Security Best Practices

for Amazon Elastic Kubernetes Service (EKS)

Jeremy Cowan, Principal SA
Containers

Agenda

How do you solve a problem like containers?
EKS Shared Responsibility Model & Security Pillars
Security best practices

- Hosts
- Container images
- Identity and Access Management (IAM)
- Network
- Pod and runtime
- Auditing and forensics

Closing thoughts

aws

Challenges posed by containers

Processes running on a shared kernel

Isolation implemented by Linux namespaces and cgroups

Short lifespans

Traditional/legacy security software is rarely container-aware

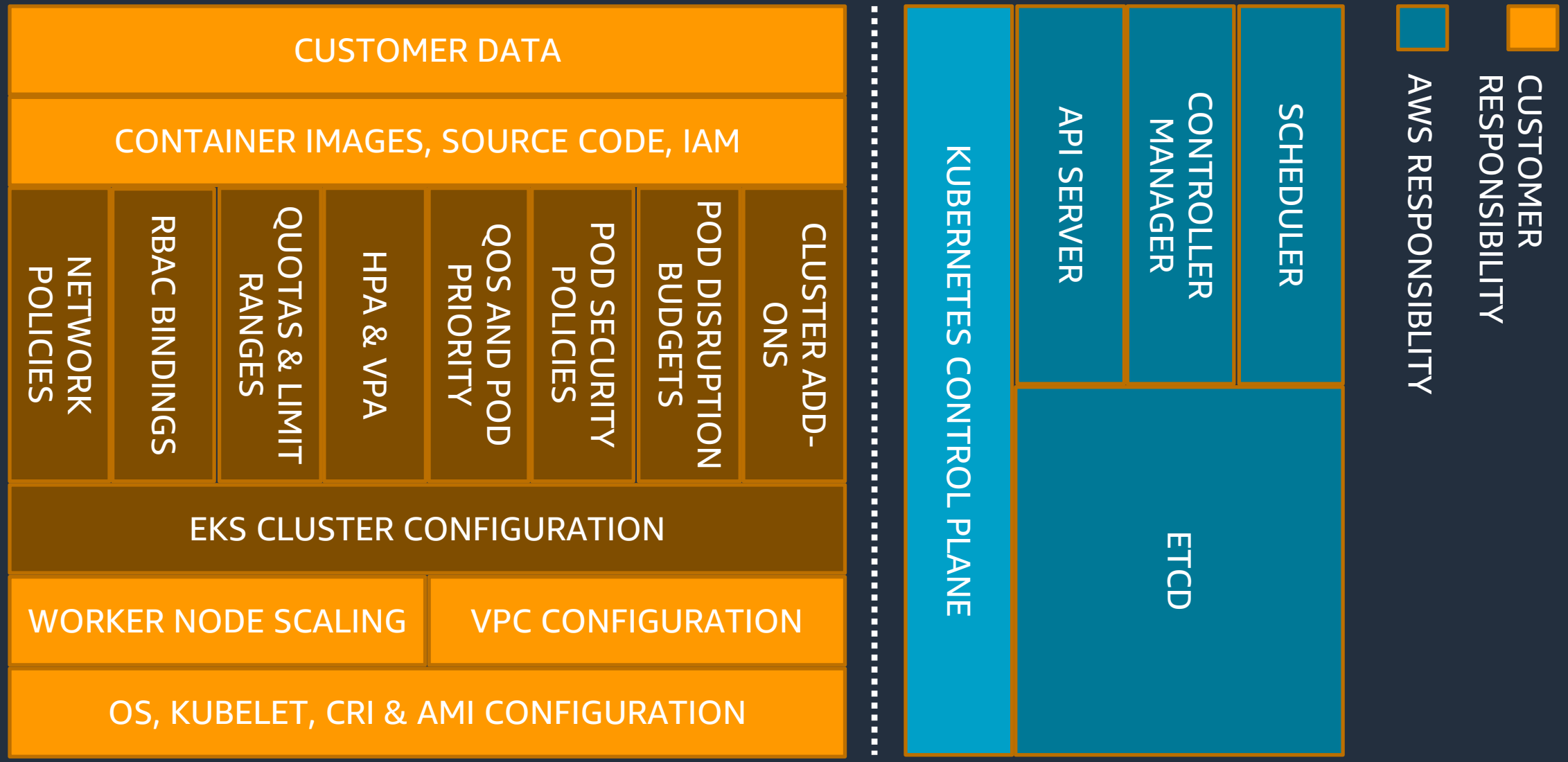
- Firewalls
- IDS/IPS
- DLP
- Forensics

Warrants a different approach

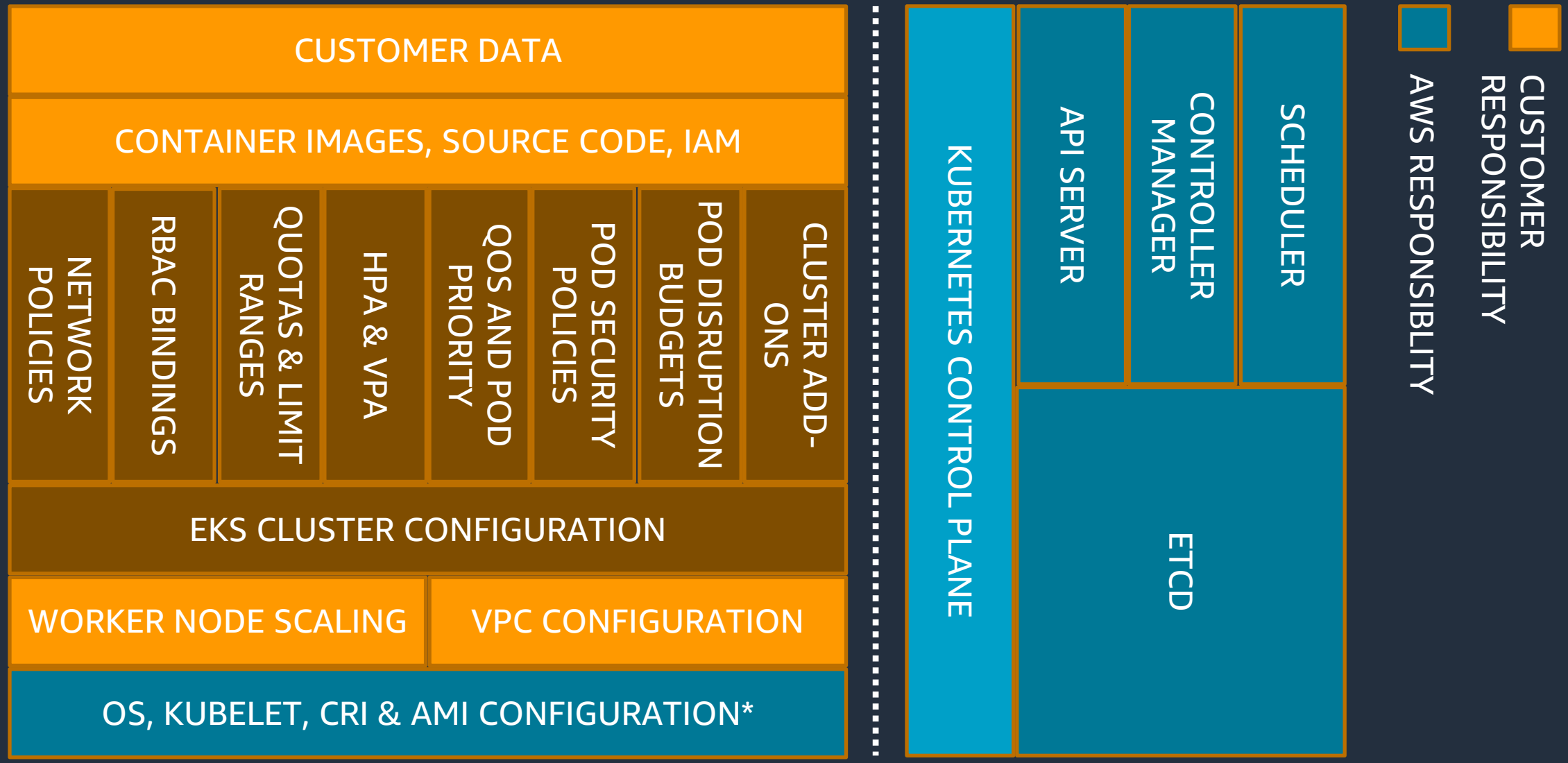
aws

Shared Responsibility Model

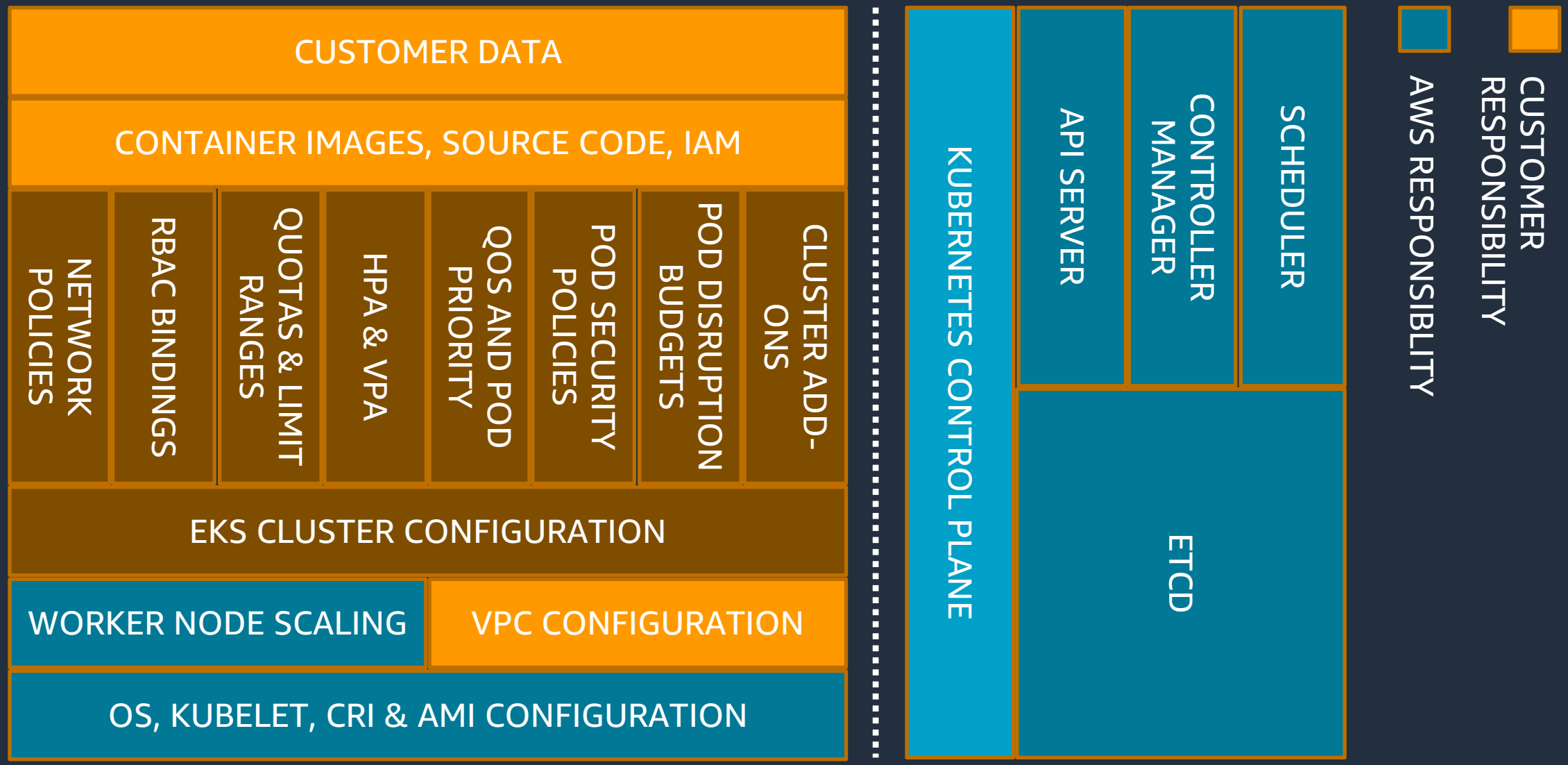
EKS with self-managed workers



EKS with managed node groups (current)



EKS Fargate



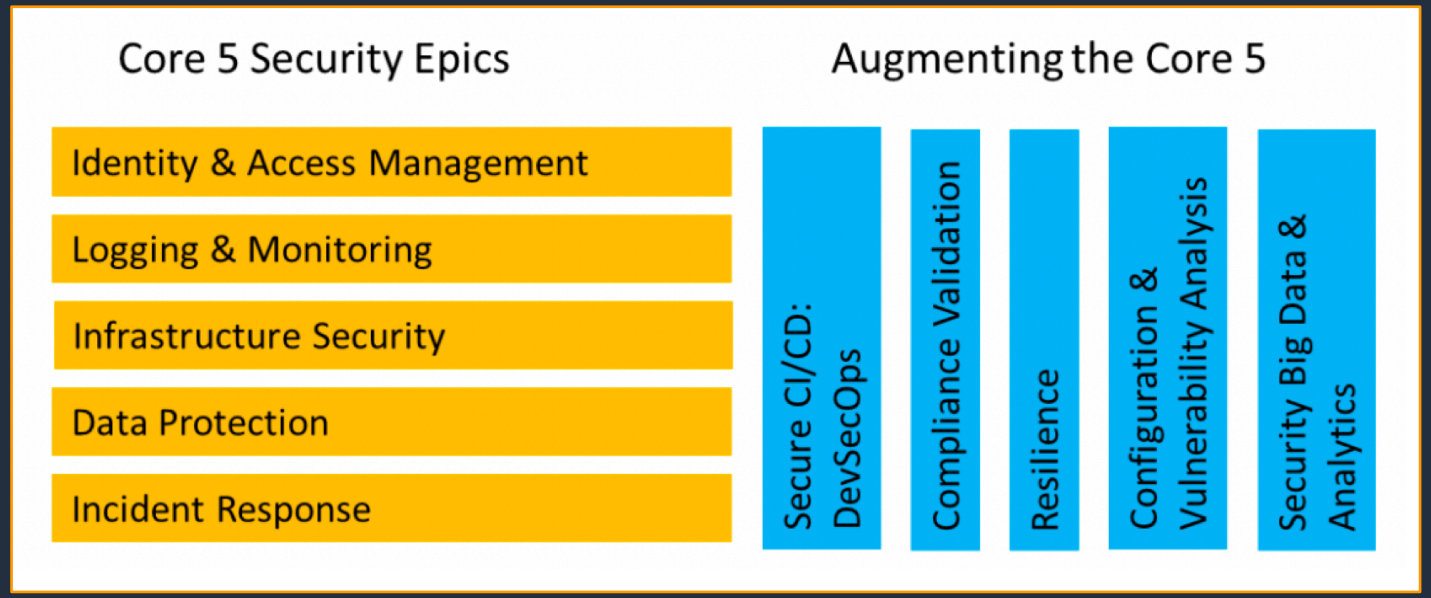
Security Pillars

Container security tenets and epics

Tenets

- All-encompassing
- Shared responsibility
- Cloud native

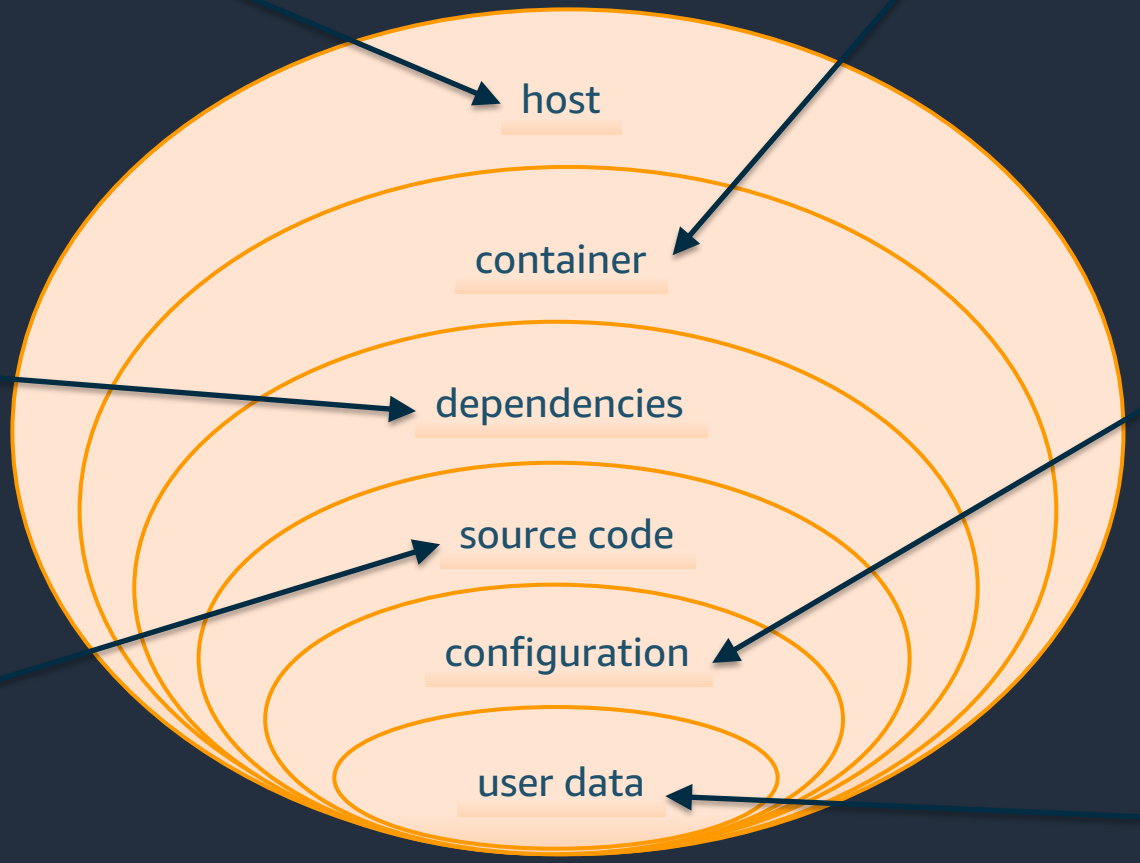
Epics



Container security onion model: defense in depth

- full blown distro (AL2) vs. container-optimized distribution
- multi-tenancy requirements
- **gotchas**: Linux packages/CVEs, leaks, GDPR (in Europe)

- runtime/standards (OCI)
- immutability of images
- all containers share a kernel (mitigation: Firecracker)
- **gotchas**: unnecessary privileged users, no scans, trust



- code analysis
- source available?
- **gotchas**: huge surface, many languages

- sensitive config (passwords, API keys, etc.)
- **gotchas**: commits-to-source, non-separated access (dev has cleartext password)

- sanitizing user input
- static code analysis
- **gotchas**: log-leaking

- business core data
- Personal Identifiable Information (PII)
- **gotchas**: leaks, GDPR (in Europe)

aws

EKS Best Practices for Security

Host Security

Use an OS that is optimized for running containers

- EKS Optimized Amazon Linux 2 & Bottlerocket (preview)
- Alternatives: Atomic, Flatcar Linux, RancherOS

Deploy workers onto private subnets

Run Amazon Inspector to continually assess alignment with best practices and compliance requirements

- kube-bench
- EKS CIS benchmarks

Minimize and audit access

- SSM

Use SELinux (RHEL & CentOS)

- Audit2Allow, Audit2Why, SEAlert

aws

Securing container images

Scan container images

- ECR, Anchore, Clair, Trivy

Use Scratch or a slim base layer

De-fang your images

- Remove files with the SETUID and SETGID bits from the image

Always run as a non-root user

- Lint your Dockerfiles

Use endpoint policies and private endpoints with Elastic Container Registry (ECR)

aws

Identity and Access Management (IAM)

General guidelines

- Practice the principle of least privilege for AWS IAM and k8s RBAC
- Configure EKS cluster endpoint to be PRIVATE
- Periodically audit access to the cluster

IAM

- Use IRSA to assign AWS identities to pods
- Block access to EC2 metadata

Kubernetes

- Use separate services accounts for each application
- Disable mounting of the default SA token

Network security

Allow all traffic is the default policy

Use k8s network policies for restricting E-W traffic within the cluster

Start with a deny-all global policy and incrementally add policies

- Port 53 (DNS) egress to kube-system
- Allow all within a namespace

Restrict outbound traffic from pods that don’t need to connect to external services

- SGs for pods & Cilium (L7 policies)

Encrypt service-to-services traffic with a mesh

- Alternatives: Select CNI plug-ins & Nitro instances

aws

Pod and runtime security

Use PSPs or OPA/Gatekeeper to implement runtime security measures:

- Deny privileged escalation
- Deny running as root
- Deny mounting hostPath
- Drop Linux capabilities

Compliment PSPs with AppArmor or Seccomp profiles (if necessary)

Use 3rd party solutions

- Aqua, Stackrox, Sysdig Falco, Twistlock

aws

Auditing and forensics

Enable control plane logs

Stream logs from containers to an external log aggregator

Periodically audit Kubernetes control plane and AWS CloudTrail logs for suspicious activity

- Search for the annotations `authorization.k8s.io/decision` and `authorization.k8s.io/reason` to ascertain why a call was allow/denied

Immediately isolate pods you suspect have been compromised

- Remove/change labels
- Create network policy to isolate the pod

Cordon the instance (if necessary)

- Capture volatile artifacts on the worker node, e.g. memory, disk, etc.

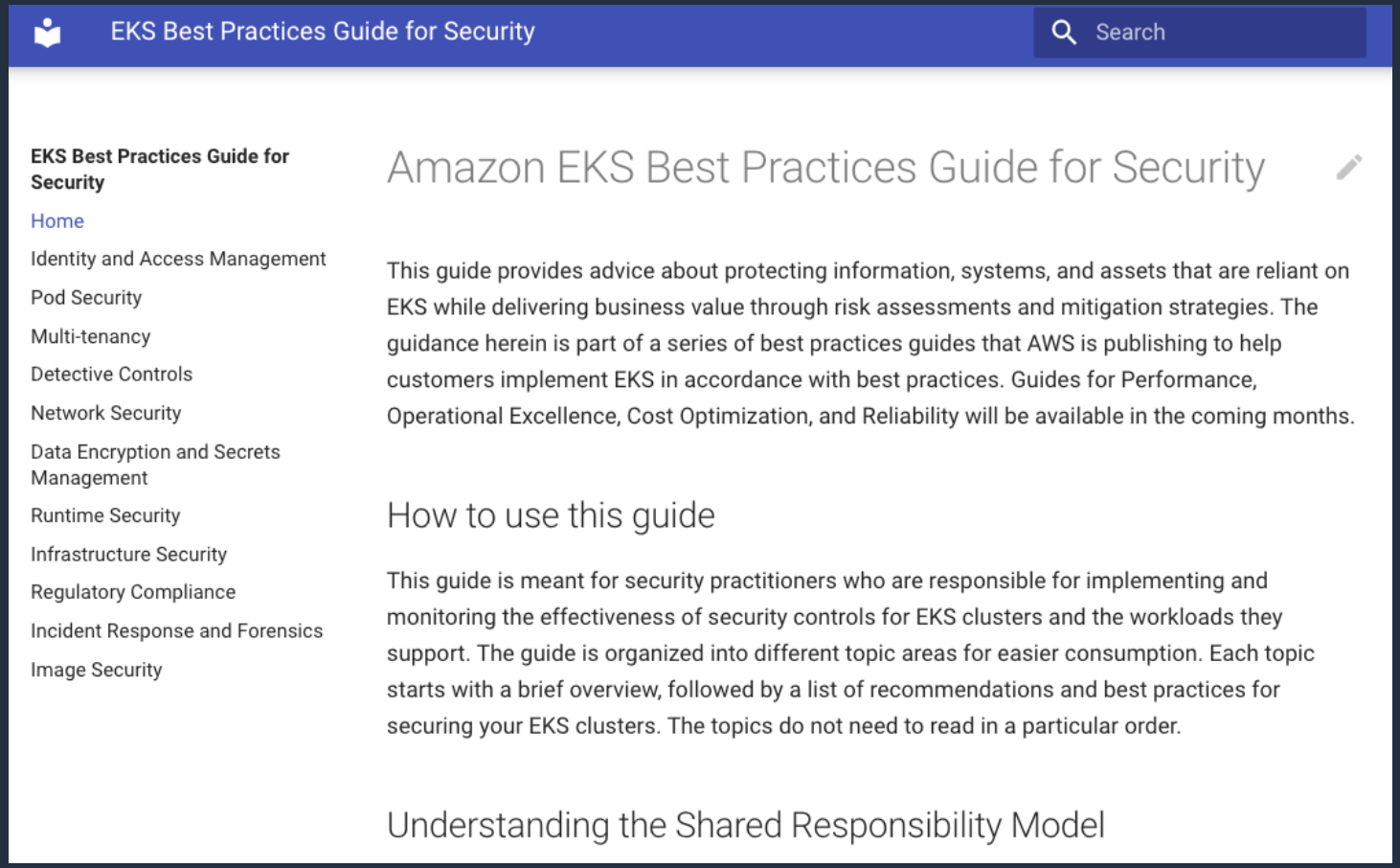
kube-forensics

aws

Closing thoughts

- Security is everyone’s responsibility
- Understand the shared responsibility model
- Shift left & DevSecOps

EKS Best Practices for Security



https://aws.github.io/aws-eks-best-practices/

Thank you!