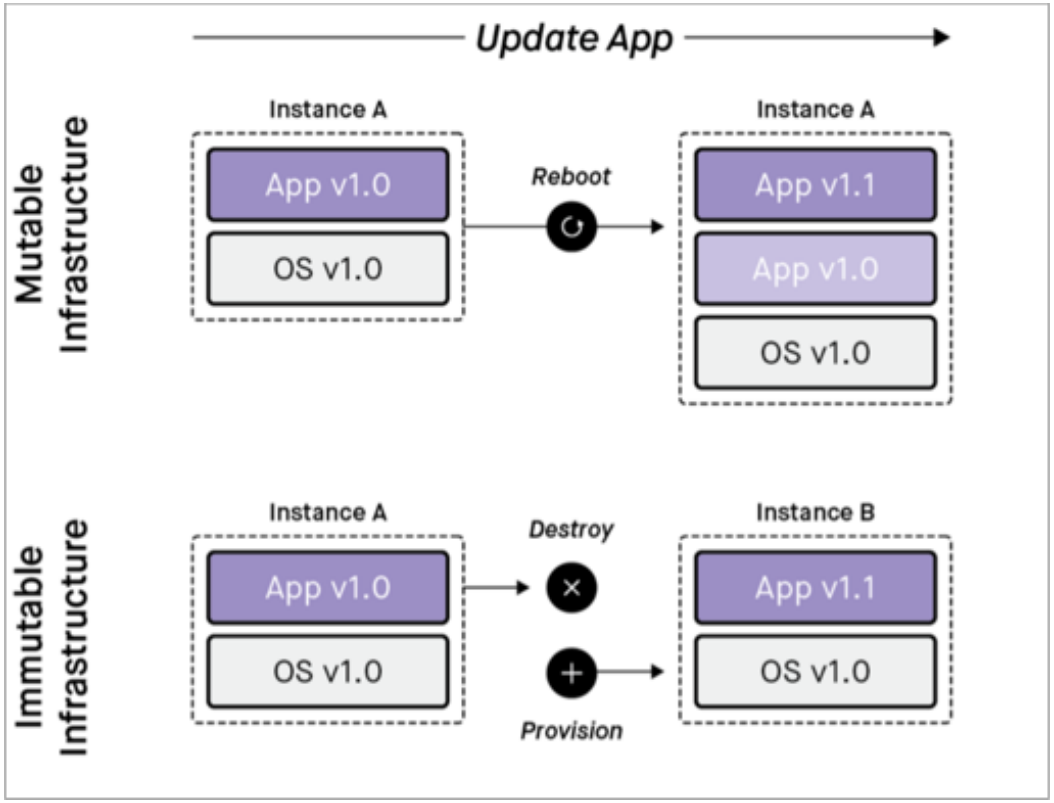
Building Infrastructure

- Is ***your*** infrastructure…
  - Self documenting?
  - Version controlled?
  - Capable of continuous delivery?
  - Integration tested?
  - Immutable?

Remember: ”It’s all software"

Immutable Infrastructure

"Immutable infrastructure is compromised of components which are replaced during deployment rather than being updated in place"

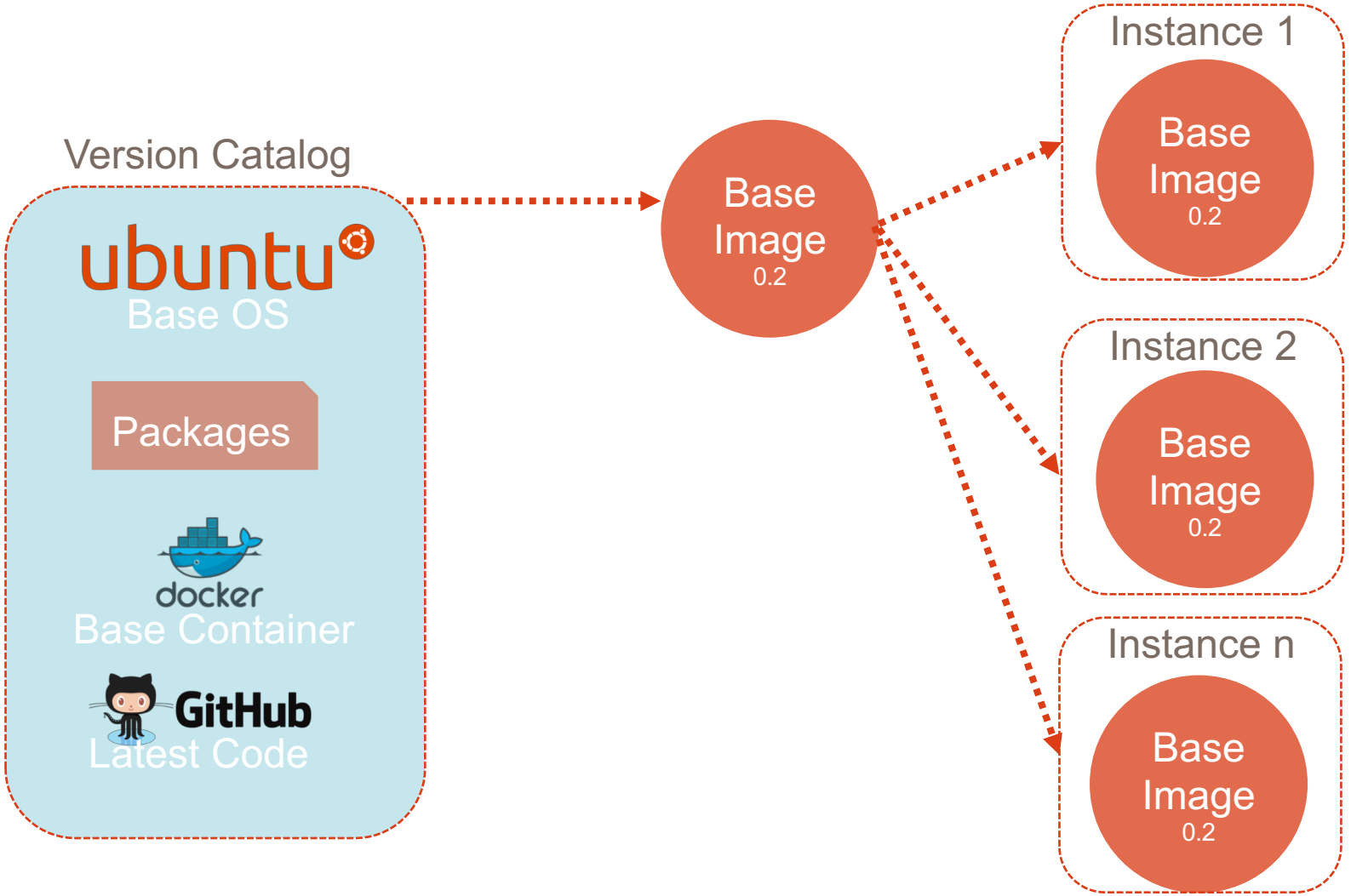


Security and Immutable Infrastructure

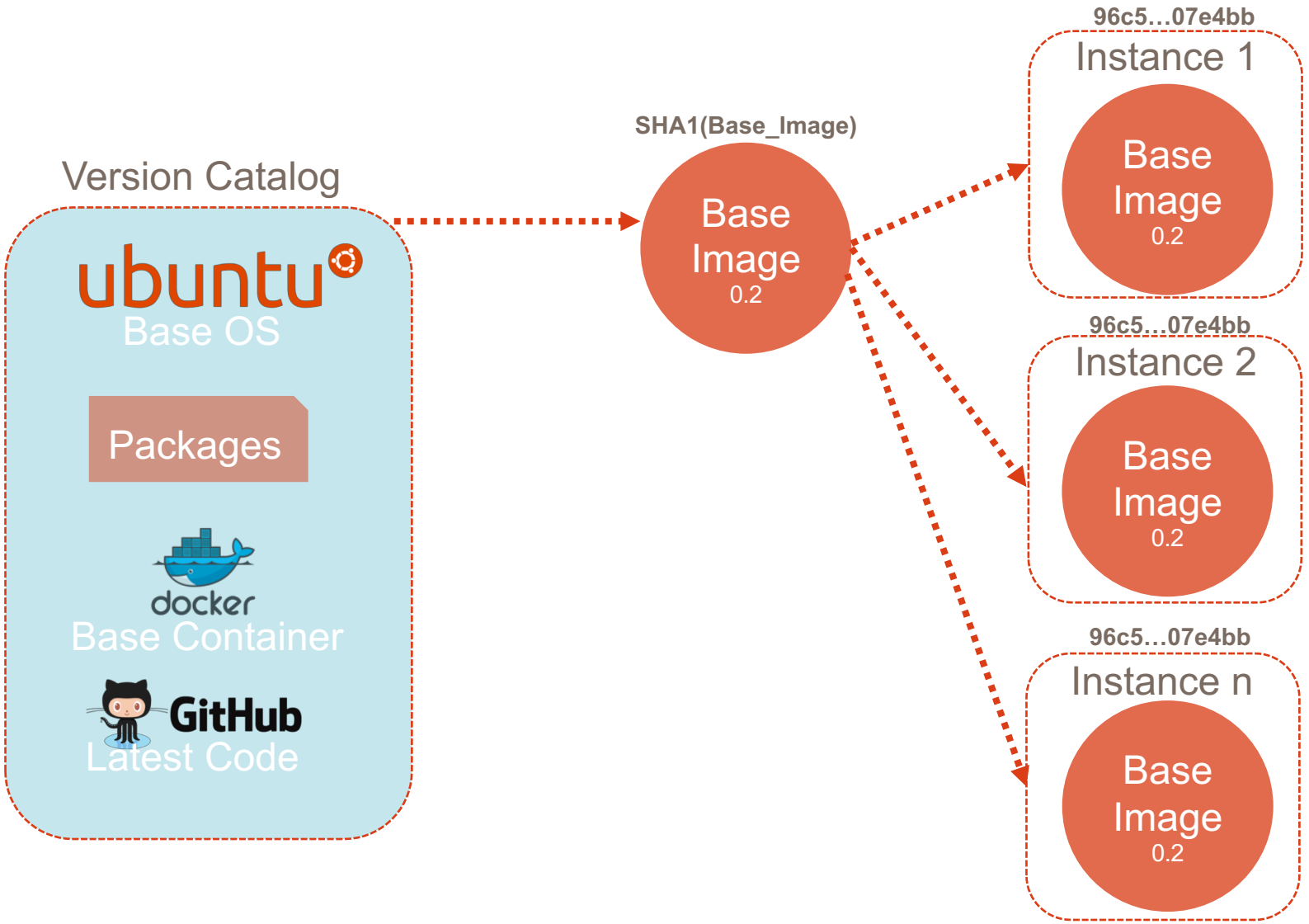
- An immutable infrastructure starts with a "Golden Image" in a version catalog
- Security teams have a central location to validate images as compliant and enforce OS hardening policies
- No more guesswork what is installed Automation can flag security anomalies vs. human intervention
- Tags help teams wrangle infrastructure

"Push Security to the Left"

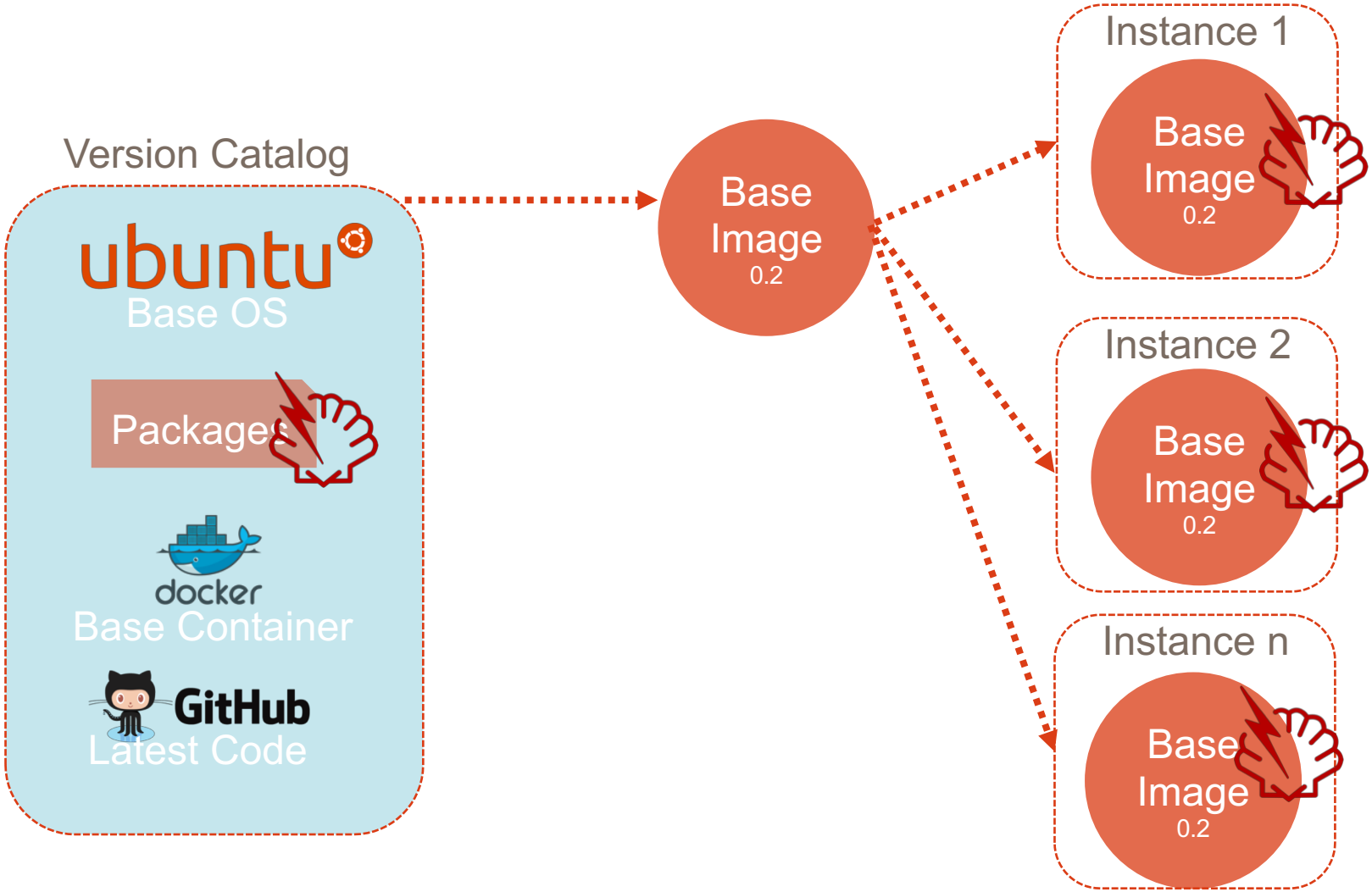
Simple Immutable Infrastructure



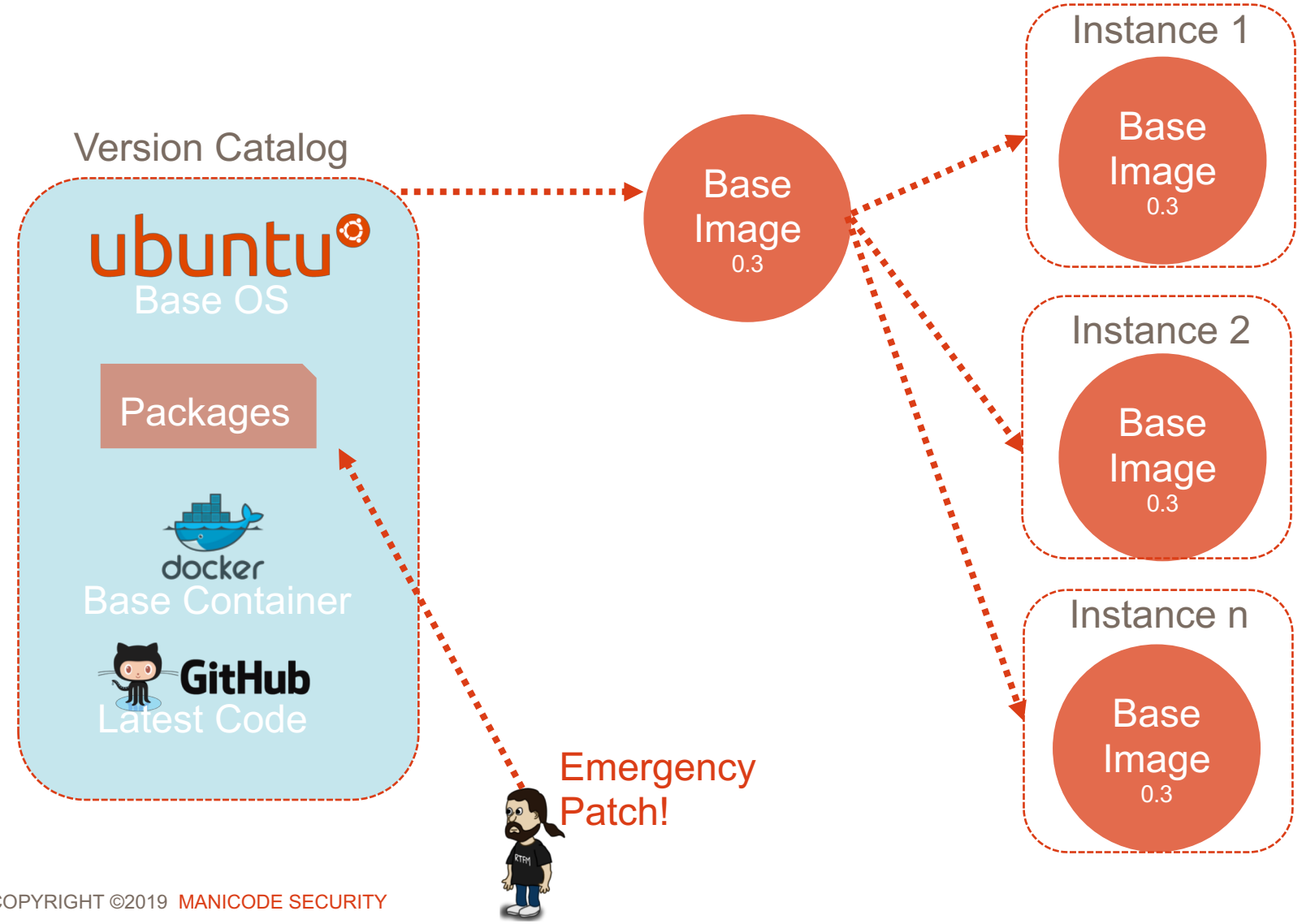
Proving Immutability



Shellshock?



Shellshock?

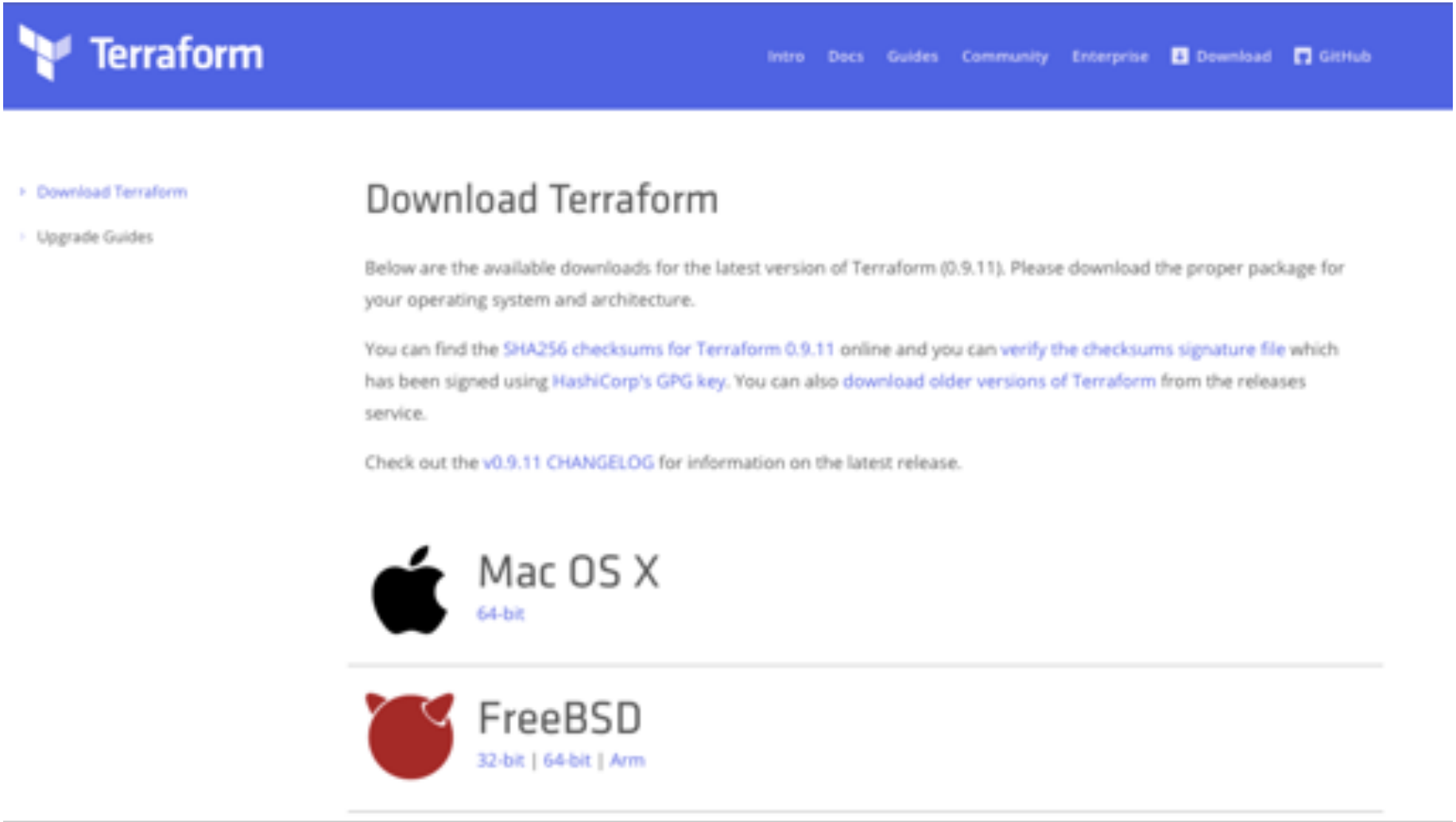


Cattle, not pets.

Security Wins

- Security team now has insight into the entire system
- Infrastructure is auditable and version controlled, just like source code
- Patching can be applied programmatically with a high level of certainty
- Alerting can be built for changes to specific areas of the infrastructure
  - A new firewall rule is created or deleted
  - Administrative user is created
  - New VPC rolled out
- Testing can occur much earlier in the pipeline

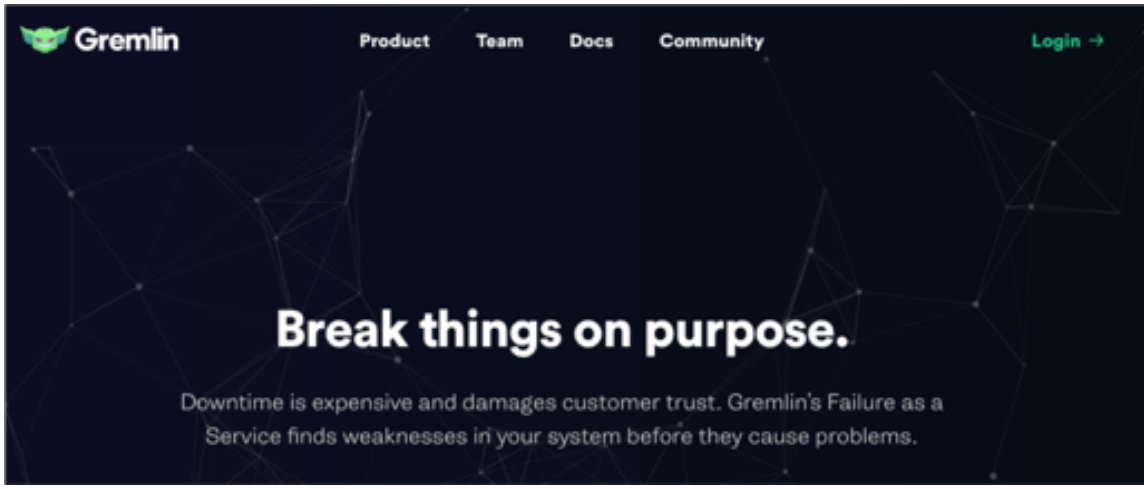
Infrastructure as Code - Terraform



Infrastructure as Code – K8s

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: my-site-ingress
  namespace: my-site-prod
  annotations:
    kubernetes.io/tls-acme: "true"
    kubernetes.io/ingress.class: "gce"
    kubernetes.io/ingress.global-static-ip-name: my-site-external-ip
spec:
  tls:
  - hosts:
    - api.my.site
    - my.site
    secretName: my-site-cert
  rules:
  - host: api.my.site
    http:
      paths:
      - path: /*
        backend:
          serviceName: app-api
          servicePort: 80
  - host: my.site
    http:
      paths:
      - path: /*
        backend:
          serviceName: my-site-prod
          servicePort: 80
```

”Chaos” Testing



Brief Introduction to Containers

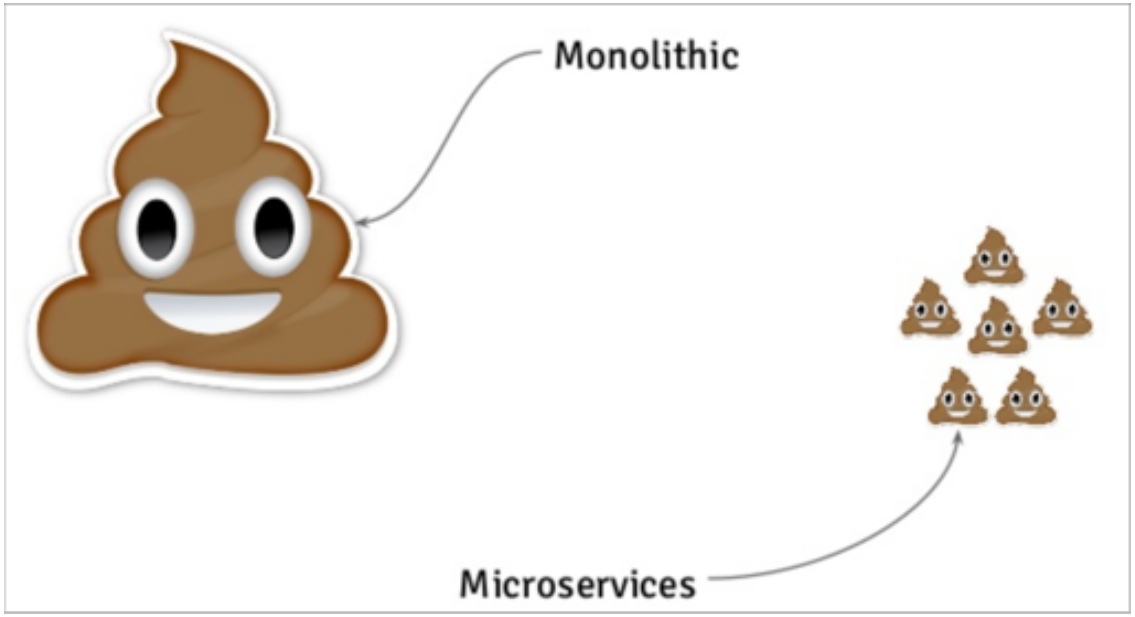
Containers, Containers, Containers, Containers…

Software Deployment is Changing

- Massive shift toward cloud computing
- Increased demand for application and infrastructure portability across environments
- Avoid vendor “lock in” when possible
- Increase in microservices AKA loosely coupled services

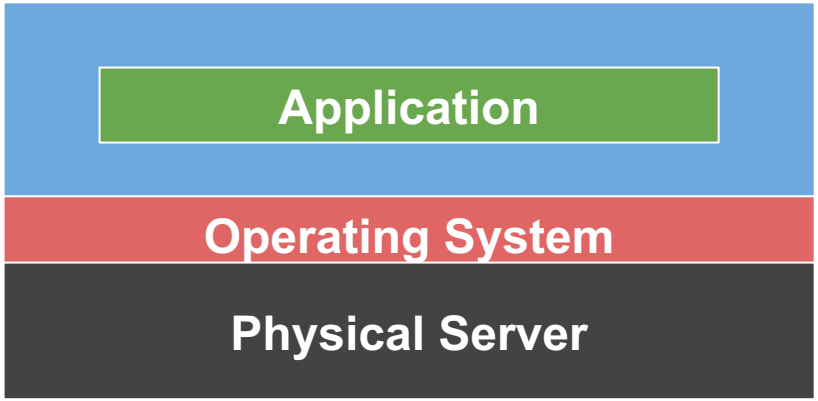
Modern Applications

- Breaking monolithic applications into smaller services offers several advantages:
  - Scale independently
  - Stateless
  - High Availability
  - API-Driven
  - Faster iteration times

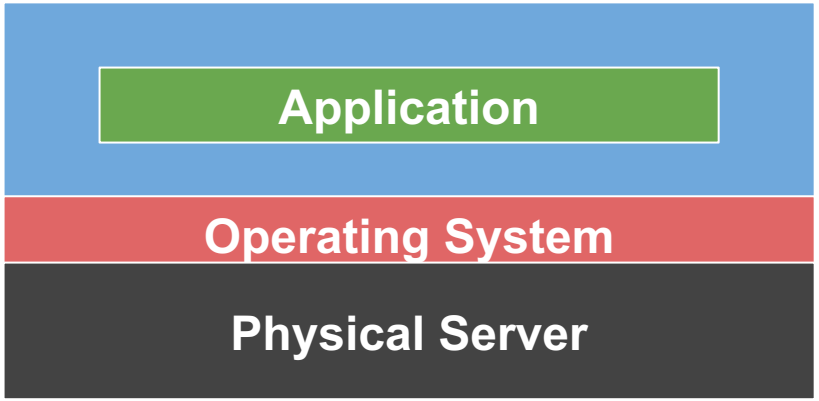


Issues with Modern Applications

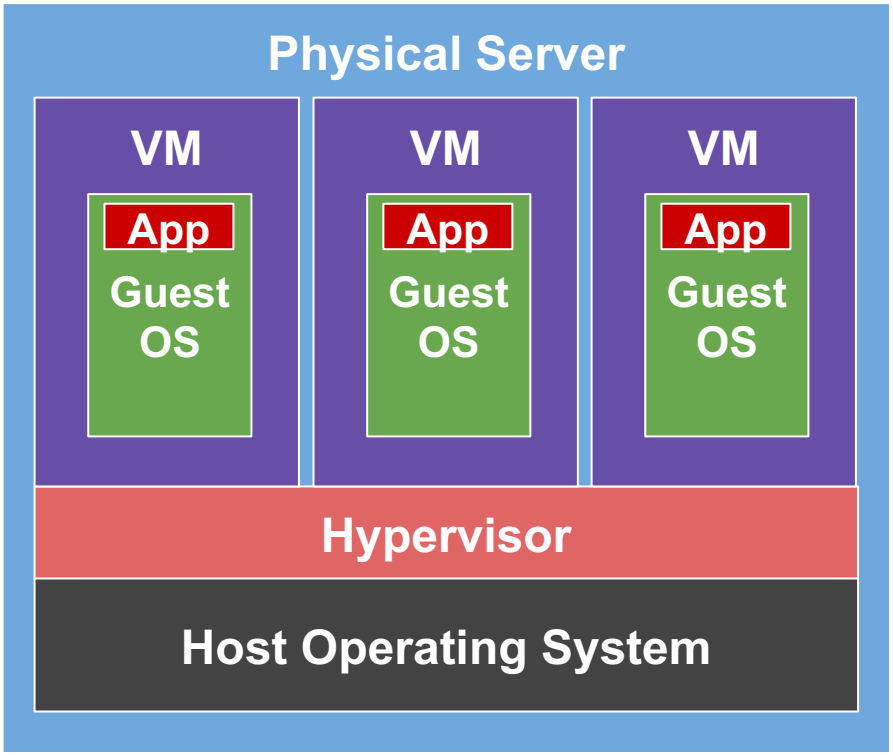
- Organizations often operate in an Ops vs. Dev vs. Sec world
- Applications and microservices are written in a variety of languages and frameworks
- Applications need to run on different technology stacks:
  - Virtual Machines
  - Windows Server
  - Bare Metal Servers
  - Cloud Environments
  - On-Prem Environments
  - Developer Laptops



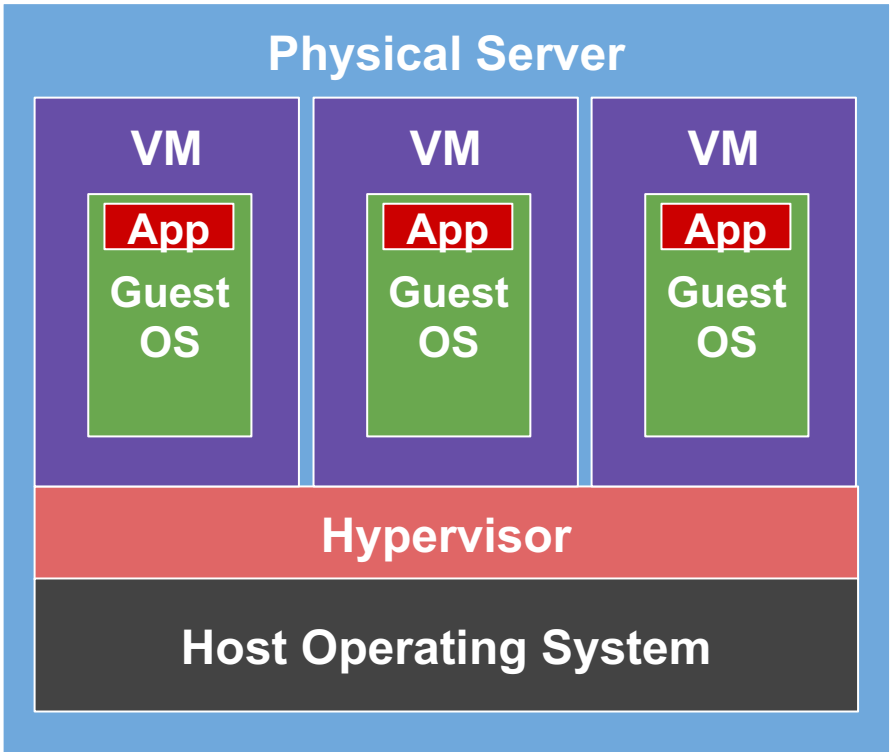
Physical Host



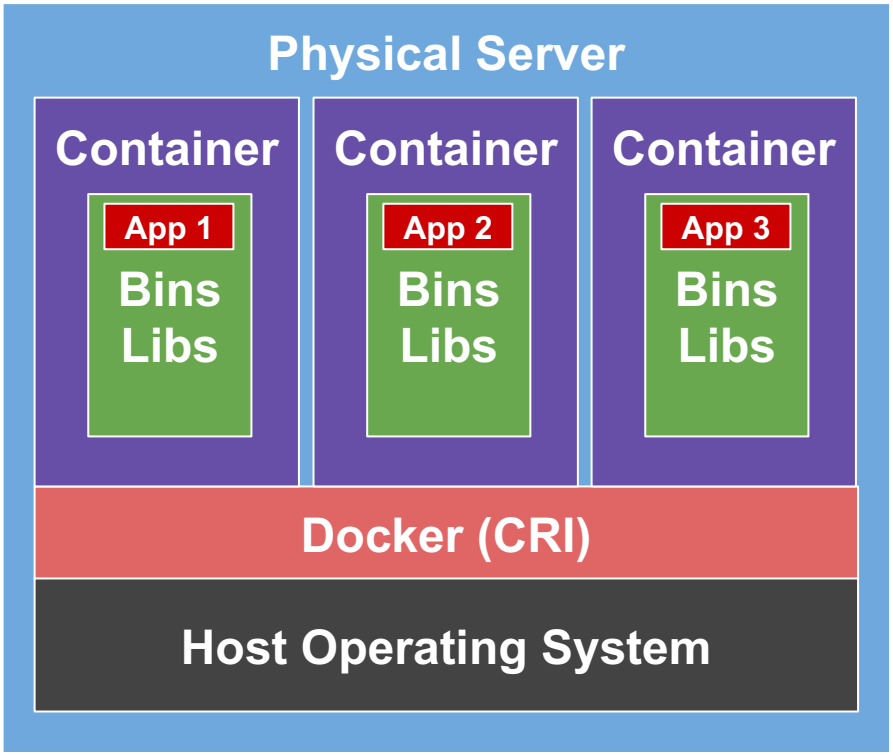
- One application per server
- Slow deployment times
- Low resource utilization
- Scaling challenges
- Migration challenges
- $$$
- Difficult to replicate locally



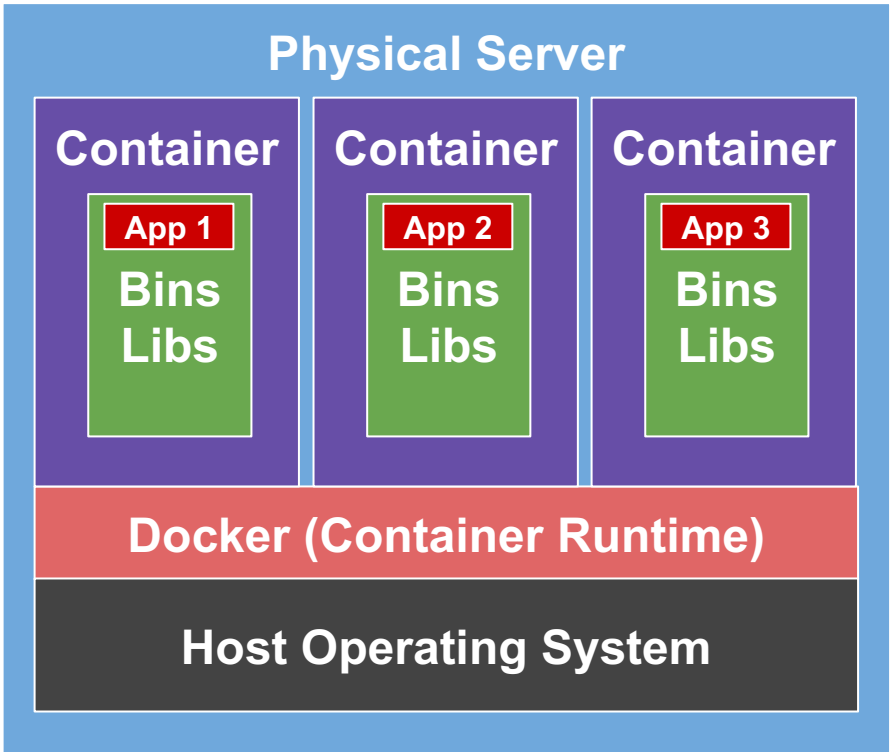
VM



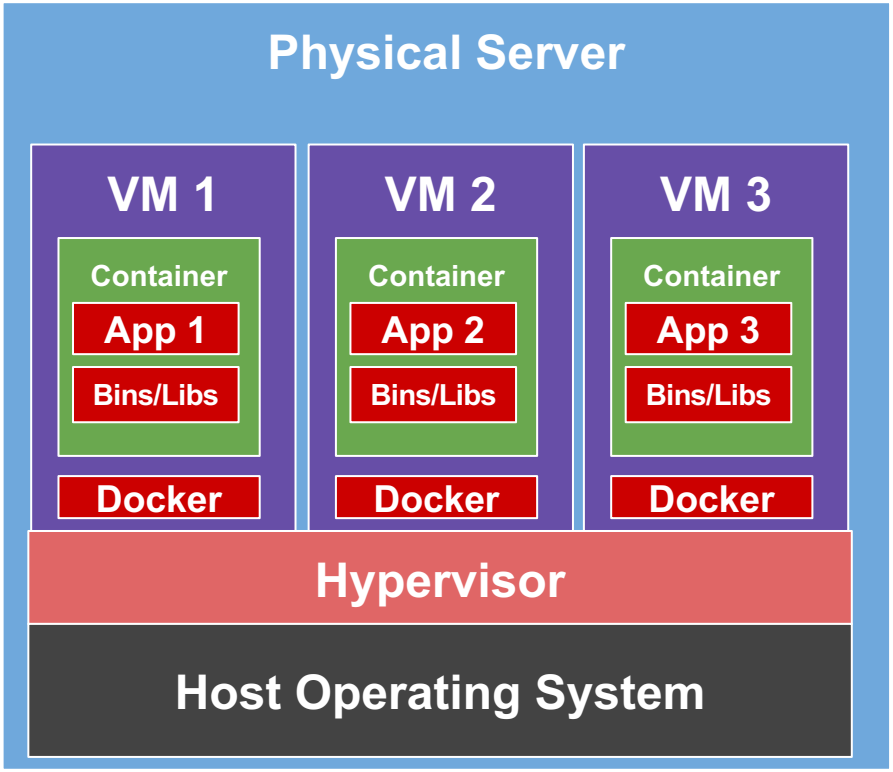
- One physical server and multiple applications
- Each application runs in a Virtual Machine
- Better resource utilization
- Easier to scale
- VMs live in the Cloud
- Still requires complete guest Operating Systems
- Application portability not guaranteed



Container



- Containers are an application layer construct
- VMs allow us to convert one physical machine into many servers
- No Operating System to boot (fast!)
- Most portable out of all options
- Less OS overhead using shared kernel model



Containers and VMs are Happy Together

It's been a pleasure.

jmesta@manicode.com

Jimmy Mesta Secure Coding Instructor *www.manicode.com*