NeuVector

kubernetes

How to Promote the use of Best Practices and Automate Security Policies Using Tools Like OPA and Kubernetes Native Declaratives

Gary Duan
Co-Founder & CTO, NeuVector

Agenda

Build Security Into Your Pipeline

Security as Code, CRD and OPA

- Demo 1: OPA and Vulnerability Scan

Kubernetes Admission Control

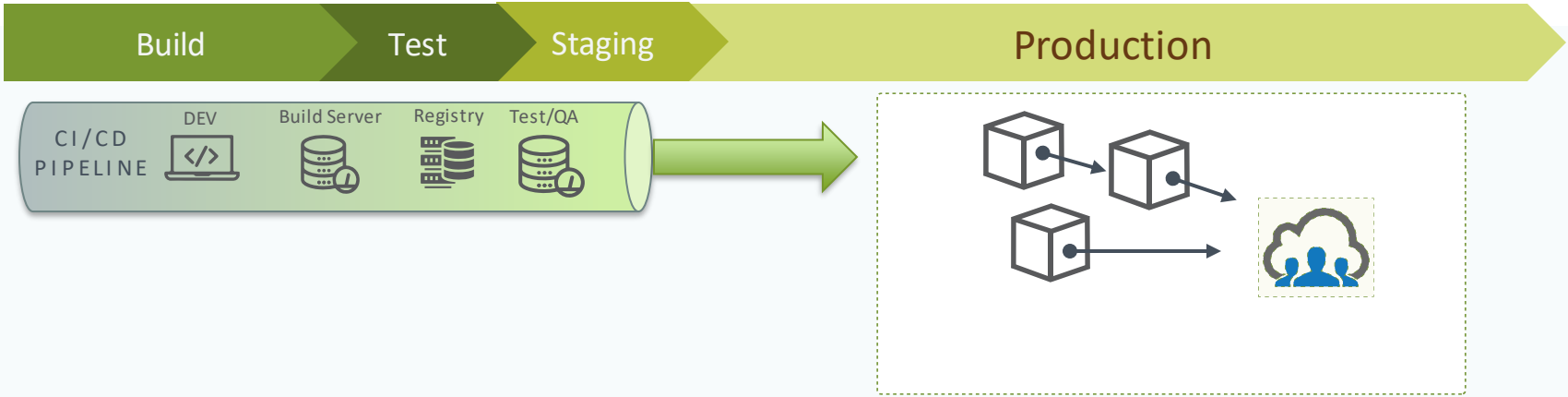
- Demo 2: OPA to Audit Admission Control

Kubernetes Run-Time Security

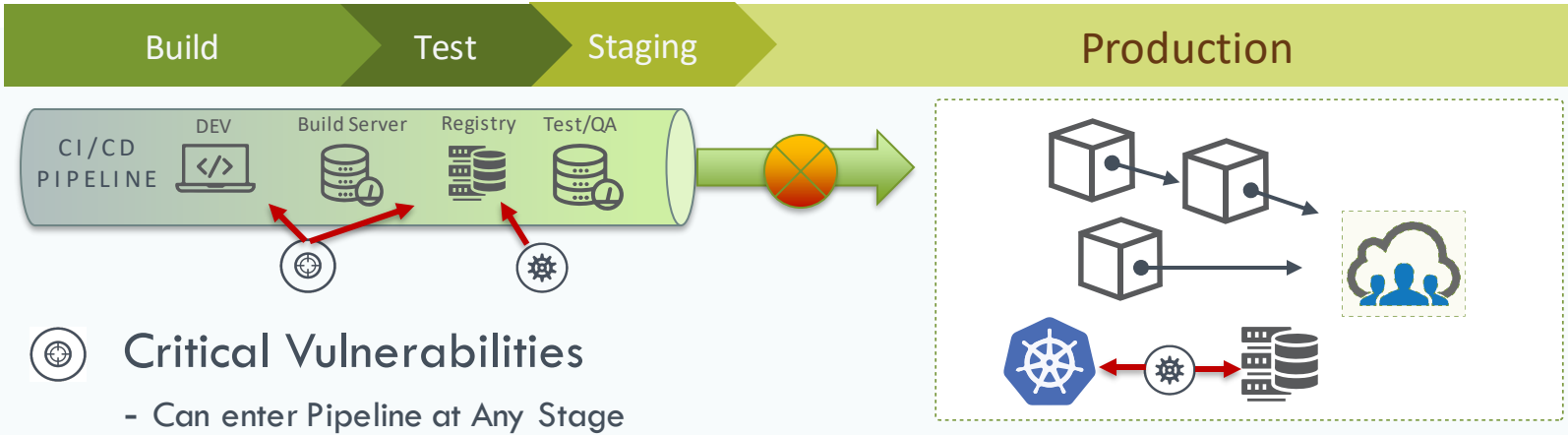
- Demo 3: OPA to Audit Run-Time Policies
- Security Posture Management and Virtual Patching

Q&A

Cloud-Native Pipeline



Top Security Issues for Cloud-Native Deployments



Critical Vulnerabilities

- Can enter Pipeline at Any Stage

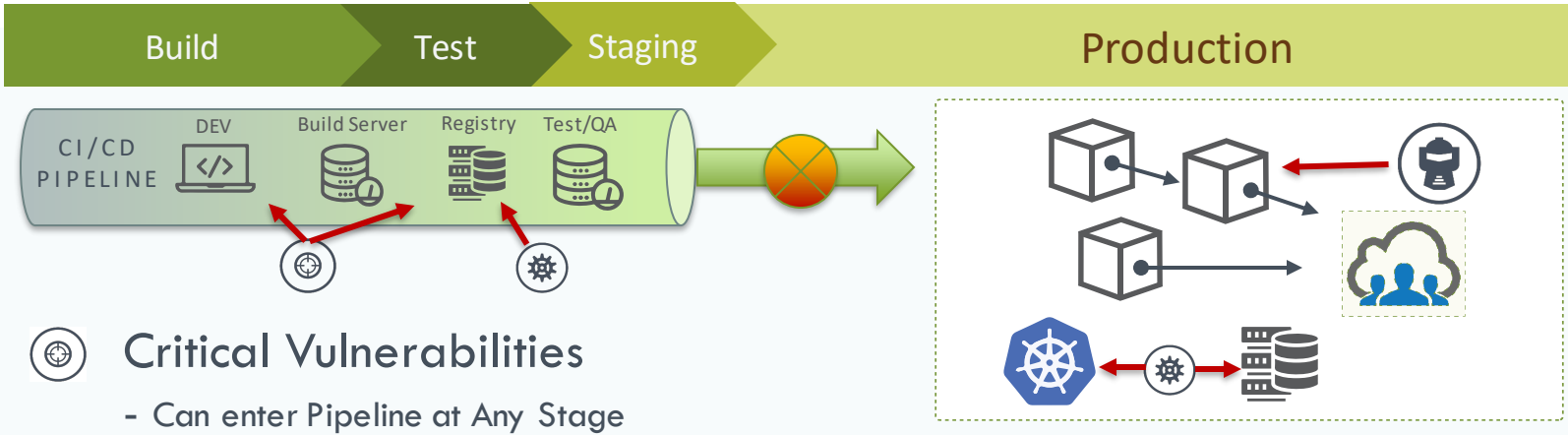
Security Slows Pipeline

- Manual Configuration, Lack of Integration

Misconfiguration

- Registries, Kubernetes, Container Hosts

Top Security Issues for Cloud-Native Deployments



Critical Vulnerabilities

- Can enter Pipeline at Any Stage

Security Slows Pipeline

- Manual Configuration, Lack of Integration

Misconfiguration

- Registries, Kubernetes, Container Hosts

New Attack Surfaces

- Kubernetes, Docker, Tool Vulnerabilities

Inadequate Protection in Production

- Container Exploits, Zero day, Insiders

\>whatis Security as Code

*Everything in Kubernetes is a **declarative configuration object** that represents the **desired state** of the system.*

- Kubernetes Up and Running - Burns / Beda / Hightower - O'Reilly Media

Security **stored in declarative configurations** and **delivered** as part of a Continuous Delivery pipeline can be referred to as Security-as-code.

Challenges for Security as Code in Kubernetes

- What can be / should be a declarative configuration?
- Rapid-deployment in a small DevOps team
- MicroService Communication & Network Visibility

Kubernetes Custom Resource Definition (CRD)

A ***resource*** is an endpoint in Kubernetes' API that stores an API object

A ***custom resource*** is an extension of the Kubernetes API for custom API objects

RBAC-enabled: ***namespace*** and ***cluster*** level resources

Introduction to Open Policy Agent (OPA)

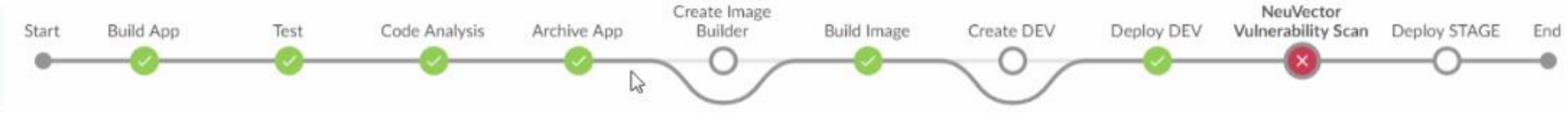
What is it?

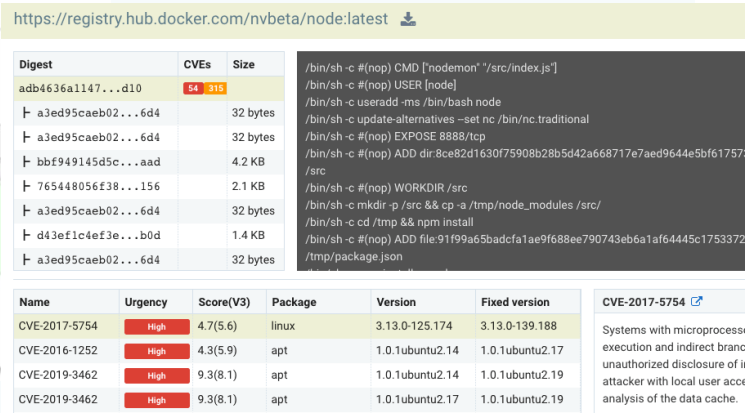
- A general-purpose policy engine
- Offers declarative language to describe the policy
- Decouple the decision-making and the policy enforcement
- Key component of security-as-code workflow

Use Cases

- Kubernetes Admission Control Gatekeeper
- Service Mesh Authorization Policy
- Policy Auditing

CI/CD Pipeline Integration – Vulnerability Scanning





- ✓ Automated Build Phase Scanning
- ✓ Scanning Public Cloud Registries, Private Registries, 3rd Party Registries
- ✓ Layered Image Vulnerability Analysis

OPA and Vulnerability Scan

Policy

- Image must not have more than 2 vulnerabilities

audit.rego

```
1  package psp
2
3  default allow = false
4
5  allow = true {
6      count(input.report.vulnerabilities) < 3
7  }
8
9
```

OPA and Admission Control

A) Pod Security Policy (PSP)
- Cluster-level resource, authorized to a service account
- Priviledged, run-as-root
- Volume, host path/namespace, read-only root fs
- Linux caps, seccomp, SELinux/AppArmor

B) NeuVector Admission Control
- Monitor and Enforce mode
- Leverage container image vulnerability scan results
- CVE severity, fix status, date and software component
- Leverage container image compliance checks

Audit Pod Security Profile

Policy

- No pod can run in privileged mode except for those in the "admin" namespace

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: psp.restricted
spec:
  privileged: false
  allowPrivilegeEscalation: false
  runAsUser:
    rule: 'MustRunAsNonRoot'
```

psp.rego

psp > psp.rego

```
package psp

import data.roles.bindings as bindings
import data.roles.permissions as role

default allow = false

allow = true {
    some spec
    profile[spec]
    spec.privileged = false
}

allow = true {
    some spec
    profile[spec]
    spec.privileged != false
    input.ServiceAccount.namespace = "admin"
}

profile[spec] {
    some i
    spec := data.spec

    bindings.subjects[i].namespace = input.ServiceAccount.namespace
    bindings.subjects[i].name = input.ServiceAccount.name

    role.metadata.name = bindings.roleRef.name

    data.metadata.name = role.rules[_].resourceNames[_]
}
```

Audit Admission Control Rules

Policy

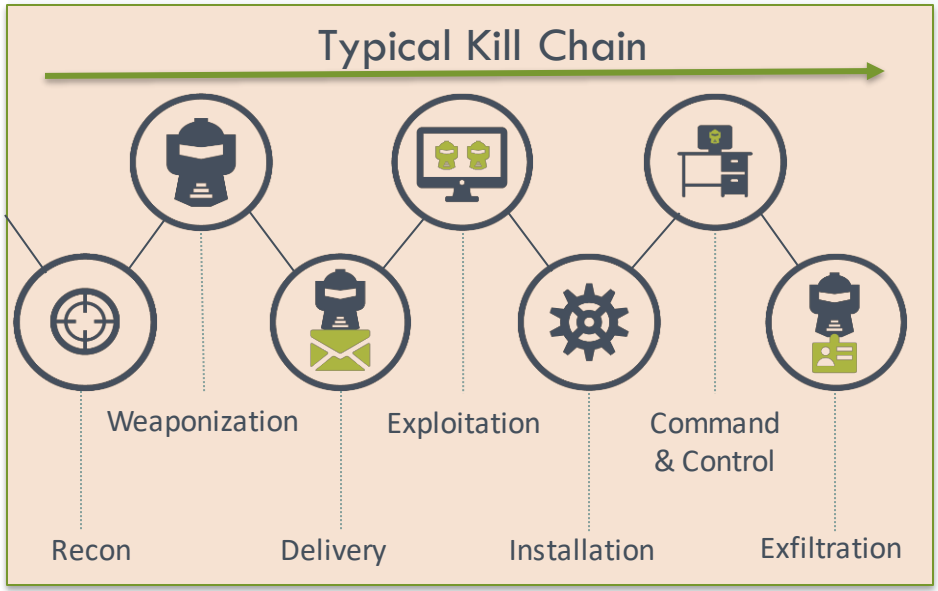
- Number of high severity vulnerabilities that have been reported for more than 30 days must be less than 5
- No restriction in the "staging" namespace

audit.rego

admission > audit.rego

```
package admission

default allow = false

allow = true {
    some i
    r := input.rules[i]
    r.disable = false
    r.rule_type = "deny"

    some c1
    r.criteria[c1].name = "namespace"
    r.criteria[c1].op = "notContainsAny"
    r.criteria[c1].value = "staging"

    some c2
    r.criteria[c2].name = "cveHighWithFixCount"
    r.criteria[c2].op = ">="
    r.criteria[c2].value = "5"

    some c2s
    r.criteria[c2].sub_criteria[c2s].name = "publishDays"
    r.criteria[c2].sub_criteria[c2s].op = ">="
    r.criteria[c2].sub_criteria[c2s].value = "30"
}
```

Run-Time Protection – Critical Attack Prevention in Production



	Network	Container	Host
Cryptojacking Bitcoin Mining	✓		✓
Port Scan	✓		
EQUIFAX DATA BREACH	✓	✓	
DDOS ATTACK	✓		
SQL	✓		
DIRTY COW		✓	✓
mongoDB	✓	✓	

Using OPA to Audit Run-Time Policies

1. Kubernetes Network Policies
   - Namespace scoped, pods selected by labels
   - Pod-to-pod policy. Limited flexibility of cluster ingress/egress.
   - Defined by protocols and ports
2. NeuVector CRD Policies
   - Monitor and Protect mode
   - Pods selected by service, label, namespace …
   - Pod-to-pod, pod-to-host and cluster ingress/egess
   - RBAC-aware, global policy
   - CIDR, FQDN and L7 protocols
   - Network, process and file system activities

Audit Kubernetes Network Security

Policy

- Applications in the demo namespace can only listen on at most 1 port
- Applications in the demo namespace cannot make outbound connections

audit.rego

```
package policy

default allow = false

allow = true {
    data.kind = "NetworkPolicy"
    data.metadata.namespace = "demo"
    ingress_port
    egress_exist
}

ingress_port = true {
    count(data.spec.ingress[0].ports) <= 1
}

ingress_port = true {
    not data.spec.ingress[0].ports
}

egress_exist = true {
    e := data.spec.egress[_]
    e.to[0].podSelector
}

egress_exist = true {
    e := data.spec.egress[_]
    e.to[0].namespaceSelector
}
```

Run-time Policy CRD Explained

- ✓ Define Application Behaviors in Kubernetes-native yaml
  - Network Connections and Protocols
  - Ingress/egress controls
  - Process & File System Activity
- ✓ Enforce Global Security Rules
  - Cluster-level policy on Ingress/egress control
- ✓ RBAC Integrated
  - Kubernetes enforcement of CRD creation permissions
  - Support Multi-team use cases
- ✓ Eases migration from staging to production
- ✓ Support Open Policy Agent (OPA), other integrations

```yaml
- apiVersion: neuvector.com/v1
  kind: NvClusterSecurityRule
  metadata:
    name: containers
  spec:
    process:
    - action: deny
      name: sshd
      path: ""
    target:
      Selector:
        criteria:
        - key: container
          op: =
          value: '*'
        name: containers
      policymode: null
    version: v1
```

```yaml
- apiVersion: neuvector.com/v1
  kind: NvClusterSecurityRule
  metadata:
    name: 0racle-0
  spec:
    ingress:
    - Selector:
        criteria:
        - key: service
          op: =
          value: myapp-pod.default
        - key: domain
          op: =
          value: default
        name: nv.myapp-pod.default
      action: allow
      applications:
      - Oracle
      ports: any
    target:
      Selector:
        criteria:
        - key: address
          op: =
          value: oracledb.acme.com
        name: OracleDB
```

Audit NeuVector run-time CRD

Policy

- Only MyApp application can connect to the Oracle DB outside of the cluster
- No containers can run sshd

audit.rego

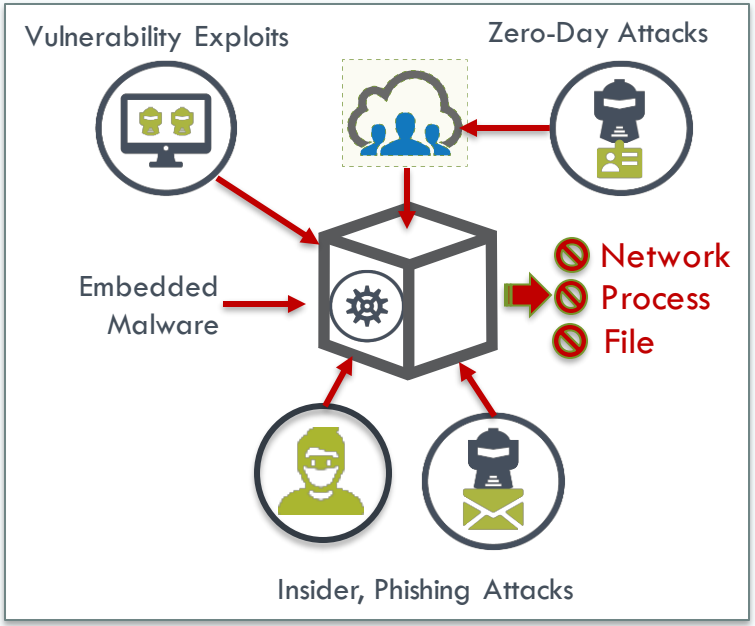
```
package crd

default allow = false

allow = true {
    # target of the network policy
    some r1
    target := data.items[r1].spec.target
    target.selector.criteria[0].key = "address"
    target.selector.criteria[0].op = "="
    target.selector.criteria[0].value = "oracledb.acme.com"

    # ingress L7 application rule to the target
    some j
    ingress := data.items[r1].spec.ingress[j]
    ingress.applications[_] = "Oracle"
    ingress.selector.criteria[0].key = "service"
    ingress.selector.criteria[0].op = "="
    startswith(ingress.selector.criteria[0].value, "myapp-pod")

    # only 'allow' rules are needed, network rules are implicit deny
    ingress.action = "allow"

    # cluster level process rule that prevents ssh daemon
    # from running in the containers
    some r2
    data.items[r2].kind = "NvClusterSecurityRule"

    # target of the process rule
    t := data.items[r2].spec.target
    t.selector.criteria[0].key = "container"
    t.selector.criteria[0].op = "="
    t.selector.criteria[0].value = "*"

    some p
    data.items[r2].spec.process[p].action = "deny"
    data.items[r2].spec.process[p].name = "sshd"
}
```

Run-Time Policy Migration using CRD

Virtual Patching Protects Production Workloads & Hosts

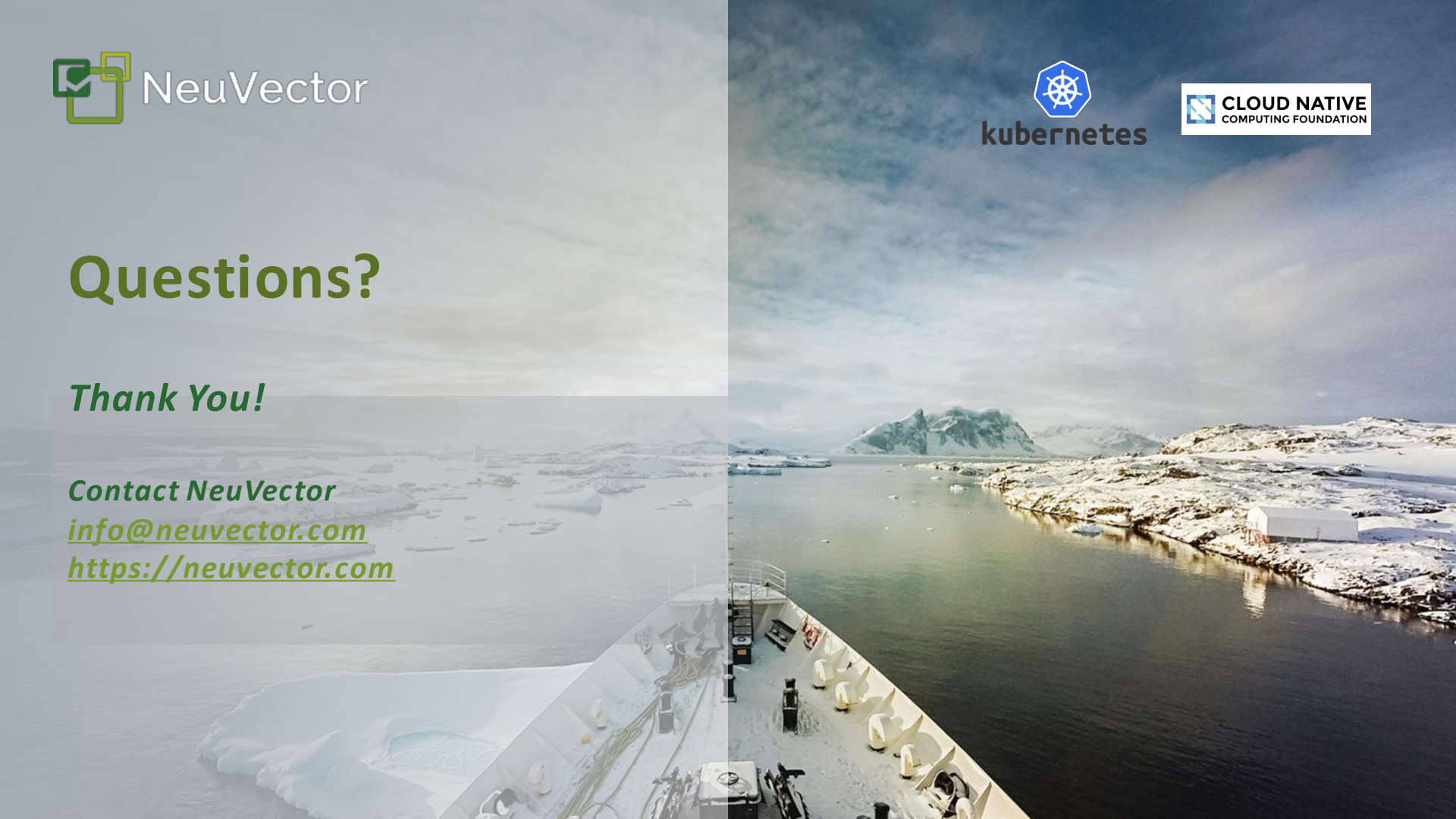
✓ Detects & Prevents
- Vulnerability Exploits
- Zero-day Attacks
- Embedded Malware
- Insider, Phishing Attacks

✓ Learns baselines & Blocks
- Unauthorized Network Connections
- Unauthorized Processes
- Unauthorized File Access



Summary

- Build Security into your pipeline from the beginning
- Automate Vulnerability Scanning
- Use admission control to gate the deployment
- Protect applications at run-time
- Employ Kubernetes-native constructs, such as CRD
- Using OPA to audit and enforce the security policies


NeuVector
kubernetes
CLOUD NATIVE
COMPUTING FOUNDATION
Questions?
Thank You!
Contact NeuVector
info@neuvector.com
https://neuvector.com