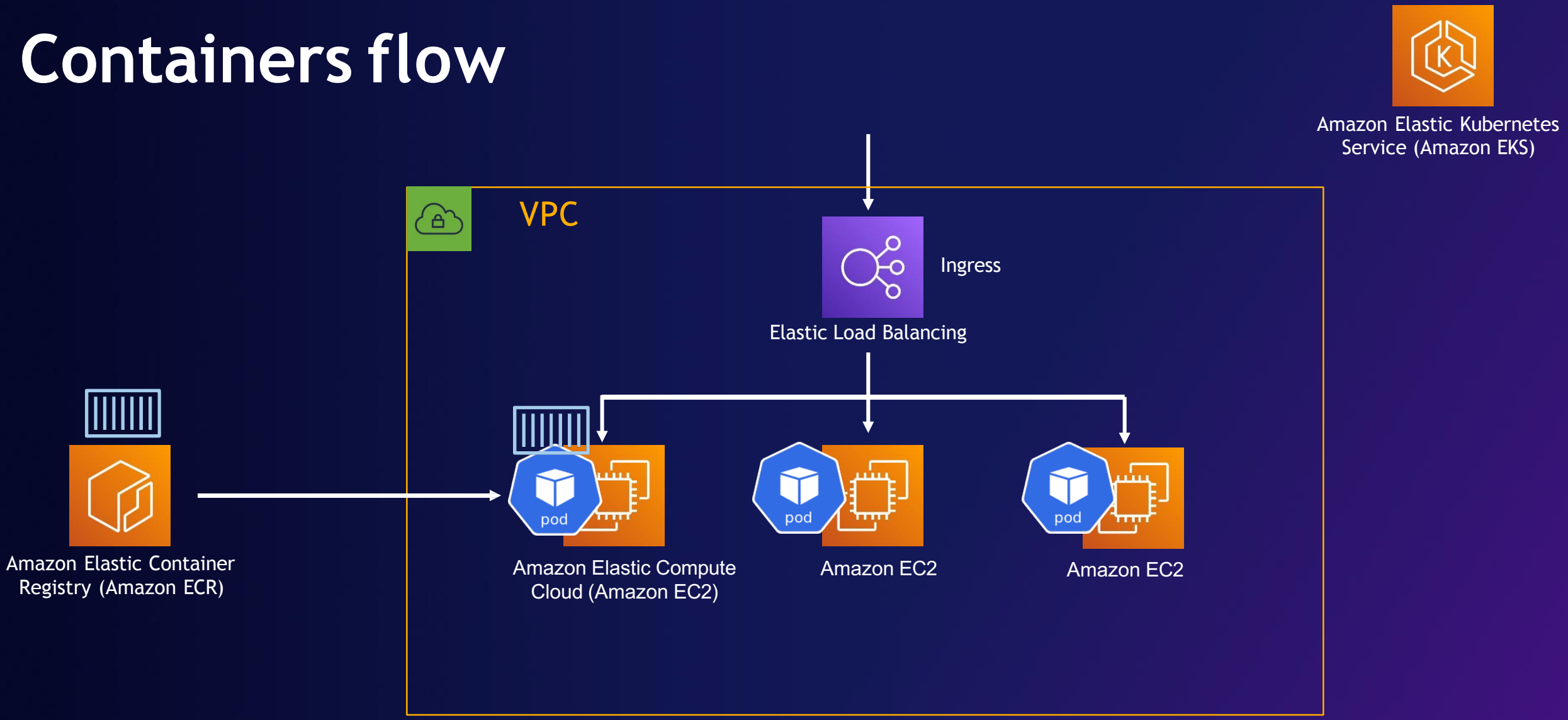
Security Best Practices for Amazon EKS

Hoseok Seo (he/him)
Solutions Architect

“Security is everyone’s job.”

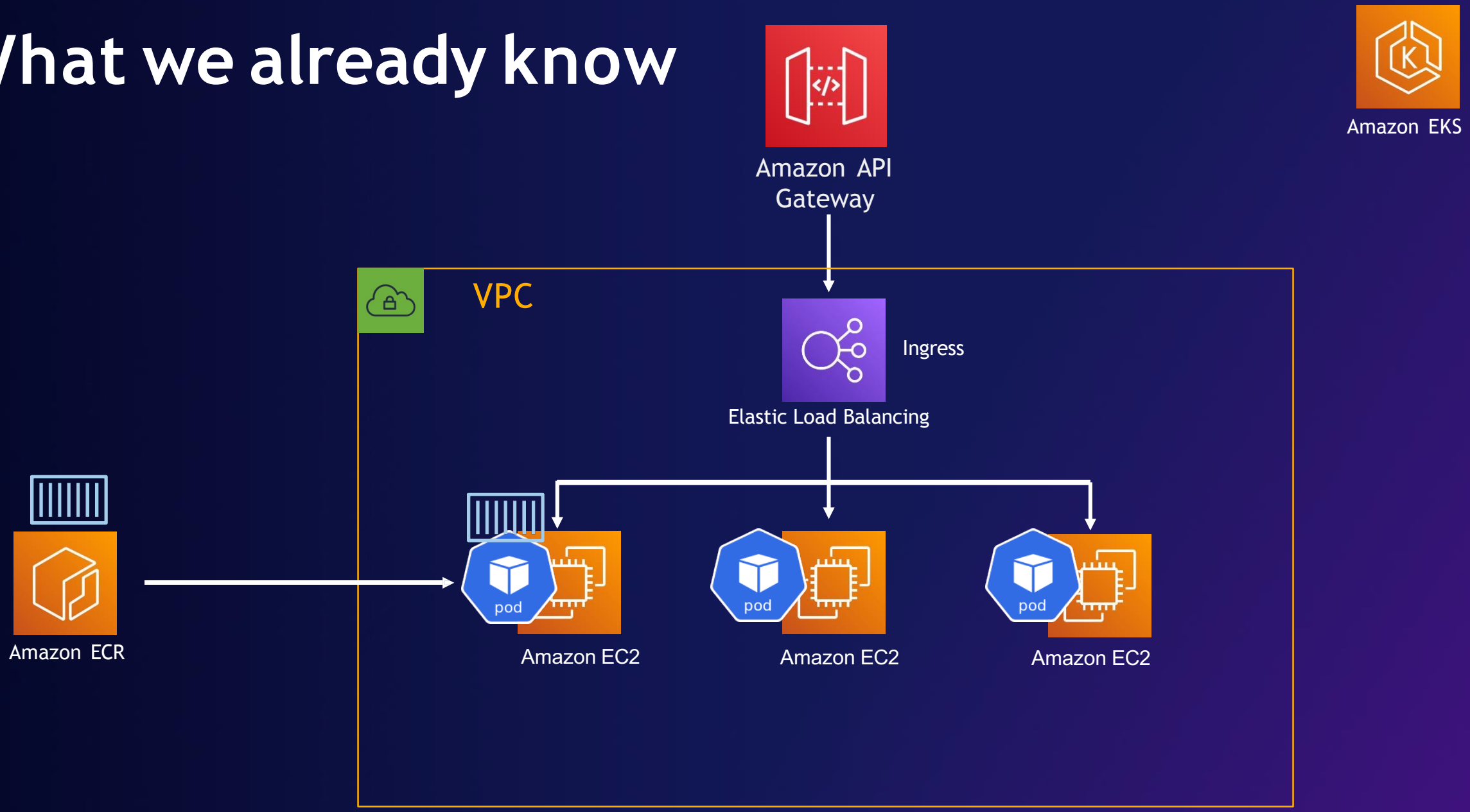
Werner Vogel
AWS CTO

Containers flow



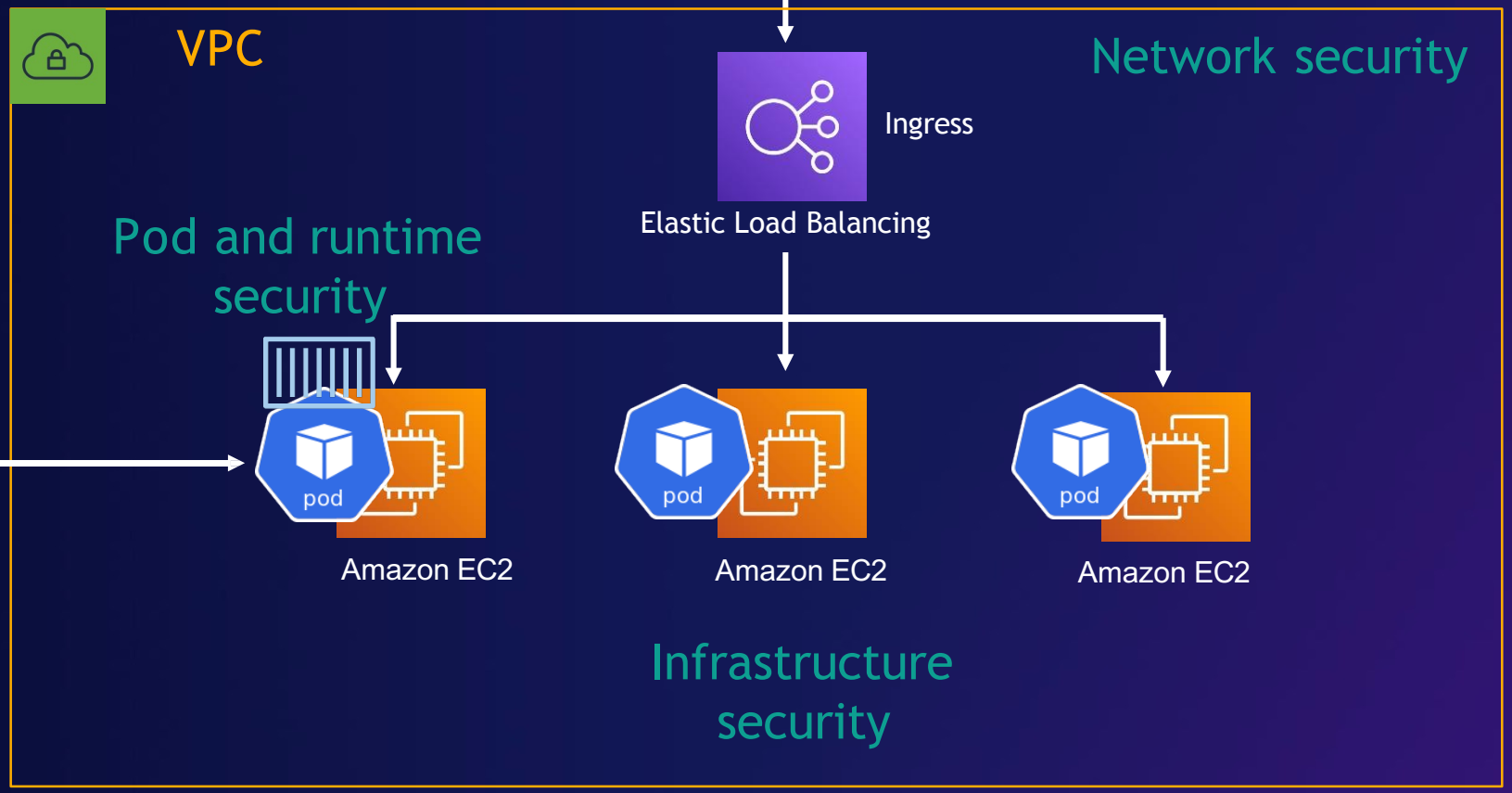
What we already know

What we already know



Containers-specific areas

Amazon EKS



Audit & forensics

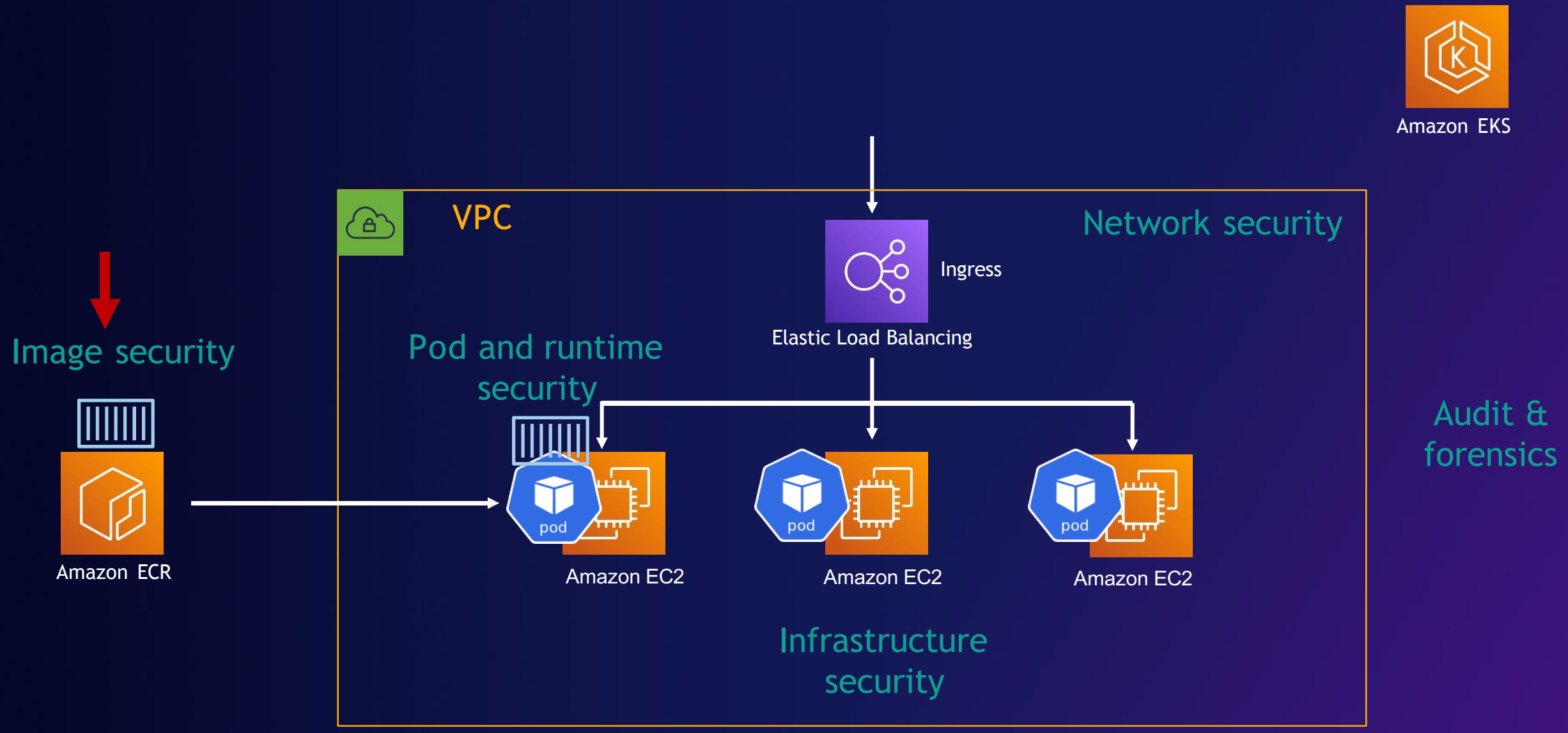


Image security

- Scan your images for vulnerabilities (CVEs)
  - ECR Basic Scanning (OS packages)
  - ECR enhanced scanning (OS and programming packages)

Inspector container image findings

Inspector > Findings > By container image > arn:aws:ecr:ap-northeast-2:628695400614:repository/spring4shell/sha256:9fd8a153a8451796c82ce0474bea5b603d5d943d172c070d1aff42e83f845d21

latest Info

Container image: sha256:9fd8a153a8451796c82ce0474bea5b603d5d943d172c070d1aff42e83f845d21

Details

AWS account
Hoseok

Repository
spring4shell

Image tags
latest

Finding summary
Package finding
22 Critical 36 High 39 Medium

By image | By layer

Findings (109)

Active | Resource ID EQUALS arn:aws:ecr:ap-northeast-2:6286954006... | Add filter

1 2 3 4

Severity	Title	Type	Age	Status
Critical	CVE-2022-1586 - pcre2	Package Vulnerability	a day	Active
Critical	CVE-2022-37434 - zlib	Package Vulnerability	19 days	Active
Critical	CVE-2022-32207 - curl, curl	Package Vulnerability	40 days	Active
Critical	CVE-2022-23219 - glibc	Package Vulnerability	83 days	Active
Critical	SNYK-JAVA-ORGCODEHAUSPLEXUS-31522 - org.codehaus.plexus:plexus-utils, org.codehaus.plexus:plexus-utils and 1 more	Package Vulnerability	7 days	Active
Critical	CVE-2021-22945 - curl, curl	Package Vulnerability	40 days	Active
Critical	GHSA-36p3-wjmg-h94x - org.springframework:spring-beans, org.springframework:spring-beans and 2 more	Package Vulnerability	83 days	Active

aws

Image security

- Scan your images for vulnerabilities (CVEs)
  - ECR Basic Scanning (OS packages)
  - ECR enhanced scanning (OS and programming packages)

- Minimize attack surface
  - Use multi stage build
  - Create images from scratch

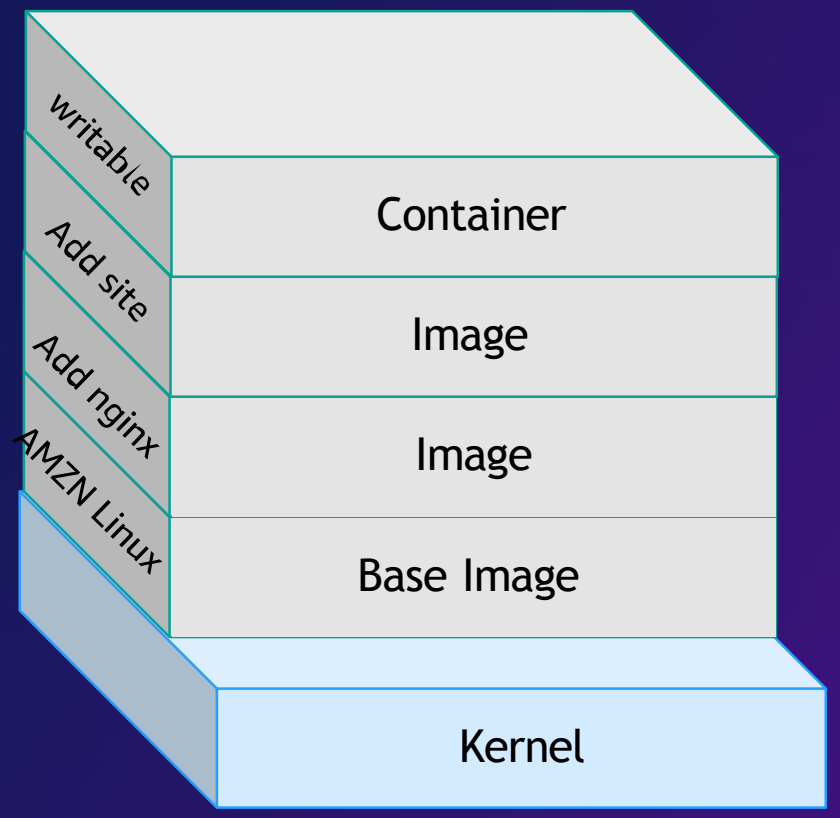
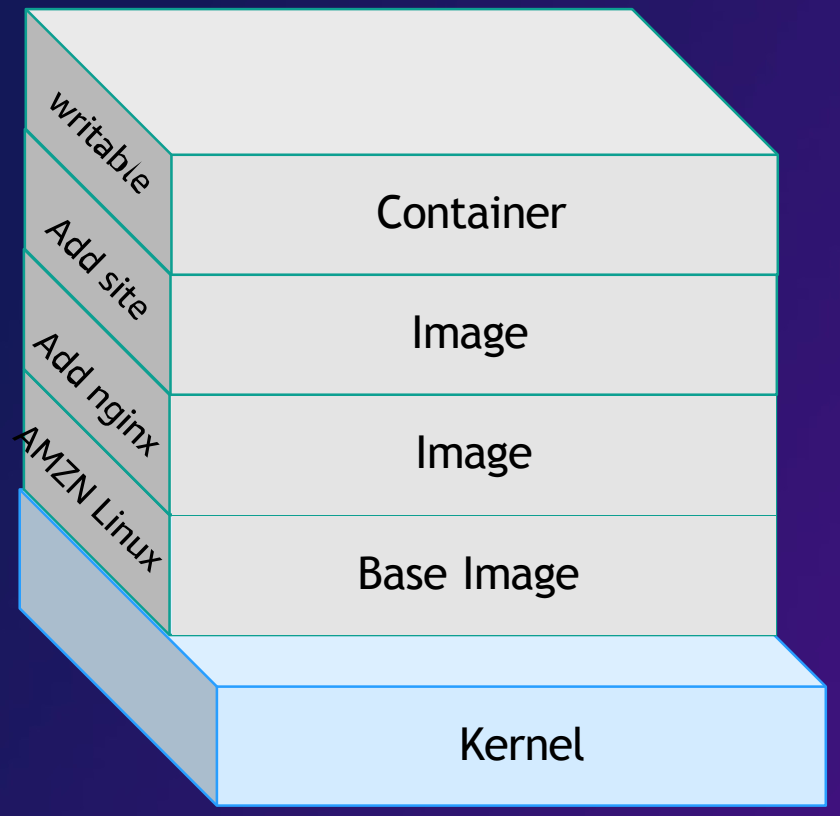


Image security

- Scan your images for vulnerabilities (CVEs)
  - ECR Basic Scanning (OS packages)
  - ECR enhanced scanning (OS and programming packages)

- Minimize attack surface
  - Use multi stage build
  - Create images from scratch

- Run the application as a non-root user
  - Lint your Dockerfiles with Dockle or Hadolint



Lint Dockerfiles

```
FROM alpine:3.16

LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>"

ENV NGINX_VERSION 1.22.0
ENV PKG_RELEASE   1

RUN set -x \
# create nginx user/group first, to be consistent throughout docker variants
    && addgroup -g 101 -S nginx \
    && adduser -S -D -H -u 101 -h /var/cache/nginx -s /sbin/nologin -G nginx -g nginx nginx \
    && apkArch="$(cat /etc/apk/arch)" \
    && nginxPackages=" \
        nginx=${NGINX_VERSION}-r${PKG_RELEASE} \
    " \
# install prerequisites for public key and pkg-oss checks
    && apk add --no-cache --virtual .checksum-deps \
        openssl \
    && case "$apkArch" in \
        x86_64|aarch64) \
# arches officially built by upstream
            set -x \
            && KEY_SHA512="e7fa8303923d9b95db37a77ad46c68fd4755ff935d0a534d26eba83de193c76166c68bfe7f65471bf8881004ef4aa6df3e34689c305662750c0172fca5d8552a *stdin" \
            && wget -O /tmp/nginx_signing.rsa.pub https://nginx.org/keys/nginx_signing.rsa.pub \
            && if [ "$(openssl rsa -pubin -in /tmp/nginx_signing.rsa.pub -text -noout | openssl sha512 -r)" = "$KEY_SHA512" ]; then \
                echo "key verification succeeded!"; \
                mv /tmp/nginx_signing.rsa.pub /etc/apk/keys/; \
            else \
                echo "key verification failed!"; \
                exit 1; \
            fi \
            && apk add -X "https://nginx.org/packages/alpine/v$(egrep -o '^[0-9]+\.[0-9]+' /etc/alpine-release)/main" --no-cache $nginxPackages \
            ;; \
        *) \
```

```
[ec2-user@ip-10-0-1-211 ~]$ ./dockle alpine:3.16
WARN  - CIS-DI-0001: Create a user for the container
  * Last user should not be root
INFO  - CIS-DI-0005: Enable Content trust for Docker
  * export DOCKER_CONTENT_TRUST=1 before docker pull/build
INFO  - CIS-DI-0006: Add HEALTHCHECK instruction to the container image
  * not found HEALTHCHECK statement
[ec2-user@ip-10-0-1-211 ~]$
[ec2-user@ip-10-0-1-211 ~]$ hadolint Dockerfile
Dockerfile:13 DL3018 warning: Pin versions in apk add. Instead of `apk add <package>` use `apk add <package>=<version>`
Dockerfile:13 DL3047 info: Avoid use of wget without progress bar. Use `wget --progress=dot:giga <url>`.Or consider using `-q` or `-nv` (shorthands for `--
quiet` or `--no-verbose`).
Dockerfile:13 SC2086 info: Double quote to prevent globbing and word splitting.
Dockerfile:13 DL4006 warning: Set the SHELL option -o pipefail before RUN with a pipe in it. If you are using /bin/sh in an alpine image or if your shell i
s symlinked to busybox then consider explicitly setting your SHELL to /bin/ash, or disable this check
```

https://github.com/nginxinc/docker-nginx/blob/master/stable/alpine-slim/Dockerfile

Image security

- Scan your images for vulnerabilities (CVEs)
  - ECR Basic Scanning (OS packages)
  - ECR enhanced scanning (OS and programming packages)

- Minimize attack surface
  - Use multi stage build
  - Create images from scratch

- Run the application as a non-root user
  - Lint your Dockerfiles with Dockle or Hadolint

- “Defang” your containers
  - Remove files with the SETUID and SETGID bits from image

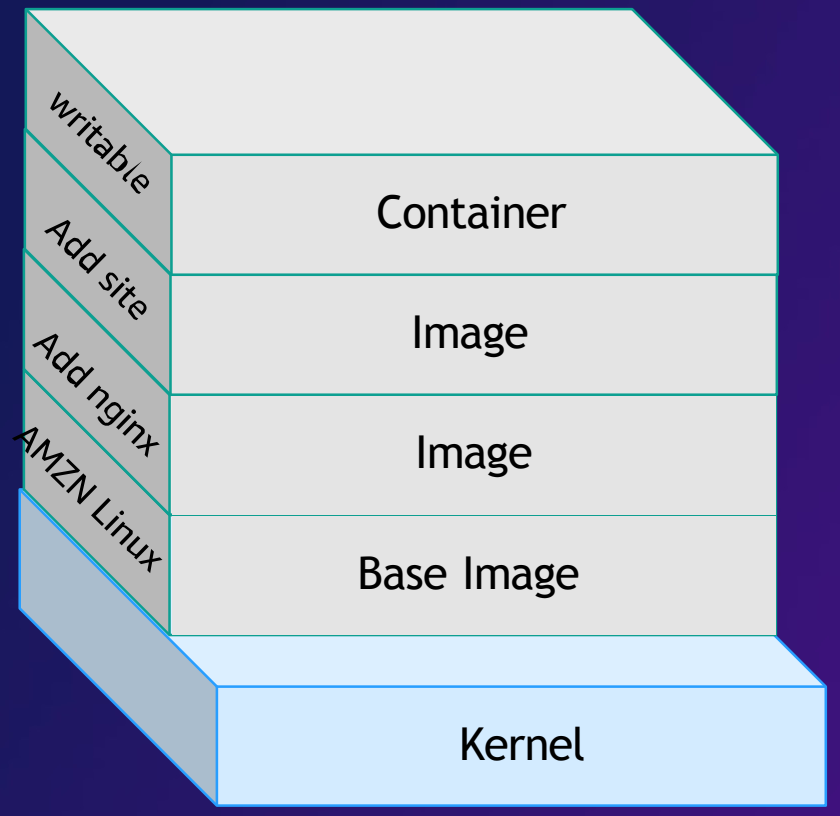


Image security - dive

De-fang your images

- Remove files with the SETUID and SETGID bits from the image
- Dive (https://github.com/wagoodman/dive)

```
1 / 1                                   Tilix: Default
1: Terminal
[● Layers]──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────[Current Layer Contents]──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
Cmp Image ID                      Size  Command                                                                                                 Permission     UID:GID       Size  Filetree
    sha256:cd7100a72410606589  4.1 MB  FROM sha256:cd7100a72410606589                                                                          drwxr-xr-x         0:0     805 kB  ├─⊕ bin
    sha256:f03b1ccbacace8c82e  2.1 kB  #(nop) ADD file:63d4894bd0857354b                                                                       drwxr-xr-x         0:0        0 B  ├── dev
    sha256:d2a26525cdac6c4358     0 B  mkdir /root/example                                                                                      drwxr-xr-x         0:0     251 kB  ├─⊕ etc
    sha256:9f50561518484bb62a  2.1 kB  cp /somefile.txt /root/example/so                                                                      drwxr-xr-x         0:0        0 B  ├── home
    sha256:edf209d09263ab03c7  2.1 kB  cp /somefile.txt /root/example/so                                                                      drwxr-xr-x         0:0     2.7 MB  ├─⊕ lib
    sha256:bb70c7aab83602e8a6  2.1 kB  cp /somefile.txt /root/example/so                                                                      drwxr-xr-x         0:0        0 B  ├── media
    sha256:b0a712d3d00bc83c48  2.1 kB  mv /root/example/somefile3.txt /r                                                                      drwxr-xr-x         0:0        0 B  │   ├── cdrom
    sha256:05e8660b8f90e77b93     0 B  rm -rf /root/example/                                                                                   drwxr-xr-x         0:0        0 B  │   ├── floppy
                                                                                                                                               drwxr-xr-x         0:0        0 B  │   └── usb
[Image & Layer Details]──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────  drwxr-xr-x         0:0        0 B  ├── mnt
                                                                                                                                               dr-xr-xr-x         0:0        0 B  ├── proc
Layer Command                                                                                                                                  drwx------         0:0     6.2 kB  ├── root
/bin/sh -c cp /somefile.txt /root/example/somefile3.txt                                                                                        drwxr-xr-x         0:0     6.2 kB  │   └── example
                                                                                                                                               -rw-r--r--         0:0     2.1 kB  │       ├── somefile1.txt
Image efficiency score: 99 %                                                                                                                   -rw-r--r--         0:0     2.1 kB  │       ├── somefile2.txt
Potential wasted space: 6.2 kB                                                                                                                 -rw-r--r--         0:0     2.1 kB  │       └── somefile3.txt
                                                                                                                                               drwxr-xr-x         0:0        0 B  ├── run
Count   Total Space  Path                                                                                                                      drwxr-xr-x         0:0     213 kB  ├─⊕ sbin
    2        4.2 kB  /root/example                                                                                                             -rw-rw-r--         0:0     2.1 kB  ├── somefile.txt
    2        2.1 kB  /root/example/somefile3.txt                                                                                               drwxr-xr-x         0:0        0 B  ├── srv
                                                                                                                                               drwxr-xr-x         0:0        0 B  ├── sys
                                                                                                                                               drwxrwxrwx         0:0        0 B  ├── tmp
                                                                                                                                               drwxr-xr-x         0:0     176 kB  ├─⊕ usr
                                                                                                                                               drwxr-xr-x         0:0        0 B  └─⊕ var
^C  Quit  ^Space  Switch view  ^/  Filter files  ^L  Show layer changes  ^A  Show aggregated changes
```

Image security

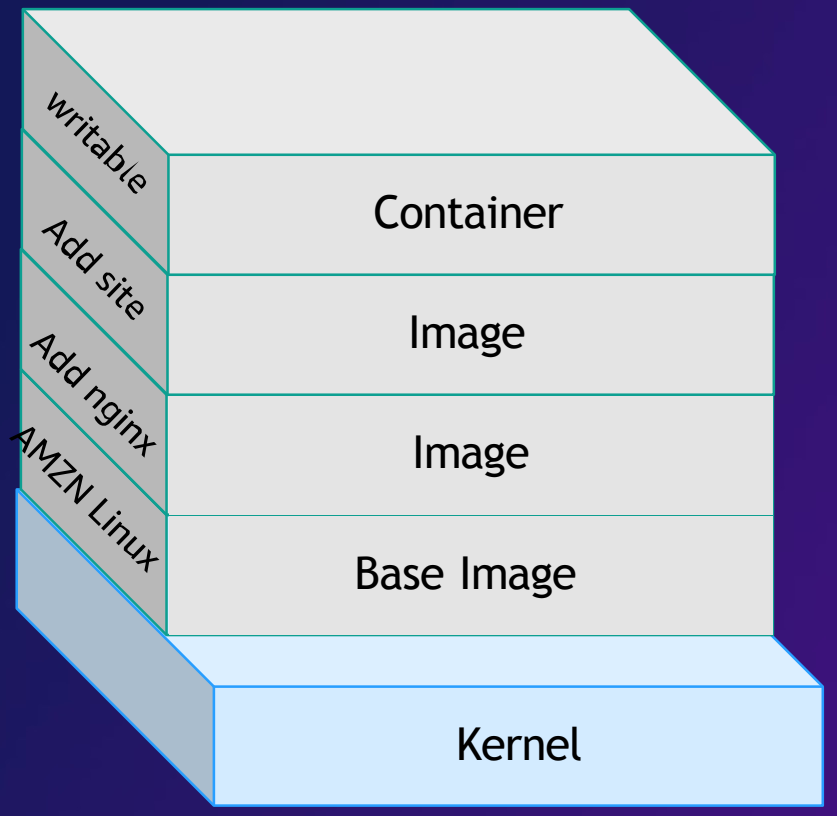
- Scan your images for vulnerabilities (CVEs)
  - ECR Basic Scanning (OS packages)
  - ECR enhanced scanning (OS and programming packages)

- Minimize attack surface
  - Use multi stage build
  - Create images from scratch

- Run the application as a non-root user
  - Lint your Dockerfiles with Dockle or Hadolint

- "Defang" your containers
  - Remove files with the SETUID and SETGID bits from image

- Use endpoint policies and private endpoints with Elastic Container Registry



VPC endpoint policy

VPC > Endpoints > Create endpoint

Create endpoint Info

There are three types of VPC endpoints – Interface endpoints, Gateway Load Balancer endpoints, and Gateway endpoints. Interface endpoints and Gateway Load Balancer endpoints are powered by AWS PrivateLink, and use an Elastic Network Interface (ENI) as an entry point for traffic destined to the service. Interface endpoints are typically accessed using the public or private DNS name associated with the service, while Gateway endpoints and Gateway Load Balancer endpoints serve as a target for a route in your route table for traffic destined for the service.

Endpoint settings

Name tag - *optional*

Creates a tag with a key of 'Name' and a value that you specify.

my-endpoint-01

Service category

Select the service category

- (●) **AWS services** — Services provided by Amazon
- ( ) **PrivateLink Ready partner services** — Services with an AWS Service Ready designation
- ( ) **AWS Marketplace services** — Services that you've purchased through AWS Marketplace
- ( ) **Other endpoint services** — Find services shared with you by service name

Services (1/5)

Filter services 1

search: ecr. ✕ | search: com.amazonaws.ap-northeast-2.s3 ✕ | Clear filters

Service Name	Owner	Type
com.amazonaws.ap-northeast-2.ecr.api	amazon	Interface
com.amazonaws.ap-northeast-2.ecr.dkr	amazon	Interface
com.amazonaws.ap-northeast-2.s3	amazon	Gateway
com.amazonaws.ap-northeast-2.s3	amazon	Interface
com.amazonaws.ap-northeast-2.s3-outposts	amazon	Interface

Create an endpoint policy for your Amazon ECR VPC endpoints

A VPC endpoint policy is an IAM resource policy that you attach to an endpoint when you create or modify the endpoint. If you don't attach a policy when you create an endpoint, AWS attaches a default policy for you that allows full access to the service. An endpoint policy doesn't override or replace IAM user policies or service-specific policies. It's a separate policy for controlling access from the endpoint to the specified service. Endpoint policies must be written in JSON format. For more information, see Controlling Access to Services with VPC Endpoints in the *Amazon VPC User Guide*.

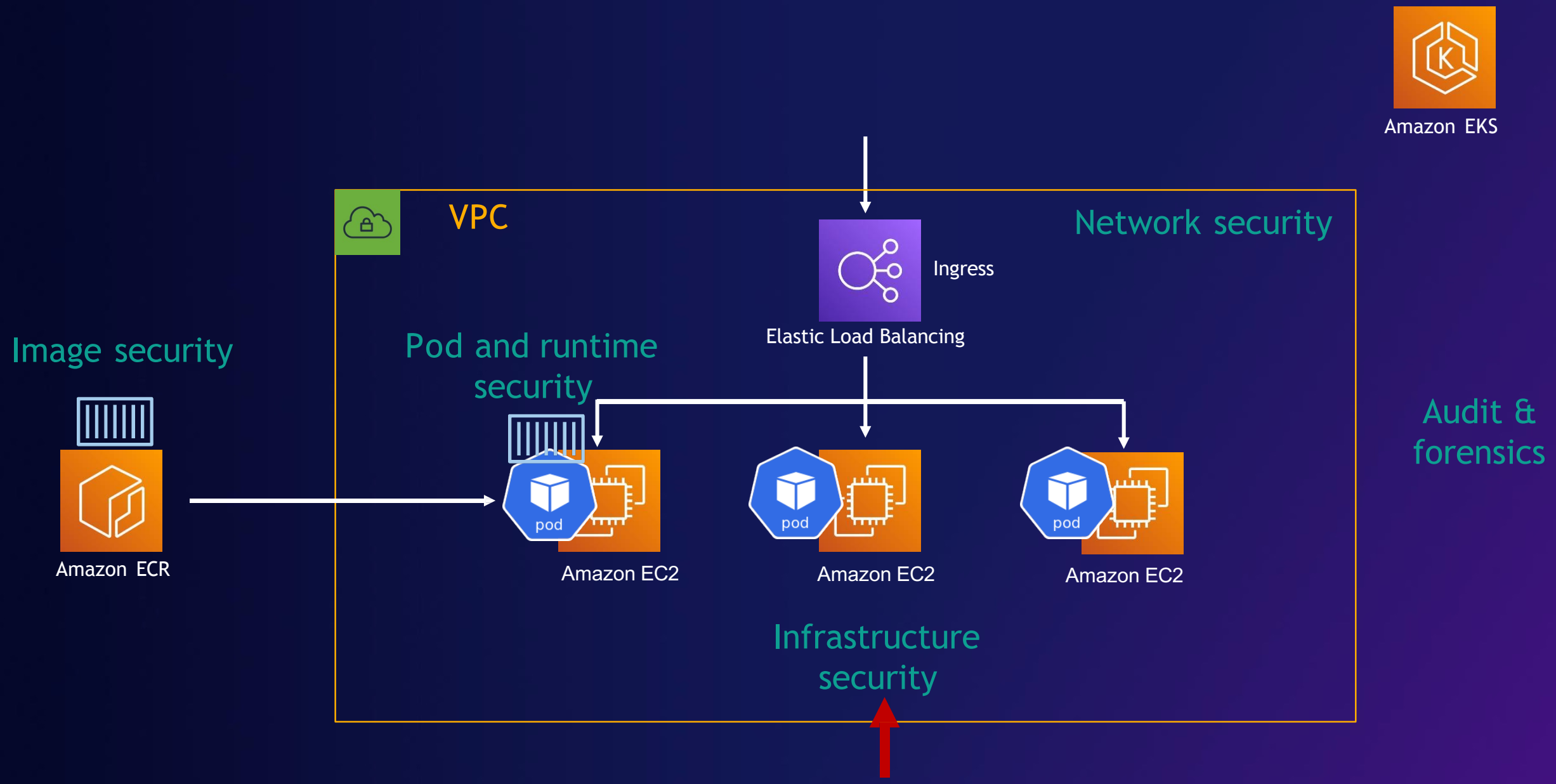
We recommend creating a single IAM resource policy and attaching it to both of the Amazon ECR VPC endpoints.

The following is an example of an endpoint policy for Amazon ECR. This policy enables a specific IAM role to pull images from Amazon ECR.

```
{
        "Statement": [{
                "Sid": "AllowPull",
                "Principal": {
                        "AWS": "arn:aws:iam::1234567890:role/role_name"
                },
                "Action": [
                        "ecr:BatchGetImage",
                        "ecr:GetDownloadUrlForLayer",
                    "ecr:GetAuthorizationToken"
                ],
                "Effect": "Allow",
                "Resource": "*"
        }]
}
```

The following endpoint policy example prevents a specified repository from being deleted.

```
{
        "Statement": [{
                        "Sid": "AllowAll",
                        "Principal": "*",
                        "Action": "*",
                        "Effect": "Allow",
                        "Resource": "*"
                },
                {
                        "Sid": "PreventDelete",
                        "Principal": "*",
                        "Action": "ecr:DeleteRepository",
                        "Effect": "Deny",
                        "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
                }
        ]
}
```



Infrastructure security

- Use an OS that is optimized for running containers
  - Amazon EKS-optimized AL2 or Bottlerocket

- Upgrade to latest AMIs
  - Use Managed Node Group for rolling deployment

Infrastructure security - AMI

amazon-eks-node-1.22-v20220 (4 filtered, 16 unfiltered)

1

Community AMIs
Community AMIs contain all AMIs that are public, therefore anyone can publish an AMI and it will show in this catalog. This catalog can also contain paid products. When using community AMIs it is best practice to ensure you know and trust the publisher before launching an AMI.

aws Verified provider

amazon-eks-node-1.22-v20220824
ami-03f5c5b7ba2015719
EKS Kubernetes Worker AMI with AmazonLinux2 image, (k8s: 1.22.12, docker: 20.10.17-1.amzn2, containerd: 1.6.6-1.amzn2)
Platform: Other Linux | Architecture: x86_64 | Owner: 602401143452 | Publish date: 2022-08-24 | Root device type: ebs | Virtualization: hvm | ENA enabled: Yes
Select

aws Verified provider

amazon-eks-node-1.22-v20220802
ami-09e0355345619d14a
EKS Kubernetes Worker AMI with AmazonLinux2 image, (k8s: 1.22.9, docker: 20.10.13-2.amzn2, containerd: 1.4.13-3.amzn2)
Platform: Other Linux | Architecture: x86_64 | Owner: 602401143452 | Publish date: 2022-08-02 | Root device type: ebs | Virtualization: hvm | ENA enabled: Yes
Select

aws Verified provider

amazon-eks-node-1.22-v20220811
ami-028f26a5a0798bf1e
EKS Kubernetes Worker AMI with AmazonLinux2 image, (k8s: 1.22.12, docker: 20.10.13-2.amzn2, containerd: 1.4.13-3.amzn2)
Platform: Other Linux | Architecture: x86_64 | Owner: 602401143452 | Publish date: 2022-08-11 | Root device type: ebs | Virtualization: hvm | ENA enabled: Yes
Select

aws Verified provider

amazon-eks-node-1.22-v20220914
ami-0167f7c2f2c676793
EKS Kubernetes Worker AMI with AmazonLinux2 image, (k8s: 1.22.12, docker: 20.10.17-1.amzn2, containerd: 1.6.6-1.amzn2)
Platform: Other Linux | Architecture: x86_64 | Owner: 602401143452 | Publish date: 2022-09-14 | Root device type: ebs | Virtualization: hvm | ENA enabled: Yes
Select

AMI Release v20220914

Latest

ljosyula released this 3 days ago · 4 commits to master since this release · v20220914 · 802add4

AMI Release v20220914

- amazon-eks-gpu-node-1.23-v20220914
- amazon-eks-gpu-node-1.22-v20220914
- amazon-eks-gpu-node-1.21-v20220914
- amazon-eks-gpu-node-1.20-v20220914
- amazon-eks-arm64-node-1.23-v20220914
- amazon-eks-arm64-node-1.22-v20220914
- amazon-eks-arm64-node-1.21-v20220914
- amazon-eks-arm64-node-1.20-v20220914
- amazon-eks-node-1.23-v20220914
- amazon-eks-node-1.22-v20220914
- amazon-eks-node-1.21-v20220914
- amazon-eks-node-1.20-v20220914

Release versions for these AMIs:

- `1.23.9-20220914`
- `1.22.12-20220914`
- `1.21.14-20220914`
- `1.20.15-20220914`

Binaries used to build these AMIs are published:

- s3://amazon-eks/1.23.9/2022-07-27/
- s3://amazon-eks/1.22.12/2022-07-27/
- s3://amazon-eks/1.21.14/2022-07-27/
- s3://amazon-eks/1.20.15/2022-07-27/

AMI details:

- kernel: 5.4.209-116.367.amzn2
- dockerd: 20.10.17-1.amzn2
- containerd: 1.6.6-1.amzn2
- runc: 1.1.3-1.amzn2
- cuda: 470.57.02-1
- nvidia-container-runtime-hook: 1.4.0-1.amzn2
- SSM agent: 3.1.1732.0-1.amzn2

https://github.com/awslabs/amazon-eks-ami/releases

Infrastructure security - Managed Node Group

eks-121-proxy02

Delete cluster

Your current user or role does not have access to Kubernetes objects on this EKS cluster
This may be due to the current user or role not having Kubernetes RBAC permissions to describe cluster resources or not having an entry in the cluster's auth config map. Learn more

New Kubernetes versions are available for this cluster. Learn more

Update now

Cluster info Info

Kubernetes version Info	Status	Provider
1.21	Active	EKS

Overview | Resources | Compute | Networking | Add-ons | Authentication | Logging | Update history | Tags

Nodes (0) Info

Filter Nodes by property or value

Node name	Instance type	Node group	Created	Status

No Nodes
This cluster does not have any Nodes, or you don't have permission to view them.

Node groups (1) Info

Edit | Delete | Add node group

Group name	Desired size	AMI release version	Launch template	Status
managed-ng01	0	1.21.12-20220629 Update now	eksctl-eks-121-proxy02-nodegroup-managed-ng01 (1)	Active

Update node group version: managed-ng01

Update node group version
Update your node group AMI release version to the latest available version for your 1.21 cluster.

Change launch template version
Change your node group to use a different launch template version.

Update strategy
Select how to update this node group.

Rolling update

This option respects pod disruption budgets for your cluster. The update fails if Amazon EKS is unable to gracefully drain the pods that are running on this node group due to a pod disruption budget issue.

Learn more

AMI release version notes

Cancel | Update

Infrastructure security

- Use an OS that is optimized for running containers
  - Amazon EKS-optimized AL2 or Bottlerocket

- Upgrade to latest AMIs
  - Use Managed Node Group for rolling deployment

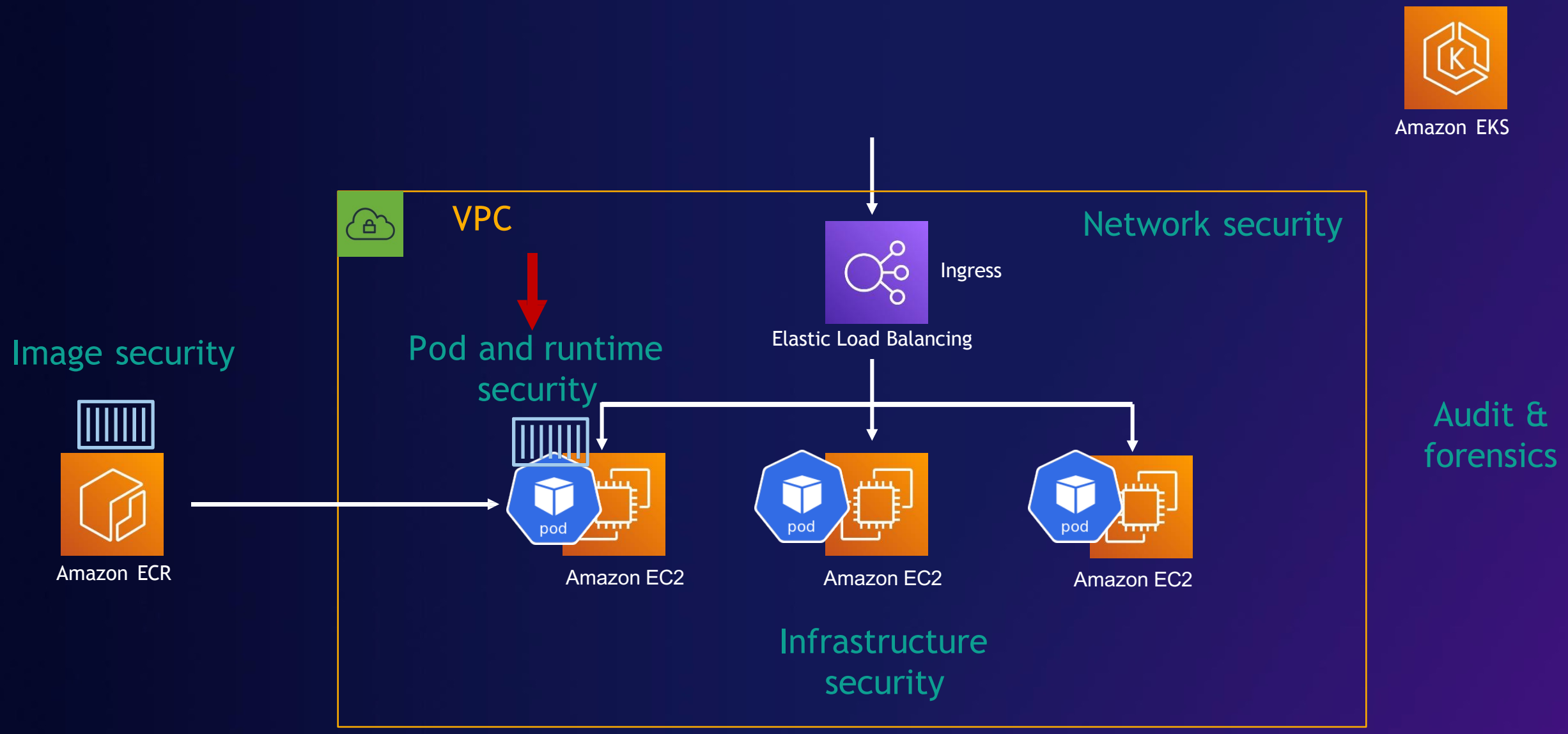
- Deploy workers onto private subnets

Infrastructure security

- Use an OS that is optimized for running containers
  - Amazon EKS-optimized AL2 or Bottlerocket
- Upgrade to latest AMIs
  - Use Managed Node Group for rolling deployment
- Deploy workers onto private subnets
- Minimize and audit host access
  - AWS Systems Manager, Session Manager

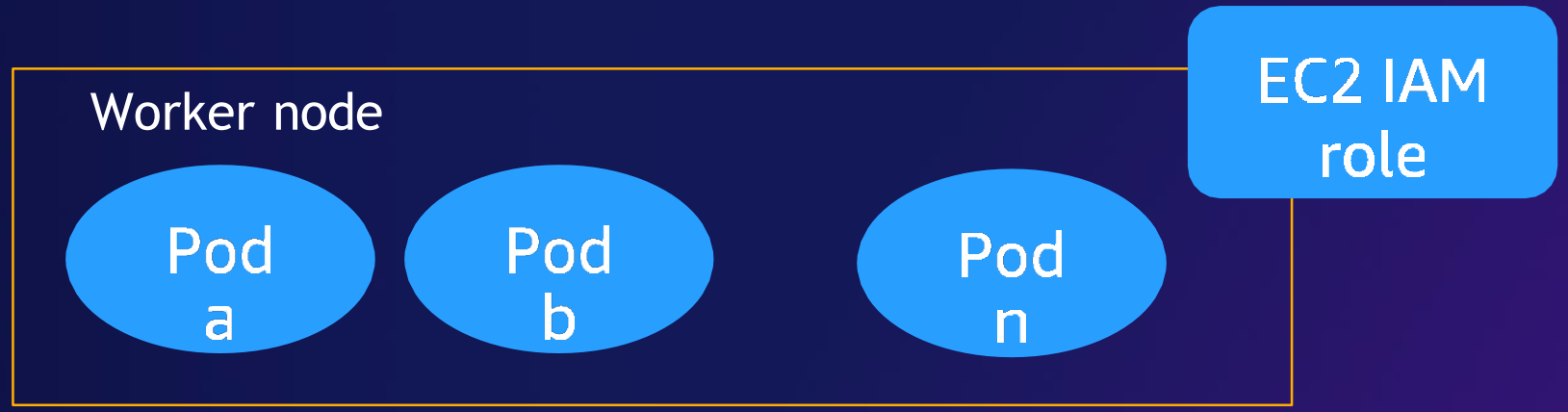
Infrastructure security

- Use an OS that is optimized for running containers
  - Amazon EKS-optimized AL2 or Bottlerocket
- Upgrade to latest AMIs
  - Use Managed Node Group for rolling deployment
- Deploy workers onto private subnets
- Minimize and audit host access
  - AWS Systems Manager, Session Manager
- Run kube-bench to continually assess alignment with Amazon EKS CIS benchmarks

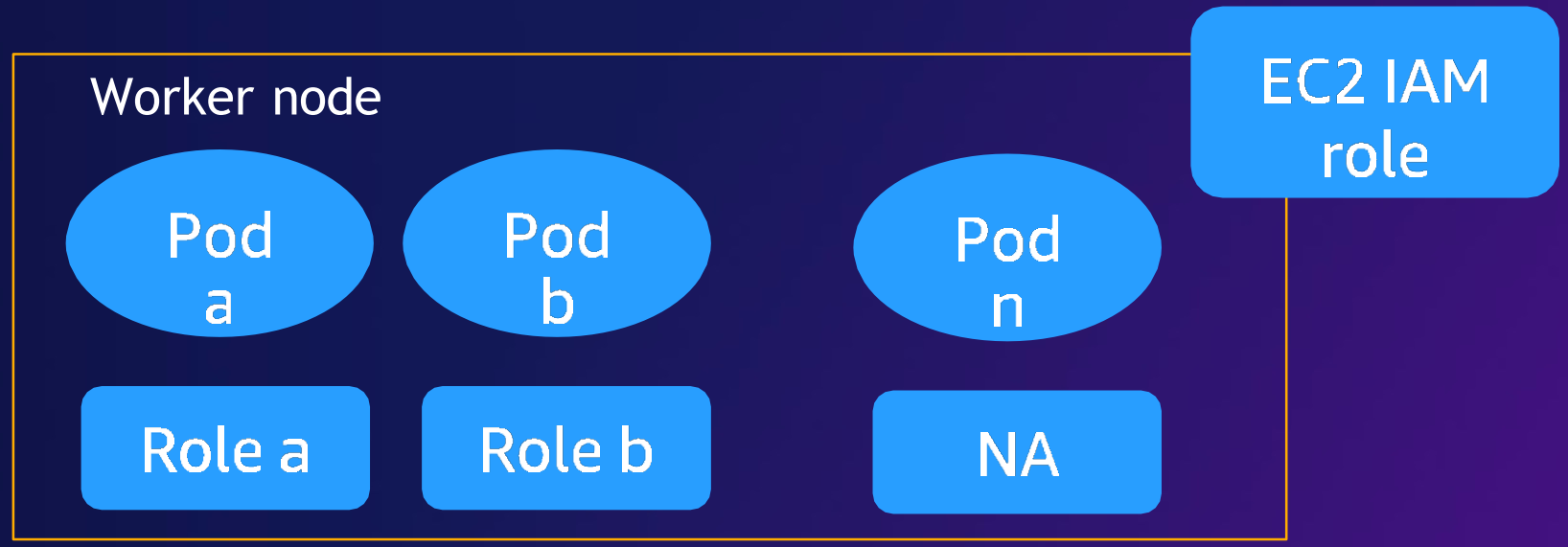


IAM roles for service accounts

No IRSA



IRSA configured

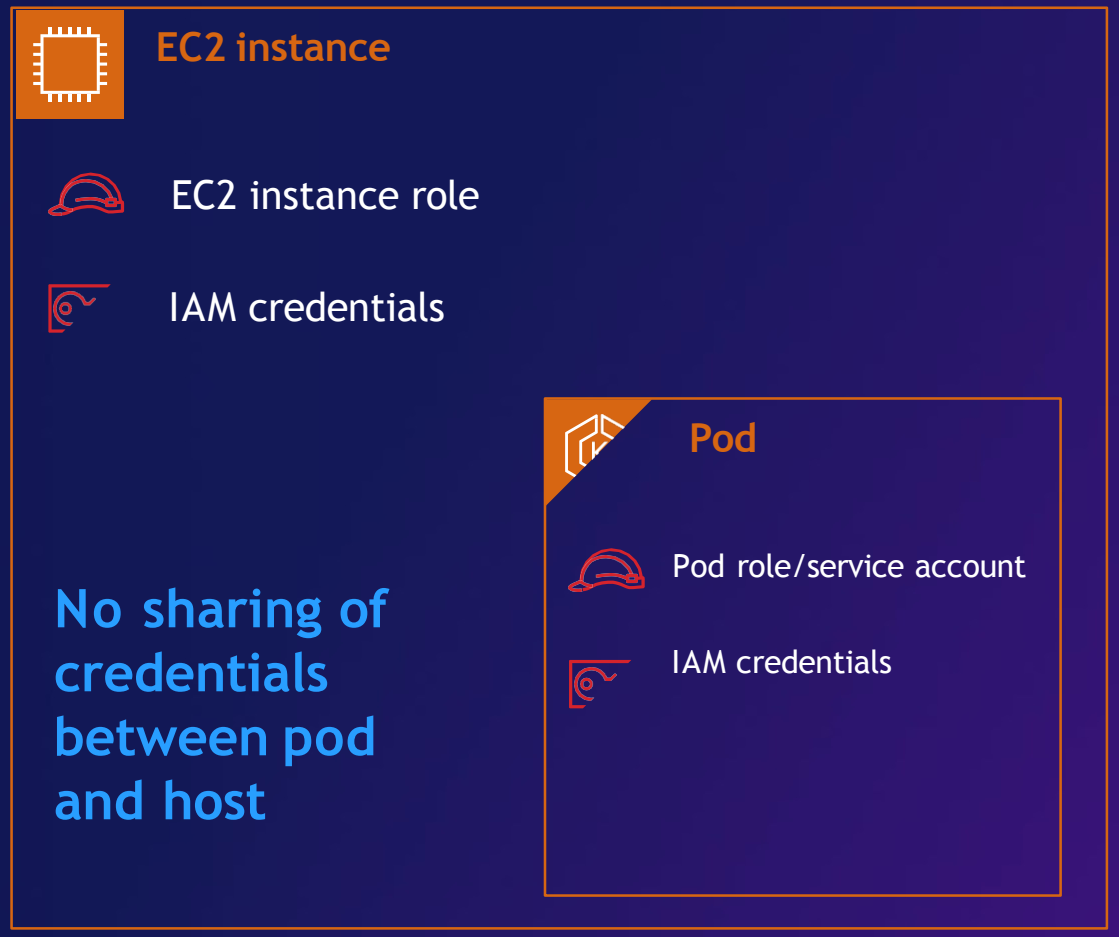


Use IAM roles for service accounts

Map IAM roles to Kubernetes service accounts

Enable least privilege for Kubernetes pods

Natively supported by Amazon EKS and available as open source



Pod and runtime security

- Implement IRSA (IAM roles for service accounts)

- Run dynamic scans – Falco, Prisma Twistlock, Aqua, etc.

Pod and runtime security – Falco

```
[ec2-user@ip-10-0-1-211 wordpress]$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
falco-25rvh               1/1     Running   2          4d10h
falco-4w2gb               1/1     Running   1          4d10h
falco-nnk4f               1/1     Running   1          4d10h
falco-rlf7j               1/1     Running   2          4d10h
falco-wsv7d               1/1     Running   1          4d10h
fluentbit-29sz8           1/1     Running   7          4d10h
fluentbit-6nsw9           1/1     Running   7          4d10h
fluentbit-9smnq           1/1     Running   7          4d10h
fluentbit-lxx4z           1/1     Running   1          4d10h
fluentbit-rk45t           1/1     Running   7          4d10h
nginx2-78848c9dcb-5ssrm   1/1     Running   1          4d10h
nginx2-78848c9dcb-dsnch   1/1     Running   1          4d10h
nginx2-78848c9dcb-r7z64   1/1     Running   1          4d11h
```

```
[ec2-user@ip-10-0-1-211 wordpress]$ kubectl exec -it nginx2-78848c9dcb-5ssrm -- /bin/sh
# touch /etc/hacker
# cd /bin/ ; mkdir tools
# cat /etc/shadow > /dev/null 2>&1
```


```
2021-07-02T10:39:34.401+09:00    {"log":"01:39:34.377084653: Notice A shell was spawned in a container with an attached terminal (user=root user_loginuid=-1 k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee shell=sh parent=runc cmdline=sh terminal=34…

{
    "log": "01:39:34.377084653: Notice A shell was spawned in a container with an attached terminal (user=root user_loginuid=-1 k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee shell=sh parent=runc cmdline=sh terminal=34816
container_id=b62164264cee image=nginx) k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee\n",
    "stream": "stdout"
}
```


Copy


```
2021-07-02T10:39:45.128+09:00    {"log":"01:39:45.118927771: Error File below /etc opened for writing (user=root user_loginuid=-1 command=touch /etc/hacker parent=sh pcmdline=sh file=/etc/hacker program=touch gparent=\u003cNA\u003e ggparent=\u003cNA\u003e gggparent=\u00…

{
    "log": "01:39:45.118927771: Error File below /etc opened for writing (user=root user_loginuid=-1 command=touch /etc/hacker parent=sh pcmdline=sh file=/etc/hacker program=touch gparent=<NA> ggparent=<NA> gggparent=<NA> container_id=b62164264cee
image=nginx) k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee\n",
    "stream": "stdout"
}
```


Copy


```
2021-07-02T10:41:03.340+09:00    {"log":"01:41:03.323755803: Error Directory below known binary directory created (user=root user_loginuid=-1 command=mkdir tools directory=/bin/tools container_id=b62164264cee image=nginx) k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm c…

{
    "log": "01:41:03.323755803: Error Directory below known binary directory created (user=root user_loginuid=-1 command=mkdir tools directory=/bin/tools container_id=b62164264cee image=nginx) k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm
container=b62164264cee k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee\n",
    "stream": "stdout"
}
```


Copy


```
2021-07-02T10:41:32.865+09:00    {"log":"01:41:32.840168387: Warning Sensitive file opened for reading by non-trusted program (user=root user_loginuid=-1 program=cat command=cat /etc/shadow file=/etc/shadow parent=sh gparent=\u003cNA\u003e ggparent=\u003cNA\u003e gggpar…

{
    "log": "01:41:32.840168387: Warning Sensitive file opened for reading by non-trusted program (user=root user_loginuid=-1 program=cat command=cat /etc/shadow file=/etc/shadow parent=sh gparent=<NA> ggparent=<NA> gggparent=<NA> container_id=b62164264cee
image=nginx) k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee k8s.ns=default k8s.pod=nginx2-78848c9dcb-5ssrm container=b62164264cee\n",
    "stream": "stdout"
}
```


Copy

Pod and runtime security – Polaris by Fairwinds

polaris Docs

Overview Insights Categories Namespaces

Cluster Overview: https://10.100.0.1:443

Smooth sailing within sight

Grade: C

Score: 73%

Score is the percentage of passing checks. Warnings get half the weight of dangerous checks.

243 passing checks

96 warning checks

41 dangerous checks

! Some checks were skipped based on configured exemptions. Click here to view the report with these checks included.

Kubernetes Version: 1.21+ Nodes: 3 Namespaces: 7 Controllers: 20 Pods: 0

https://github.com/FairwindsOps/polaris

Pod and runtime security

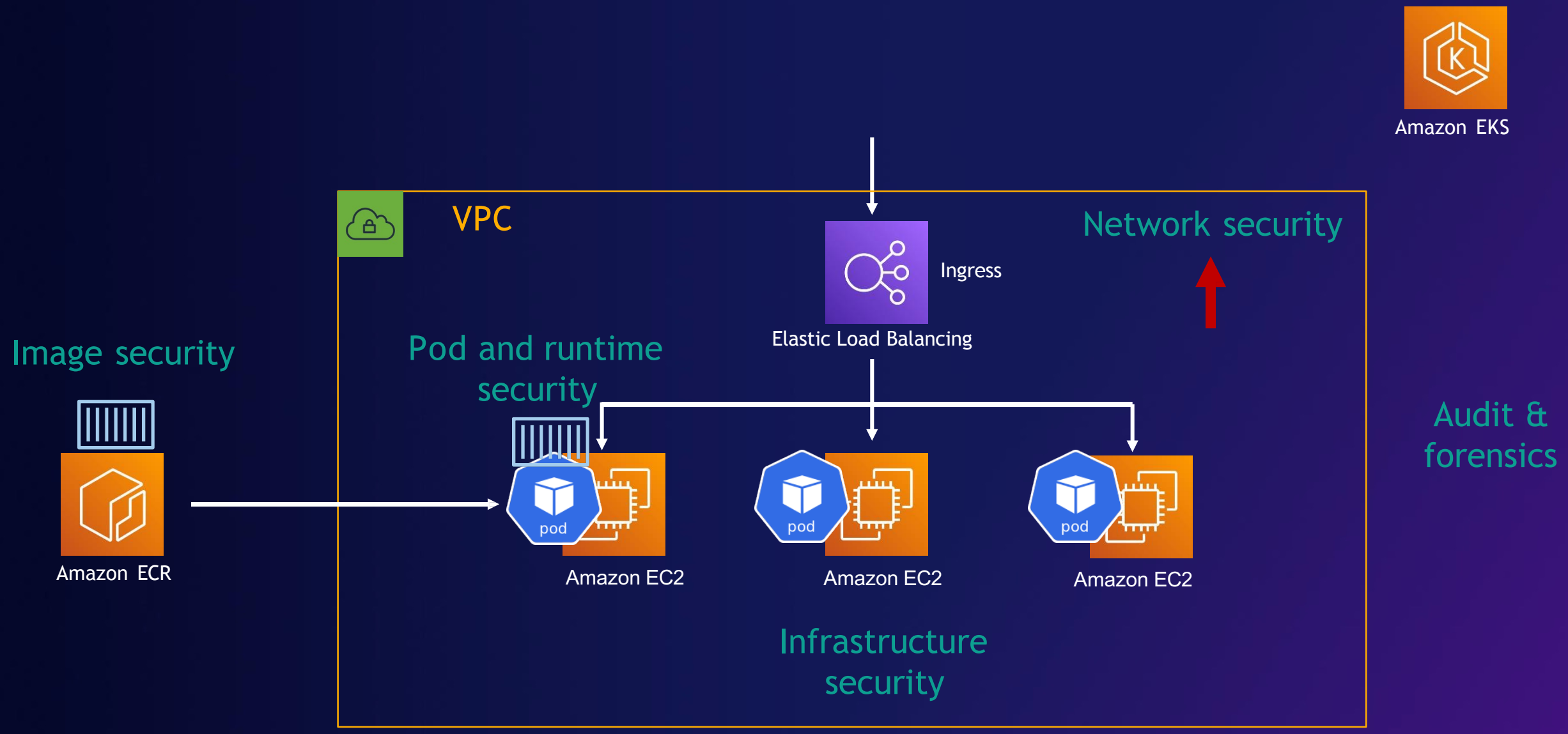
- Implement IRSA (IAM roles for service accounts)
- Run dynamic scans – Falco, Prisma Twistlock, Aqua, etc.
- Use Pod Security Standards or Policy as code
  - Deny privileged escalation
  - Deny running as root
  - Drop Linux capabilities
  - Implement OPA with Gatekeeper
  - Use other tools like Kyverno

Pod and runtime security - Container and pods

- Dedicate service account for each application
- Do not allow privileged escalation
- Containers must be required to run as non-root
- Restrict the containers that can run as privileged
- Drop capabilities – default capabilities are not as strict as you think

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: security-pod
spec:
  serviceAccountName: sec-pod
  containers:
  - name: my-pod
    image: node
    securityContext:
      allowPrivilegeEscalation: false
      runAsUser: 1000
      runAsNonRoot: true
      privileged: false
      capabilities:
        drop:
          - all
```

aws

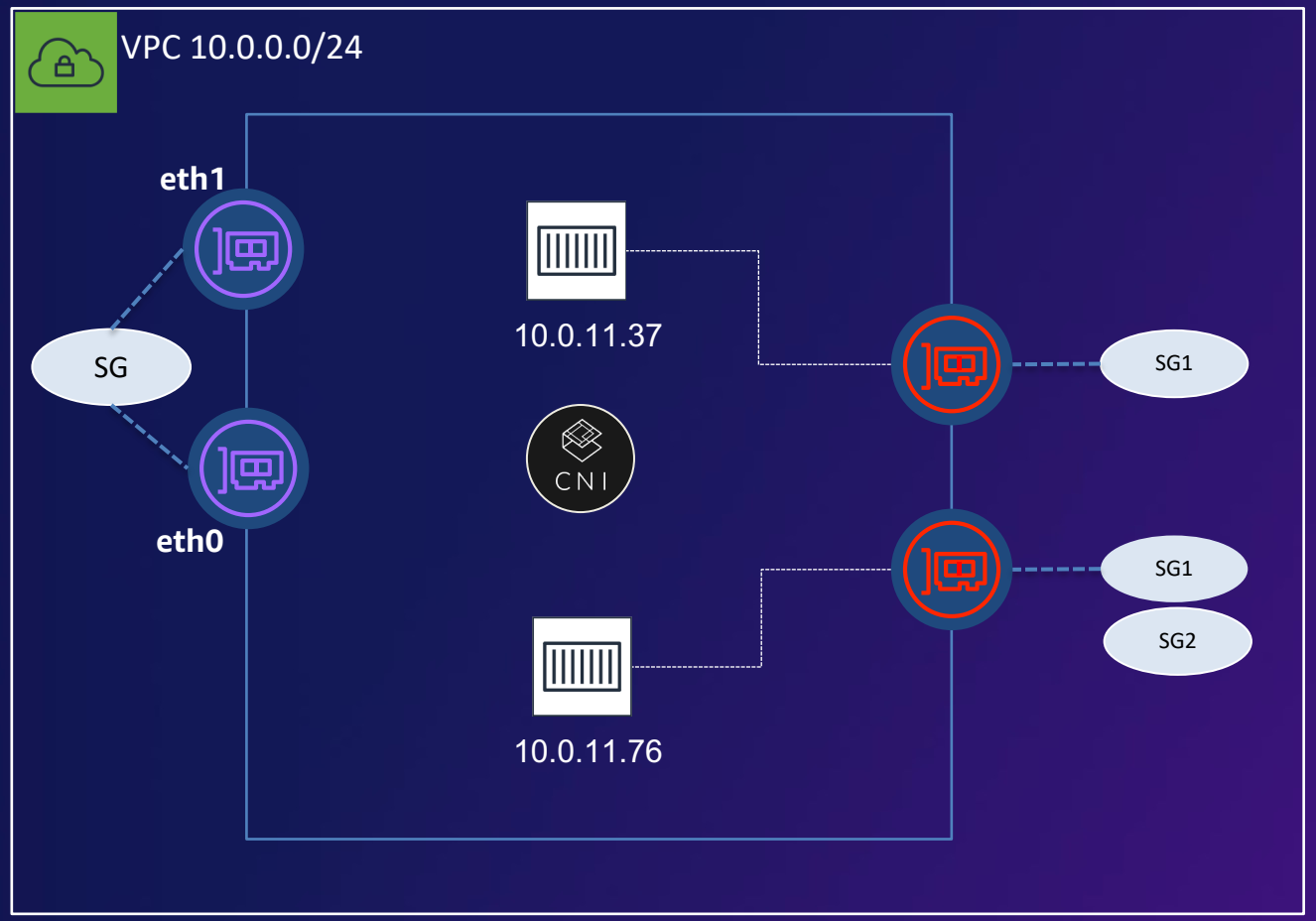


Network security

- Use security groups for pods to restrict traffic to AWS resources

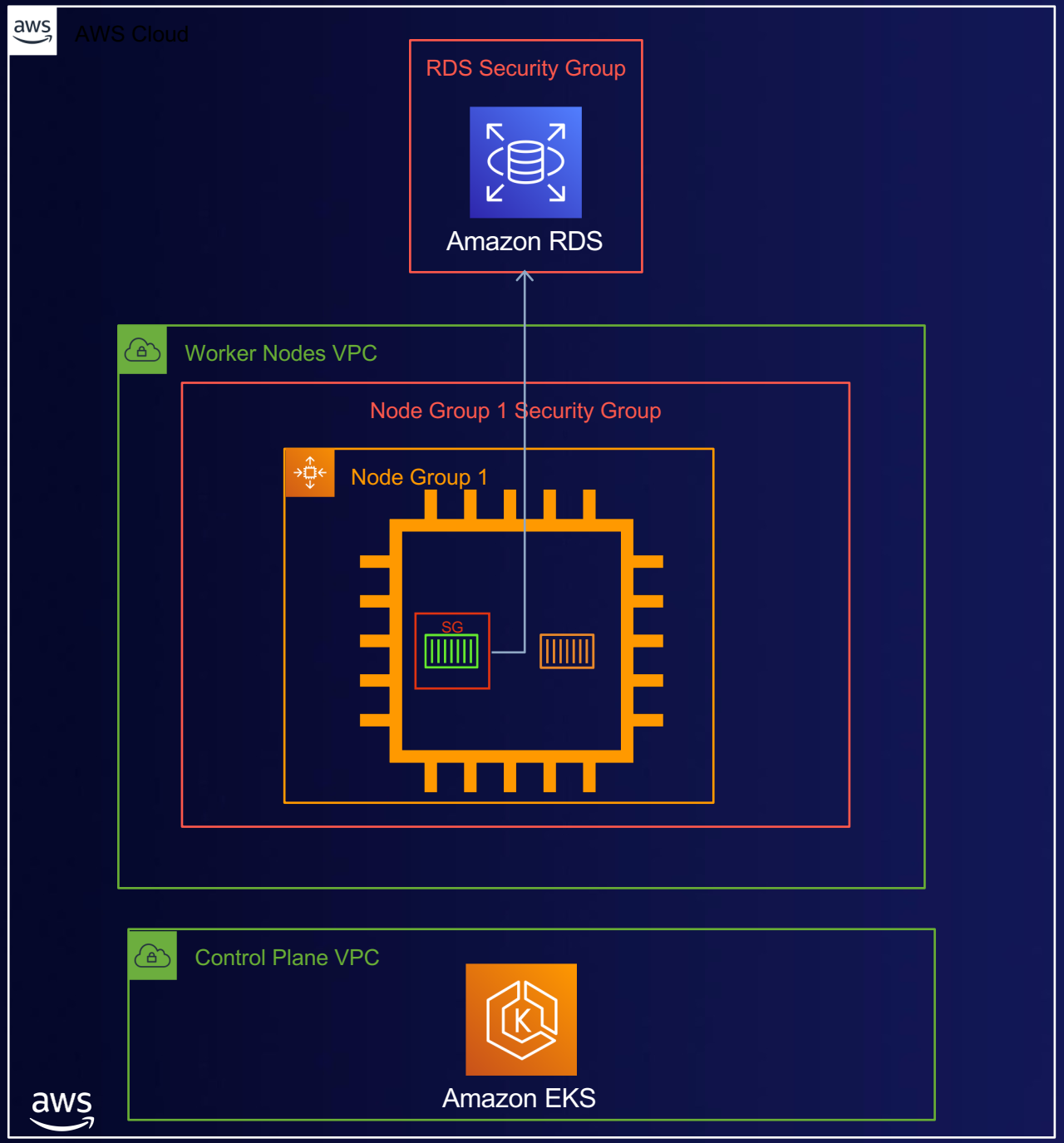
Securing Pods with EC2 Security Groups

- Worker node and all its Pods share the same security group(s) attached to the primary ENI
- Assign each Pod to a dedicated ENI with its own separate set of security groups(s)



Worker Node
10.0.11.205

aws



RDS Security Group

Type	Protocol	Ports	Source
PostgreSQL	TCP	5432	sg-yyyyyyy (POD)

API Extension with new CRD

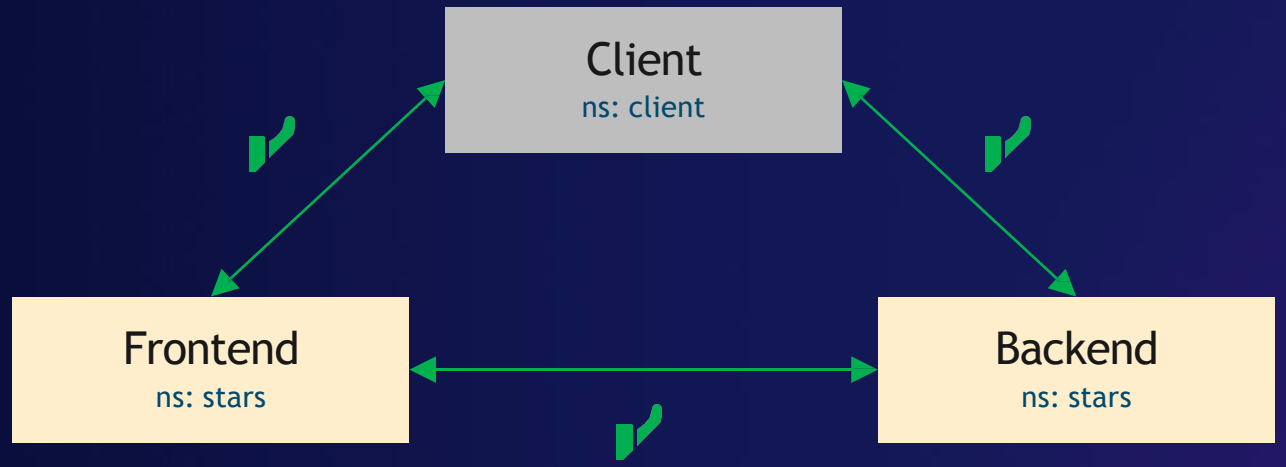
```yaml
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: RDSClient
spec:
  serviceAccountSelector:
    matchLabels:
      role: RDSClient
  securityGroups:
    groupIds:
      - sg-yyyyyy
```

Network security

- Use security groups for pods to restrict traffic to AWS resources
- Use k8s network policies to restrict traffic within cluster

Network policy example

- Initial state without any network policy in effect



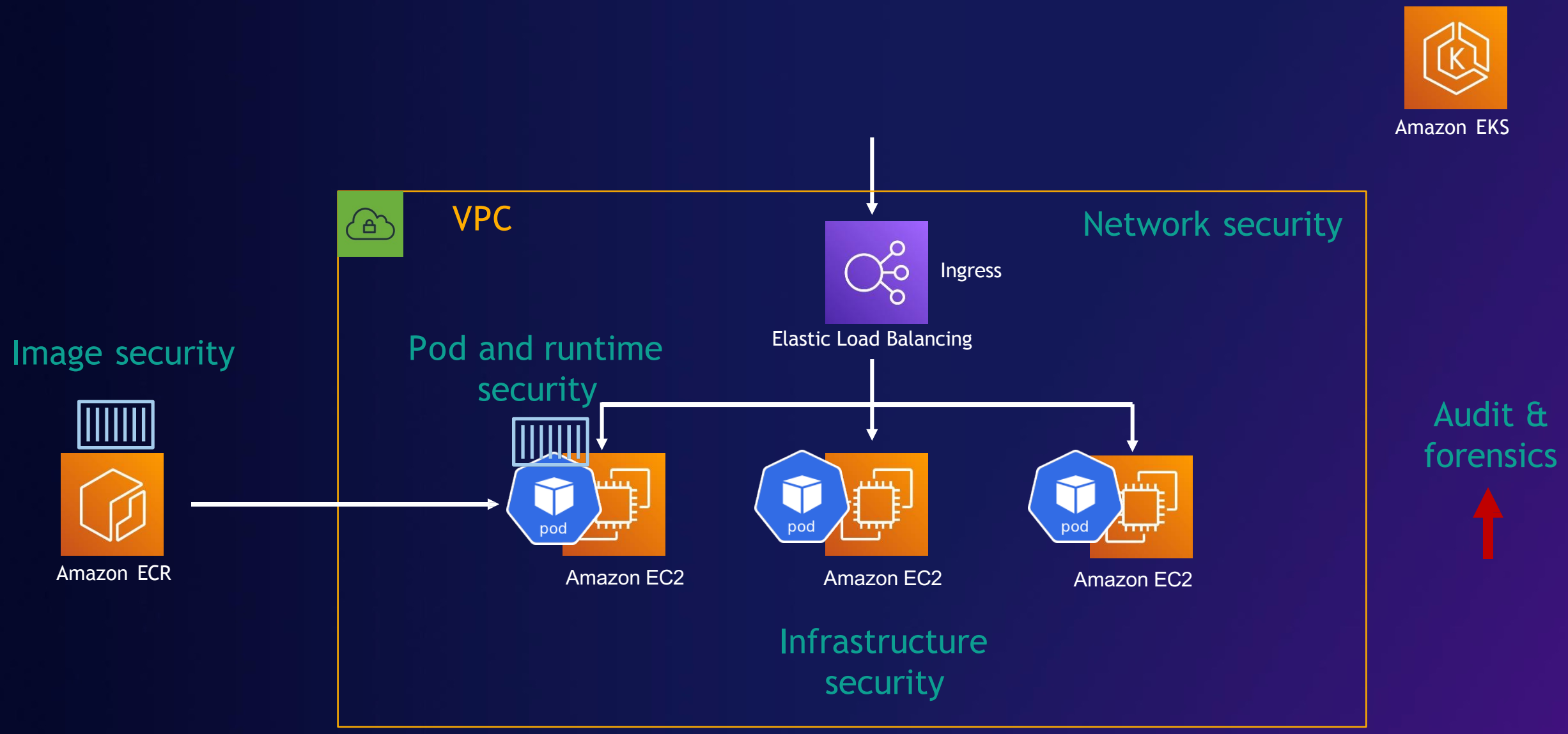
Network policy: Ingress rule

- Network policy to allow traffic to ingress into Backend pods from Frontend pods


```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

Network security

- Use security groups for pods to restrict traffic to AWS resources
- Use k8s network policies to restrict traffic within cluster
  - Enforce using network policy engine like Calico
- Encryption in transit
  - Terminate HTTPS external traffic at ELB
  - Use service mesh like App Mesh for mTLS service-service communication



Auditing and forensics

- Enable control plane logs
- Stream logs from containers to external log aggregator
- Periodically audit control plane and AWS CloudTrail logs for suspicious activity
- 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.

EKS security best practices guide

EKS Best Practices Guides

Search

EKS Best Practices Guides

Guides

Security

Home

Identity and Access Management

Pod Security

Multi-tenancy

Detective Controls

Network Security

Data Encryption and Secrets Management

Runtime Security

Infrastructure Security

Regulatory Compliance

Incident Response and Forensics

Image Security

Amazon EKS Best Practices Guide for Security

This guide provides advice about protecting information, systems, and assets that are reliant on EKS while delivering business value through risk assessments and mitigation strategies. The guidance herein is part of a series of best practices guides that AWS is publishing to help customers implement EKS in accordance with best practices. Guides for Performance, Operational Excellence, Cost Optimization, and Reliability will be available in the coming months.

How to use this guide

This guide is meant for security practitioners who are responsible for implementing and monitoring the effectiveness of security controls for EKS clusters and the workloads they support. The guide is organized into different topic areas for easier consumption. Each topic starts with a brief overview, followed by a list of recommendations and best practices for securing your EKS clusters. The topics do not need to be read in a particular order.

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

Thank you!