EIGHTH ANNUAL LEADERSHIP EVENT

CS SUMMIT

Security solutions through collaboration

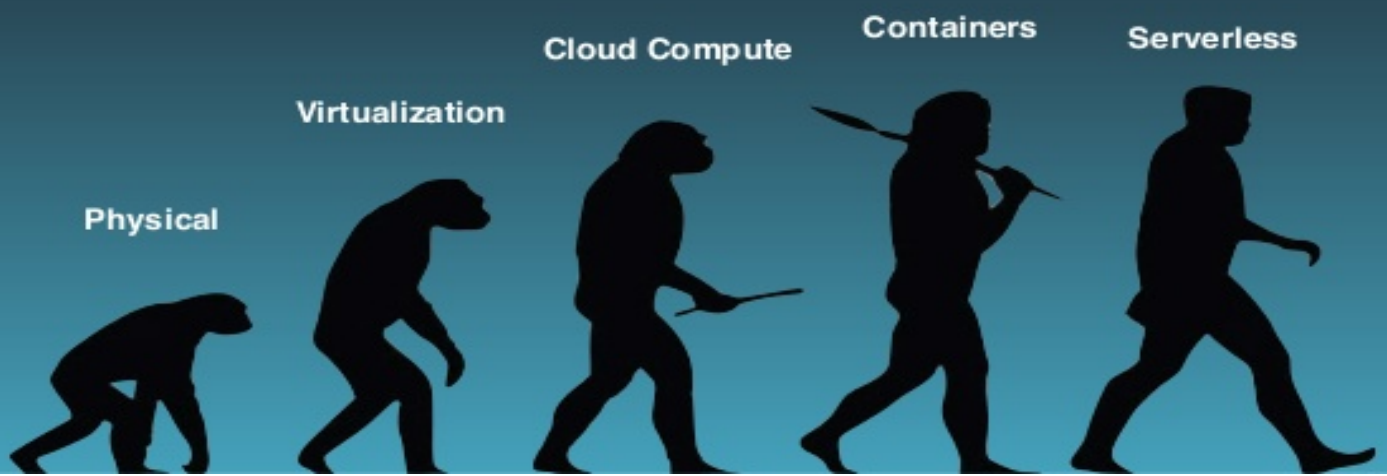
Why Security needs to be at the DevOps Table

Chris Perkins, Sr. Principal Security Technologist. Medtronic, PLC

Evolution of Compute and Development Methodologies

- Both processes evolved over time, with a similar pathway
- Technology advances helped drive speed in both arenas
- Pivotal concepts also helped “break norms”

The Evolution of Compute

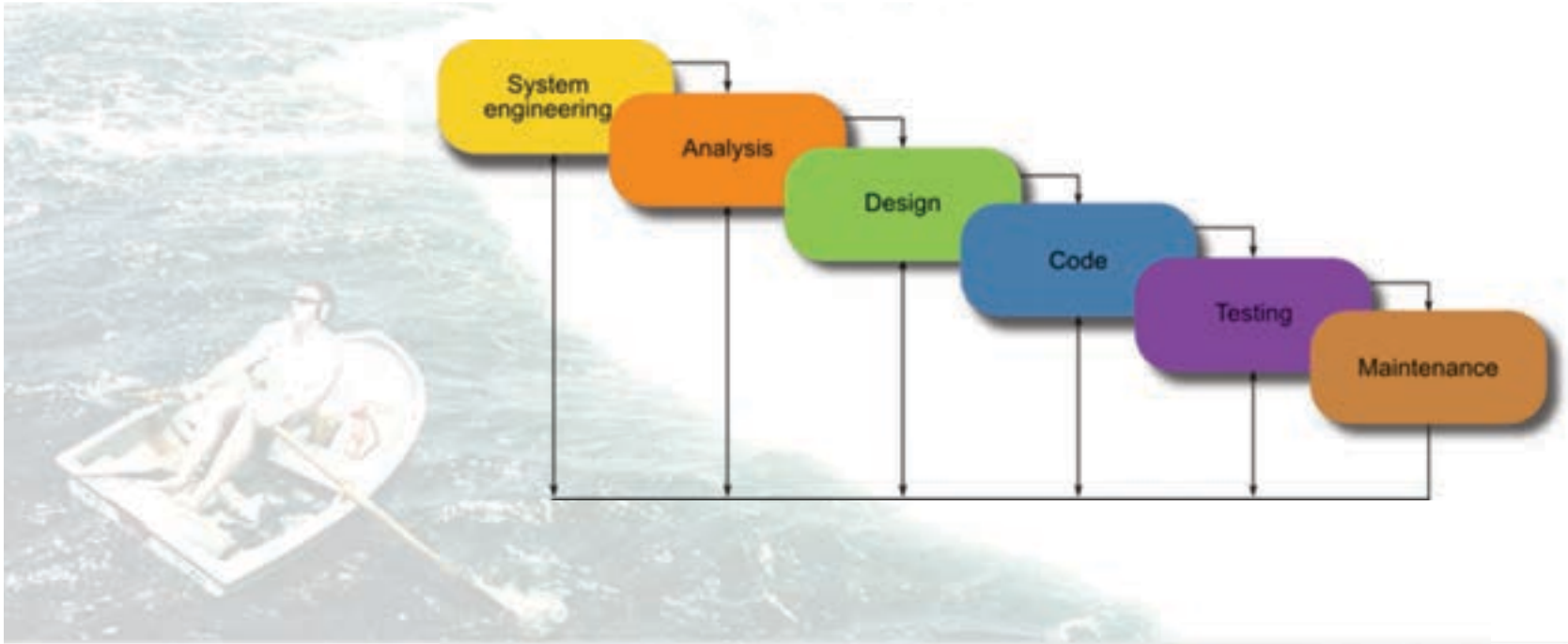


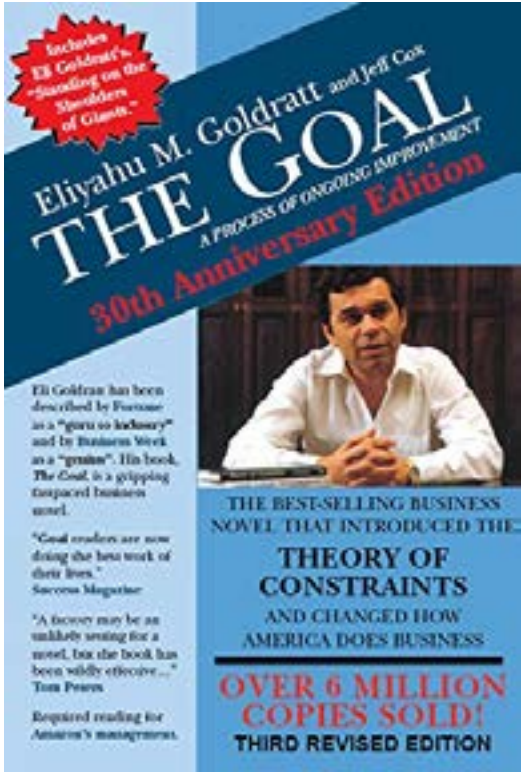

Software Development Evolution
Waterfall
Agile
DevOps
www.heliossolutions.in

In the Beginning..

- Each application was “racked and stacked”
- Each box housed a dedicated database, dedicated service
  - Web Services
  - Email
  - Authentication

Waterfall

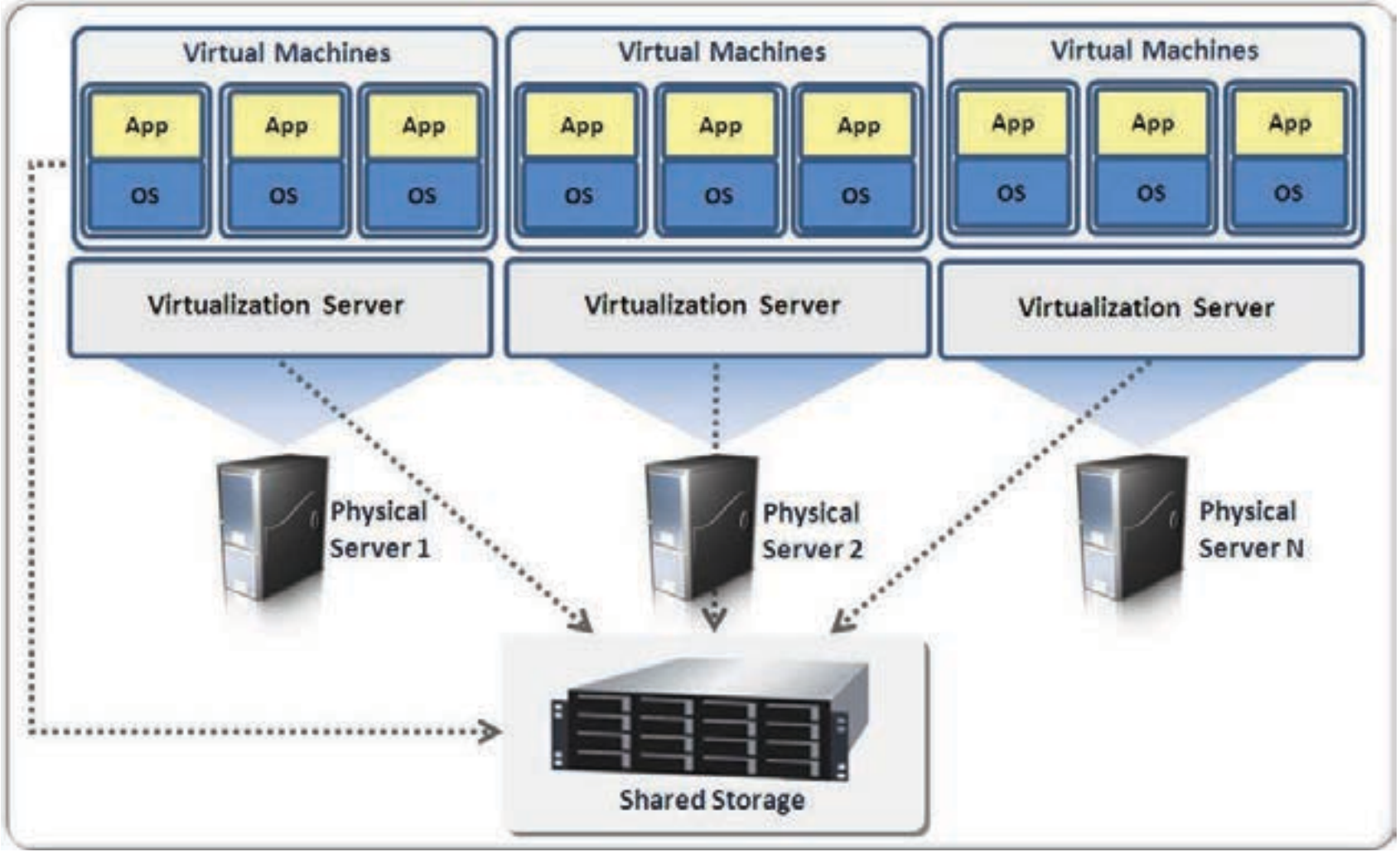

System engineering
Analysis
Design
Code
Testing
Maintenance



- Introduced the Theory of Constraints (TOC) concept
- Process of Ongoing Improvement
- Critical Chain Project Management (CCPM)
- First published in 1984

Then came Virtualization

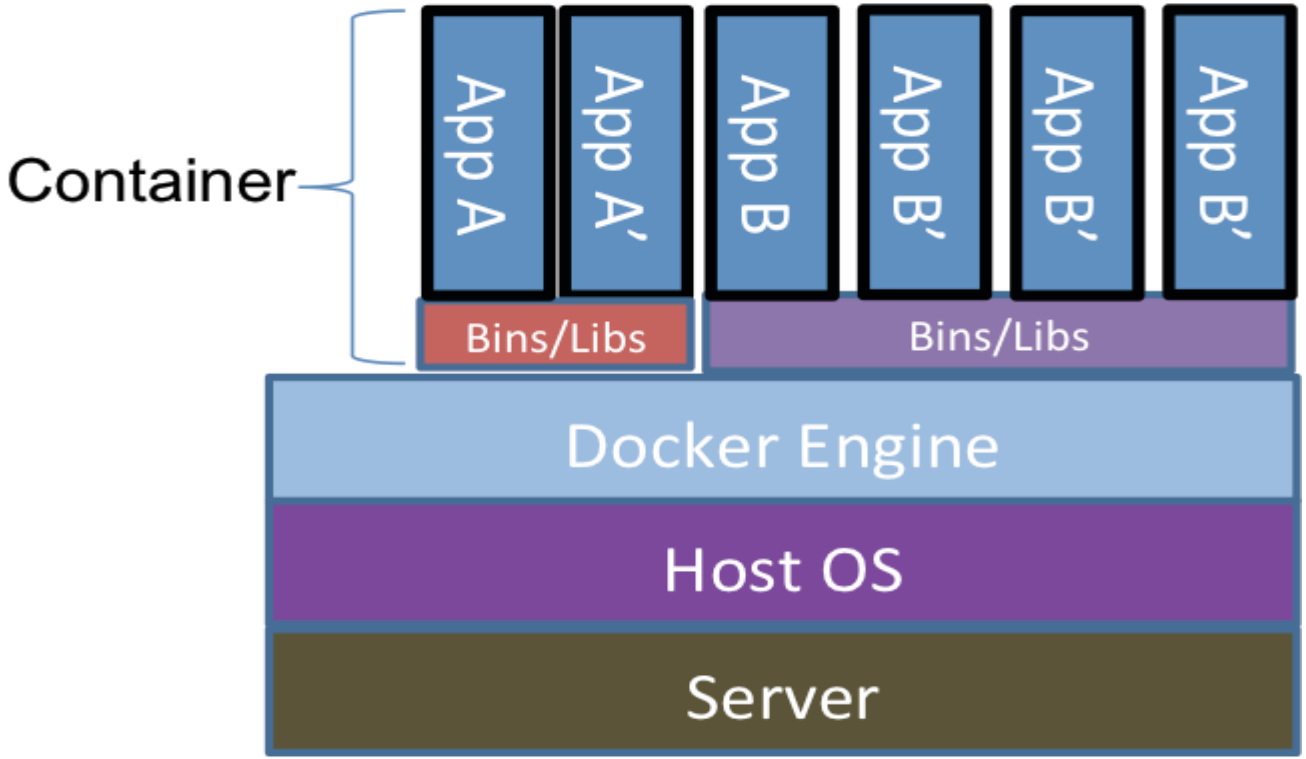
- The process of creating logical computing resources from available physical resources
- Layer of abstraction between workloads and the underlying physical hardware via Hypervisor
- Allowed for many to one, physical server usage optimized


Virtual Machines
App
OS
Virtualization Server
Physical Server 1
Physical Server 2
Physical Server N
Shared Storage

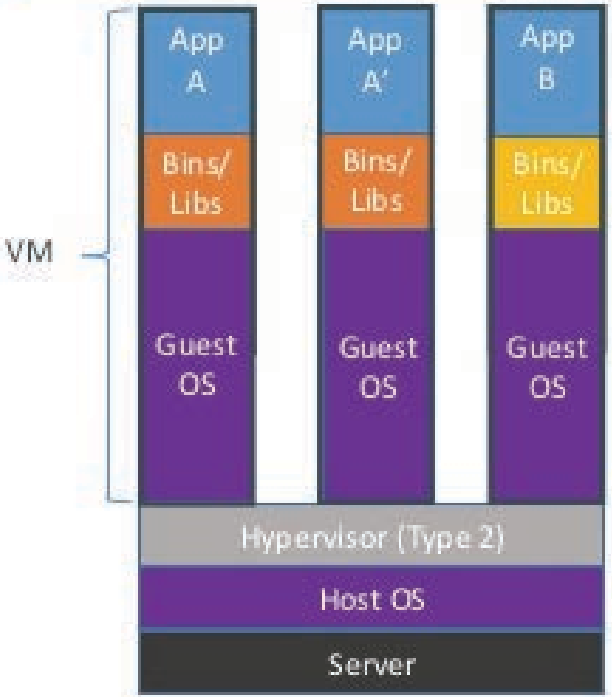
Agile


Agile
Development
1) Requirements
2) Plan
3) Design
4) Develop
5) Release
6) Track & Monitor

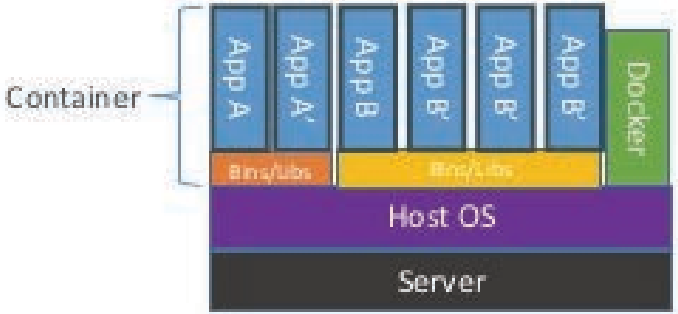
Containers continued the concepts



Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



http://www.theplatform.net/2015/03/23/hpc-schedulers-snap-to-docker/


WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

The Phoenix Project
3 Ways of DevOps

Strategies for Improving Operations

Container Threat Modeling

S.T.R.I.D.E

- **S**poofing: Attacker can prove that they are an authorized system of the user
- **T**ampering: Attacker can successfully add, modify and delete data
- **R**epudiation: Attacker can deny or make it impossible to prove their delinquency
- **I**nformation Disclosure: Attacker can gain access to Privileged information
- **D**enial of Service: Attacker can make the system unresponsive for legitimate use
- **E**levation of Privilege: Attacker can elevate their privileges

Spoofing

- Authentication Attacks against Host
- Docker Daemon Direct Access
- Trojanized Docker Images
- Exposure of Private Docker Registry
- ARP Spoofing
- Docker Registry Certificate Spoofing
- Insecure Docker API configuration

Tampering

- Trojanized Docker Image
- Docker Daemon Direct Access
- Docker Daemon Configuration Attacks
- Docker Registry Certificate Spoofing
- Content Trust
- Host File System Integrity Breaches
- Docker Daemon Tampering Host Network Configurations

Repudiation

- No Audit/Delete Audit Docker Images
- Docker Daemon API Logs - Compromise
- Host File System Integrity Breaches

Information Disclosure

- Secrets being disclosed to outside entities
- Exposed Ports and Services
- Network Traffic Compromise

Denial of Service

- CPU/Memory Exhaust
- Network Exhaust
- HDD Exhaust

Elevation of Privileges

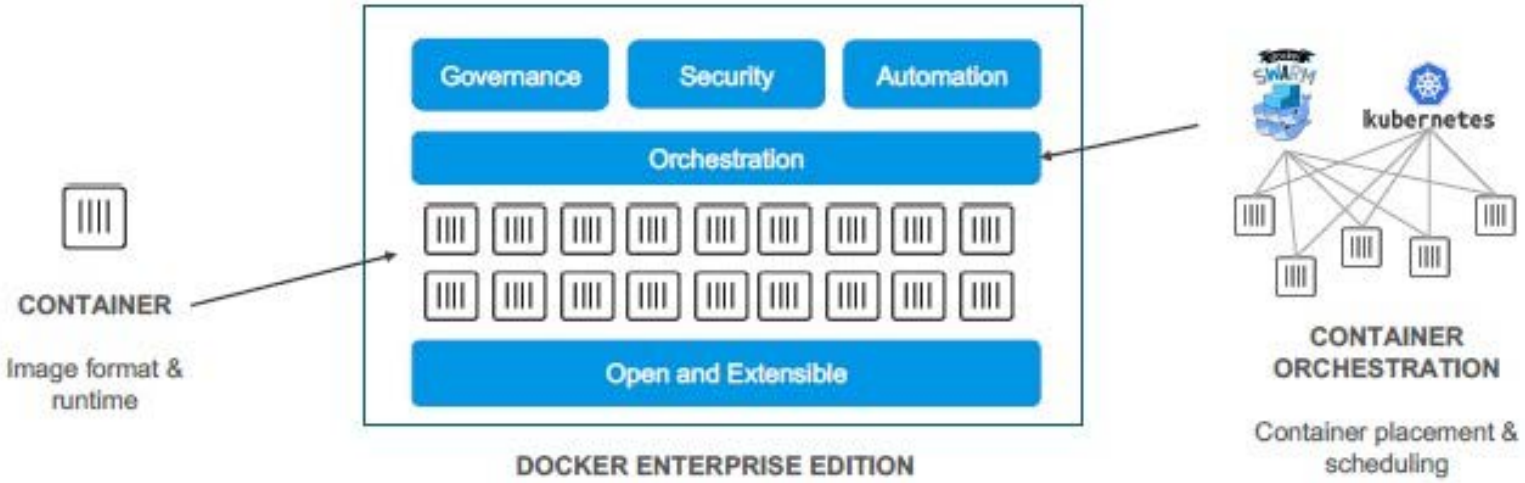
- Container Breakout
- Container Privileges
- Container Services - Compromise

DevOps


CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

Containers managed individually..

Led to Orchestration



What is Container Orchestration?

- Deploy and Configure
- Fault Isolation & Healing
- Secure
- Upgrades
- Scaling Up and Down

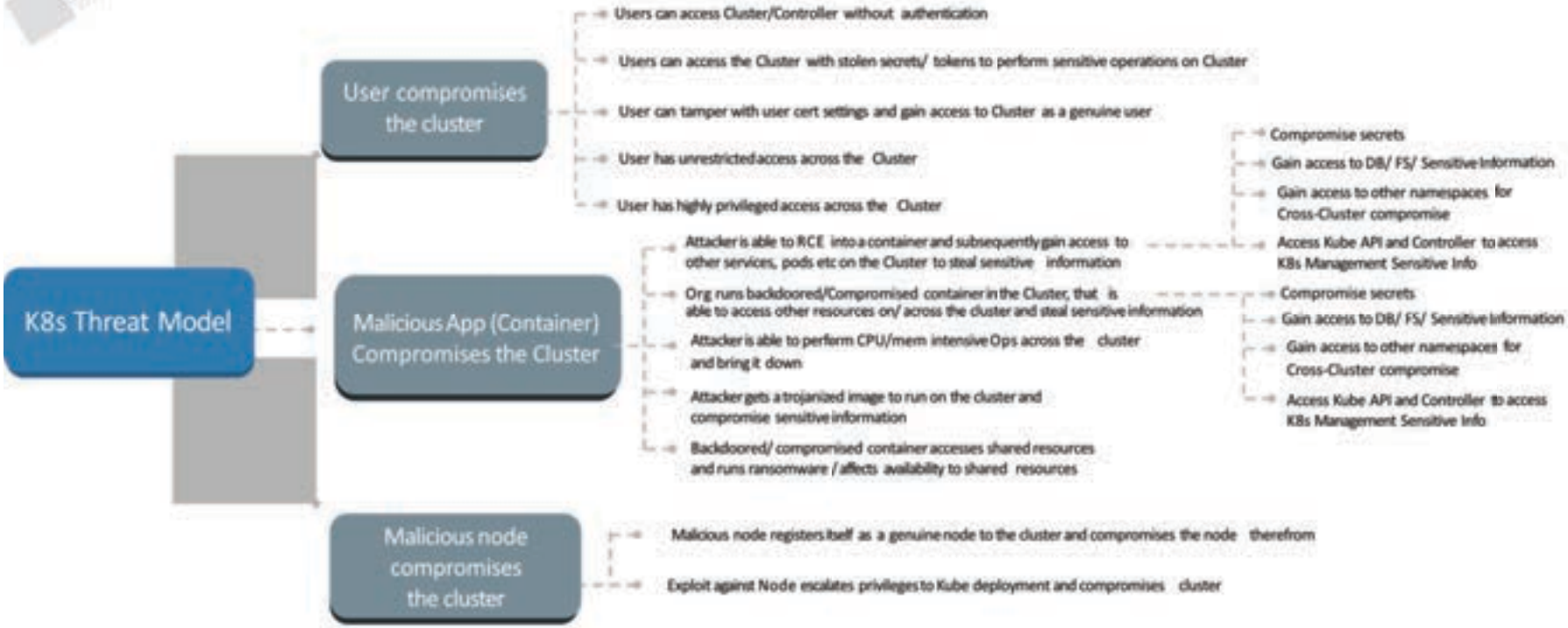
Stateless Applications

- Nothing to disk
- Web front-end
- Can stop and start as many containers as you like
- Container is ephemeral
- No container instance-specific configuration

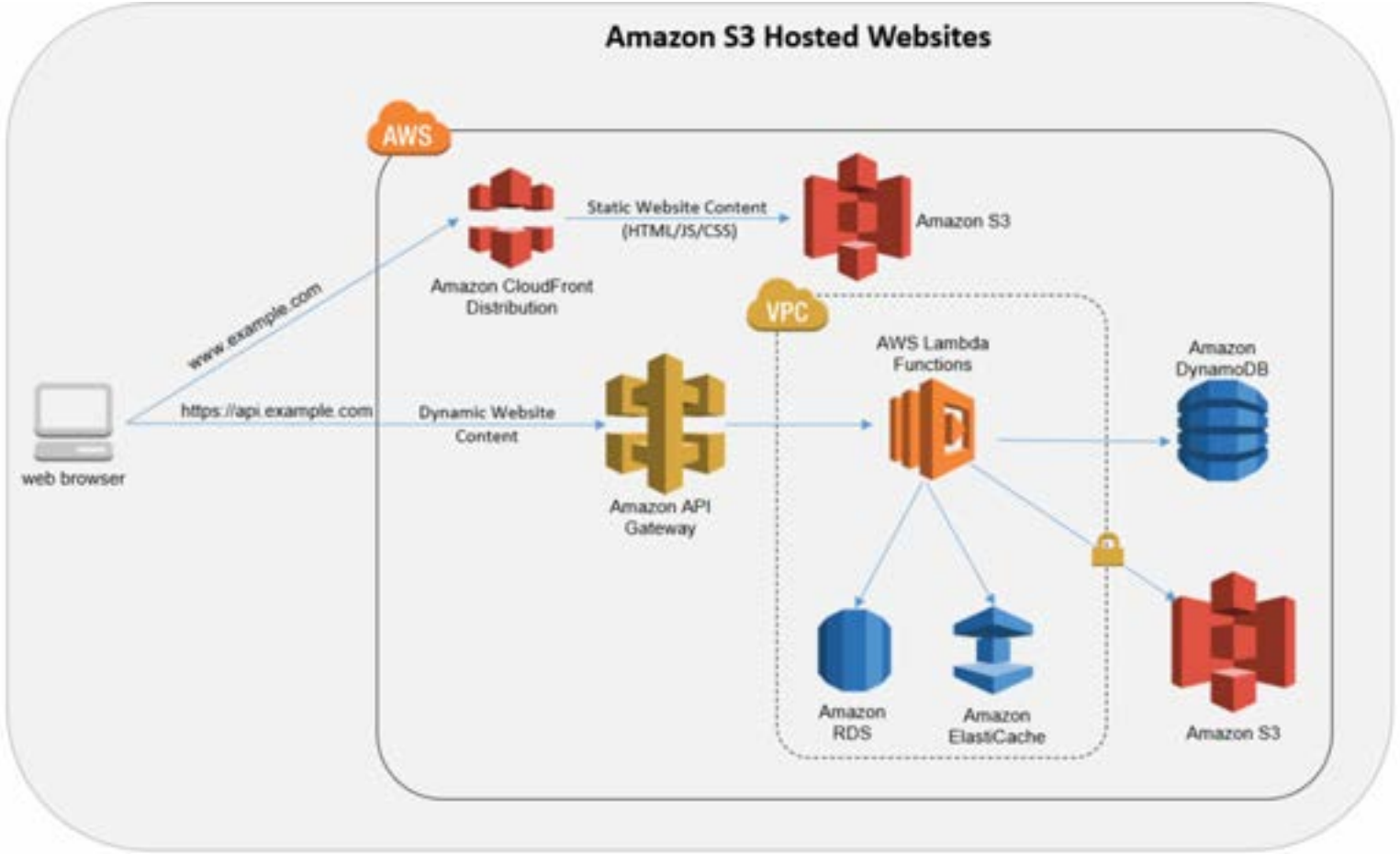
Stateful Applications

- Container-specific: Host names, IP addresses
- Big Data service configuration information
- Security secrets: passwords, KDC keys

The Kubernetes Threat Model



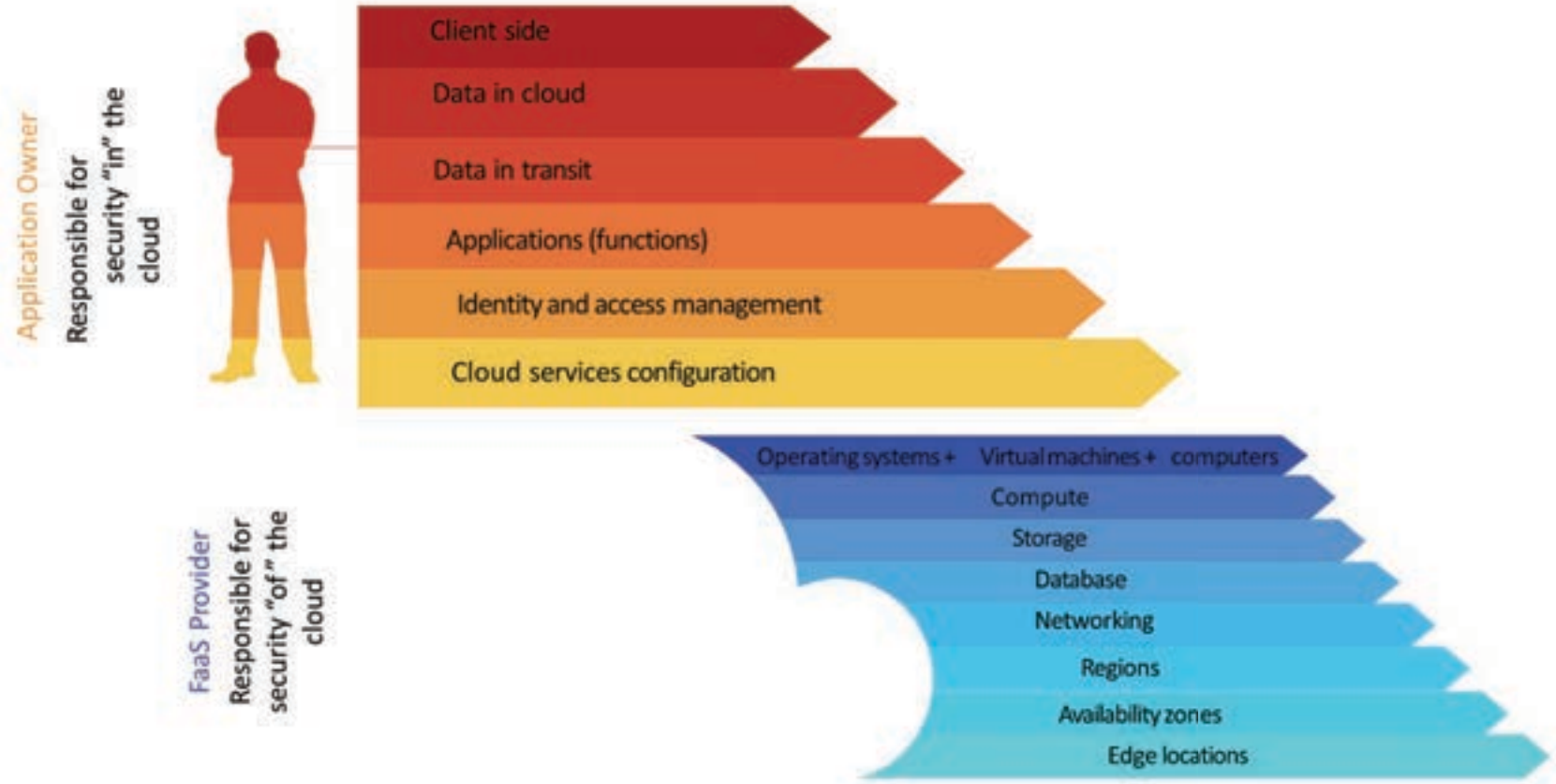
And Finally, Serverless!



Advantages of Serverless Computing

	Bare Metal	VM	Container	Serverless
Boot Time	~20 mins	~2 mins	2 secs	~0.0003 secs
App deployment lifecycle	Deploy in Weeks Live for years	Deploy in minutes Live for weeks	Deploy in Seconds Live for minutes/hours	Deploy in milliseconds Live for seconds
Development Complexity	Need to know: 1. Hardware 2. OS 3. Runtime Environm 4. Application code	Need to know: 1. OS 2. Runtime Environme 3. Application code	Need to know: 1. Runtime Environment 2. Application code	Need to know: 1. Application code
Investment	Buy/rent dedicated server	Rent a dedicated VM, on a shared server	Rent Containers, pay for the actual runtime	Pay for compute resouces used during runtime
Scaling	Takes months Should be approved by a panel of experts	Takes hours Should be approved by adminstators	Takes seconds Policy driven scaling	Takes milliseconds Scaling is event driven

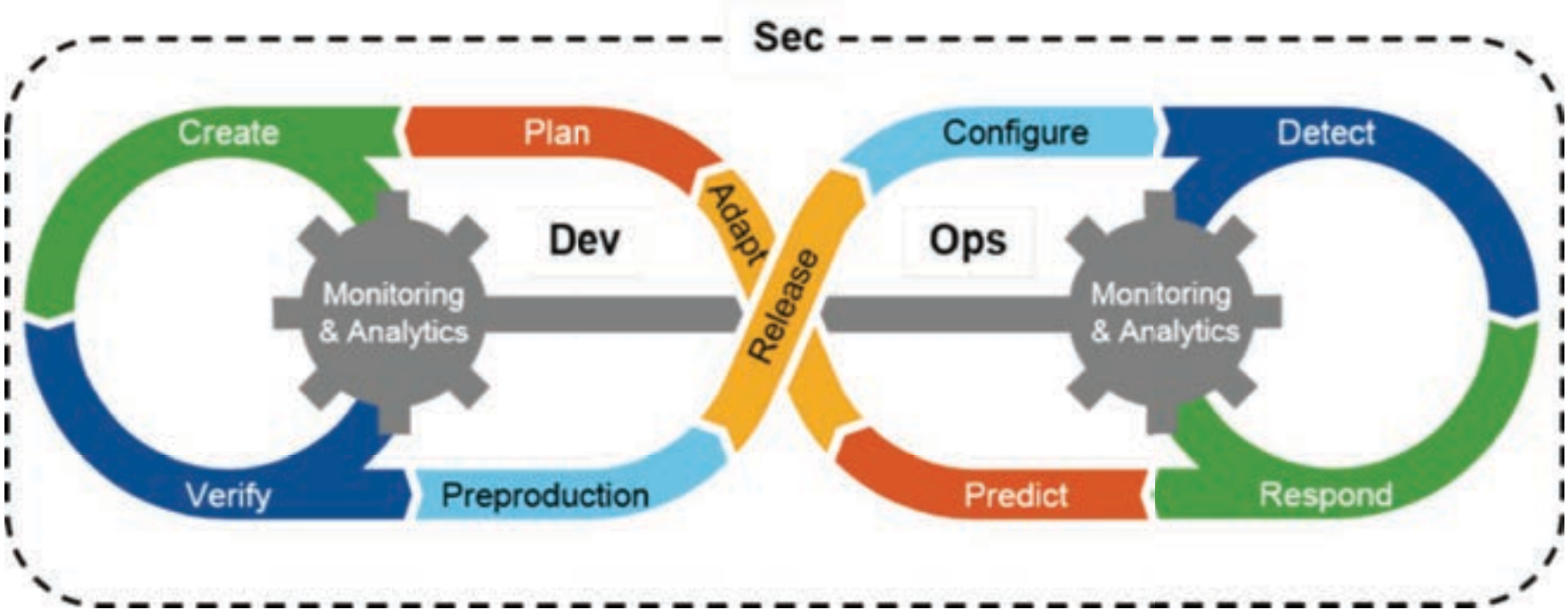
Serverless - Security Responsibility Model



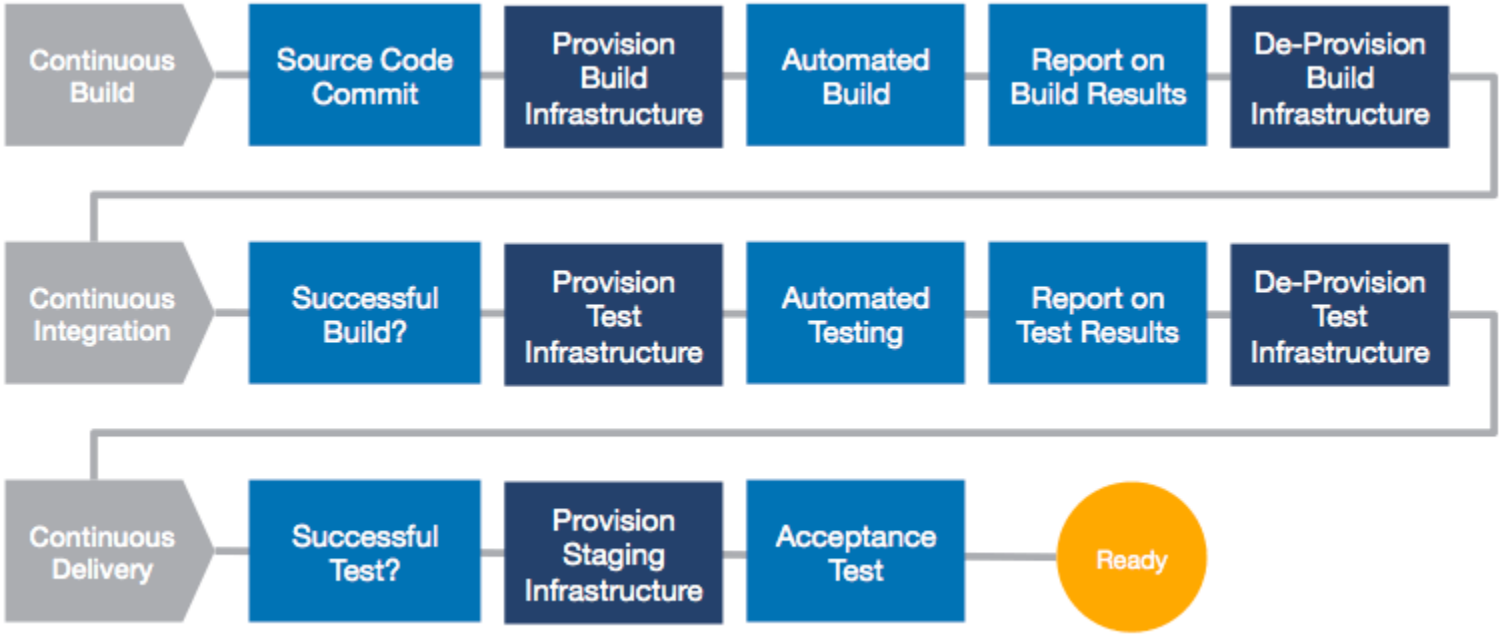
Serverless Architectures Security - Top 10

- SAS 1: Function Event Data Injection
- SAS 2: Broken Authentication
- SAS 3: Insecure Serverless Deployment Configuration
- SAS 4: Over-Privileged Function Permissions and Roles
- SAS 5: Inadequate Function Monitoring and Logging
- SAS 6: Insecure 3rd Party Dependencies
- SAS 7: Insecure Application Secrets Storage
- SAS 8: Denial of Service and Financial Resource Exhaustion
- SAS 9: Serverless Function Flow Manipulation
- SAS 10: Improper Exception Handling and Verbose Errors

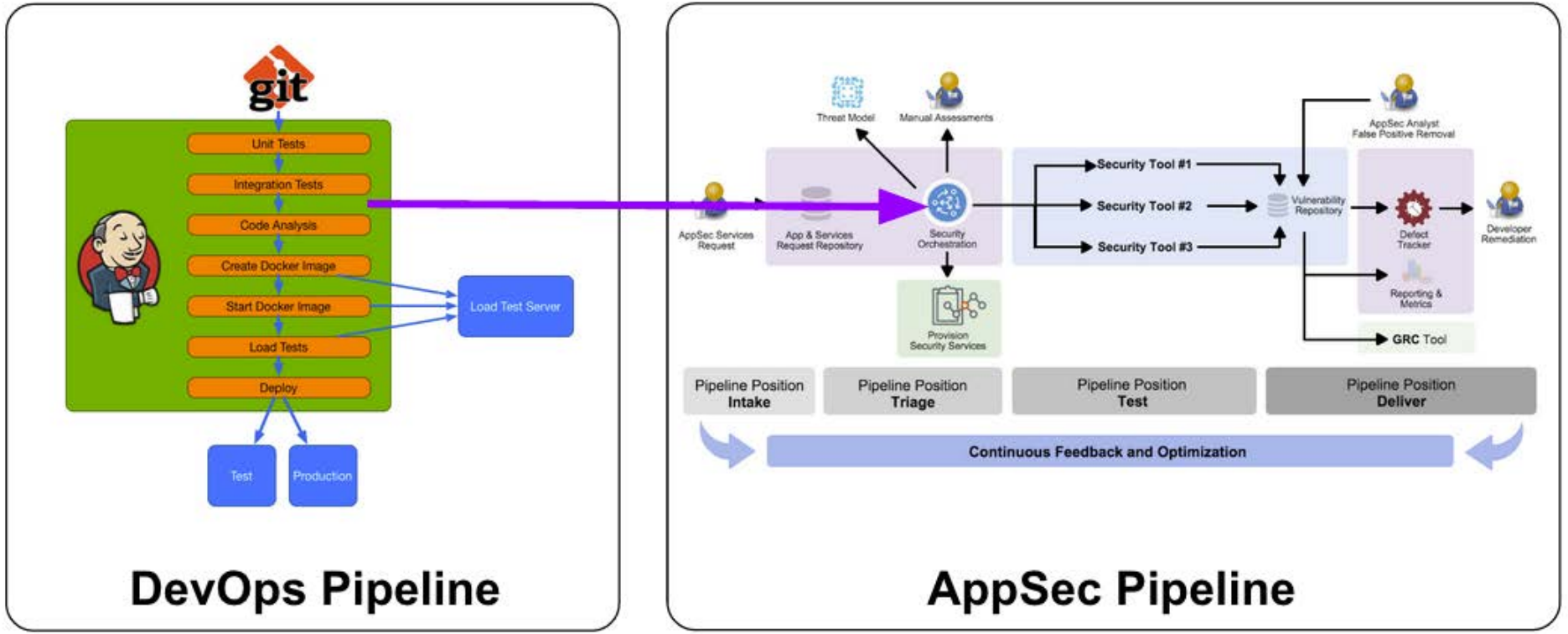
DevSecOps


Sec
Create
Plan
Configure
Detect
Dev
Adapt
Release
Ops
Monitoring & Analytics
Monitoring & Analytics
Verify
Preproduction
Predict
Respond

Continuous Build, Integration and Delivery– Foundational and Automated (CI/CD)



Example CI/CD pipeline with AppSec addition



Great place to start!

DevSecOps Studio is one of its kind, self contained DevSecOps environment/ distribution to help individuals in learning DevSecOps concepts. It takes lots of efforts to setup the environment for training/demos and more often, its error prone when done manually. DevSecOps Studio is easy to get started and is mostly automatic.

DevSecOps Studio project aims to reduce the time to bootstrap the environment and help you in concentrating on learning/teaching DevSecOps practices.

Features:
- Easy to setup environment with just one command “vagrant up”
- Teaches Security as Code, Compliance as Code, Infrastructure as Code
- With built-in support for CI/CD pipeline
- OS hardening using ansible
- Compliance as code using Inspec
- QA security using ZAP, BDD-Security and Gauntlt
- Static tools like bandit, brakeman, windbags, gitrob, gitsecrets
- Security Monitoring using ELK stack.

https://www.owasp.org/index.php/OWASP_DevSecOps_Studio_Project