

Elevating EKS Security:

Admin and Advanced Network Policies for Multi-Layer Protection

Toshikazu Ichikawa

Senior Partner Solutions Architect
Amazon Web Services Japan G.K.



The security challenge of microservices network

Teams often default to overly restrictive or permissive rules



High complexity

Microservices architectures are becoming increasingly complex



Slow time-to-market

Unclear network patterns leads to security blockers on development velocity



Operational overhead

Network security becomes challenging on multi-tenant environments

Amazon EKS to the rescue!

Enhanced Network Policies support

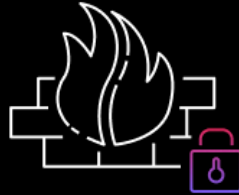


Network Policy support



Single plugin

Natively built-in into
EKS VPC-CNI



Integrated

Seamlessly integrated
into Amazon VPC



Performant

Better performance with
eBPF policy engine

Network Access Control with Network Policies



Declarative

Define pod connectivity policy

Secure

Polices to control intra-cluster traffic

Application centric

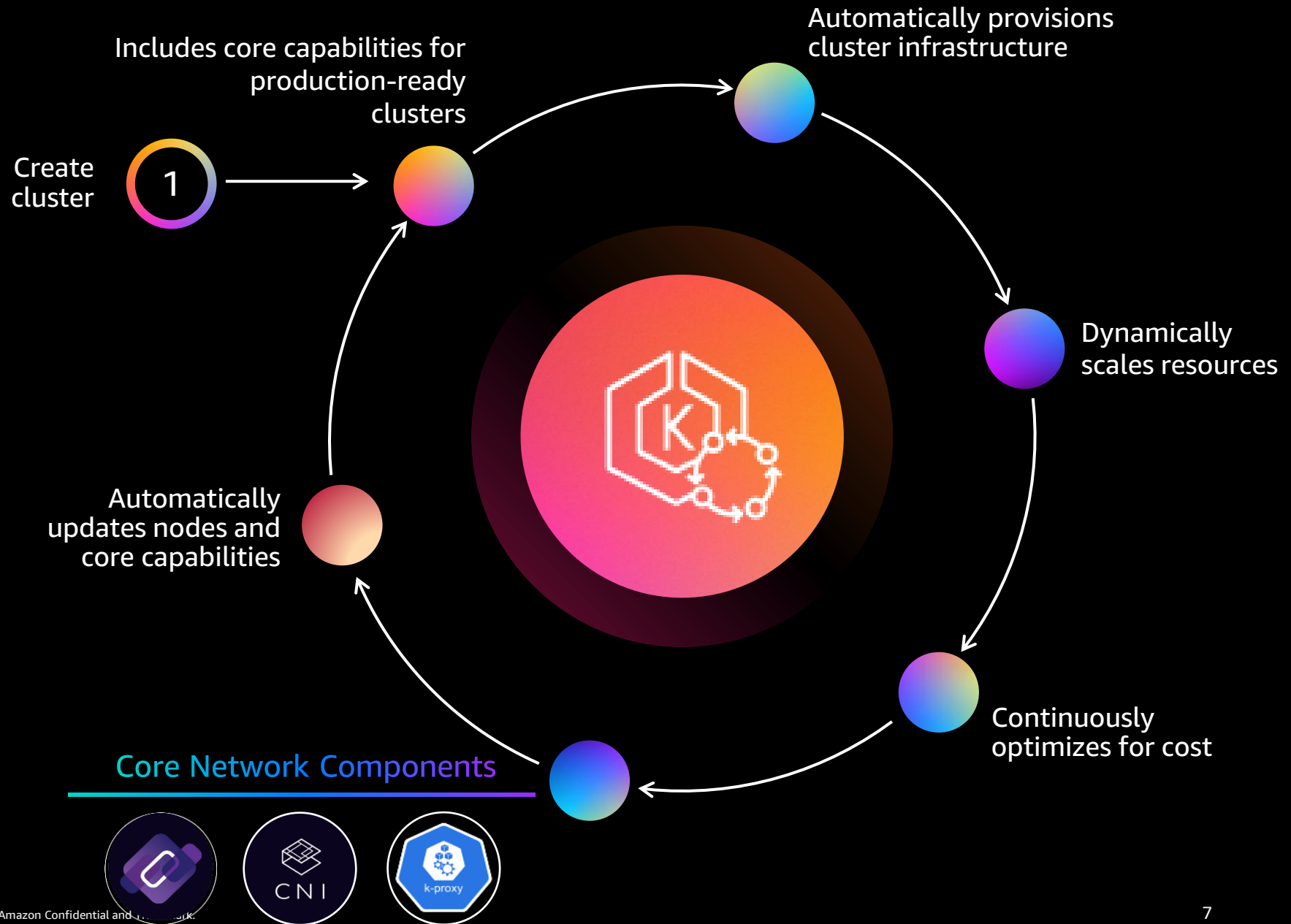
Pod-level isolation

Custom addon setup required

Self-managed VPC-CNI Support

```
aws eks update-addon --cluster-name my-cluster \  
  --addon-name vpc-cni \  
  --addon-version v1.14.0-eksbuild.3 \  
  --service-account-role-arn arn:aws:iam::123456789012:role/AmazonEKSVPCCNIRole \  
  --resolve-conflicts PRESERVE \  
  --configuration-values '{"enableNetworkPolicy": "true"}'
```

How does EKS Auto Mode can help?



Enabling Network Policy on EKS Auto Mode

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: netpol-node-class
spec:
  networkPolicy: DefaultDeny
  networkPolicyEventLogs: Enabled
```

When enabling NetworkPolicy on EKS Auto Mode NodeClass, will be rolled out slowly with the Node lifecycle.

How to control access?

```
metadata:
  name: allow-ingress-namespace
  namespace: orders
  apiVersion: network.k8s.io/v1
```

```
spec:
  policy: allow
  namespace: ingress
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
```

```
spec:
  namespace: ingress
  policyTypes:
    - Ingress
```

```
podSelector:  
  matchLabels:  
    app: checker
```

```
ingress:
- from:
- namespaceSelector:
```

```

    podSelector:
      matchLabels:
        name: all-in-one
        namespace: kube-system
  metadata:
    name: kube-apiserver
    namespace: kube-system

```

```
ingress:
  - from:
      - namespaceSelector:
          matchLabels:
            kubernetes.io/metadata.name: kube-system
```

```

    namespaceSelector:
      matchLabels:
        kubernetes.io/metadata-labels:
          egress:
            - to:

```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
```

```
metadata:
  name: allmyegress-namespaces
  namespace: ui
```

```
spec:
  policyTypes:
  - Egress
collection:
```

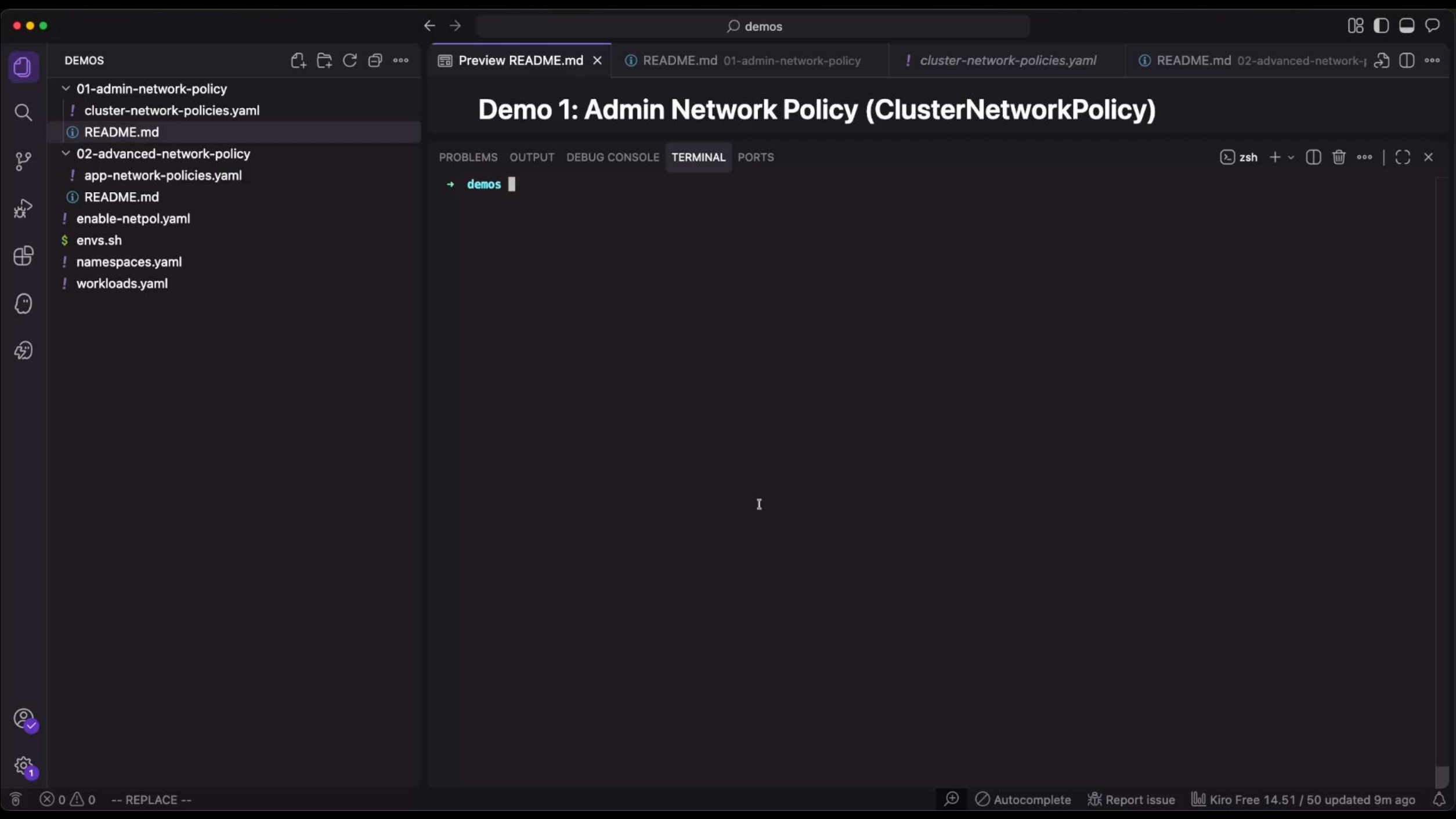
Admin Network Policies

Or ClusterNetworkPolicies

Centralized policy
enforcement for cluster-
wide security standards and
simplified compliance

Available Now!

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ClusterNetworkPolicy
metadata:
  name: deny-all
spec:
  tier: Admin # Admin rules can't be overridden
  priority: 10 # Lower number higher priority
  subject:
    namespaces:
      matchLabels: {} # Match all namespaces
  egress:
    - name: deny-all-egress
      action: Deny
      to:
        - networks:
            - 0.0.0.0/0
  ingress:
    - name: deny-all-internet
      action: Deny
      from:
        - namespaces:
            matchLabels: {} # Match all namespaces
```



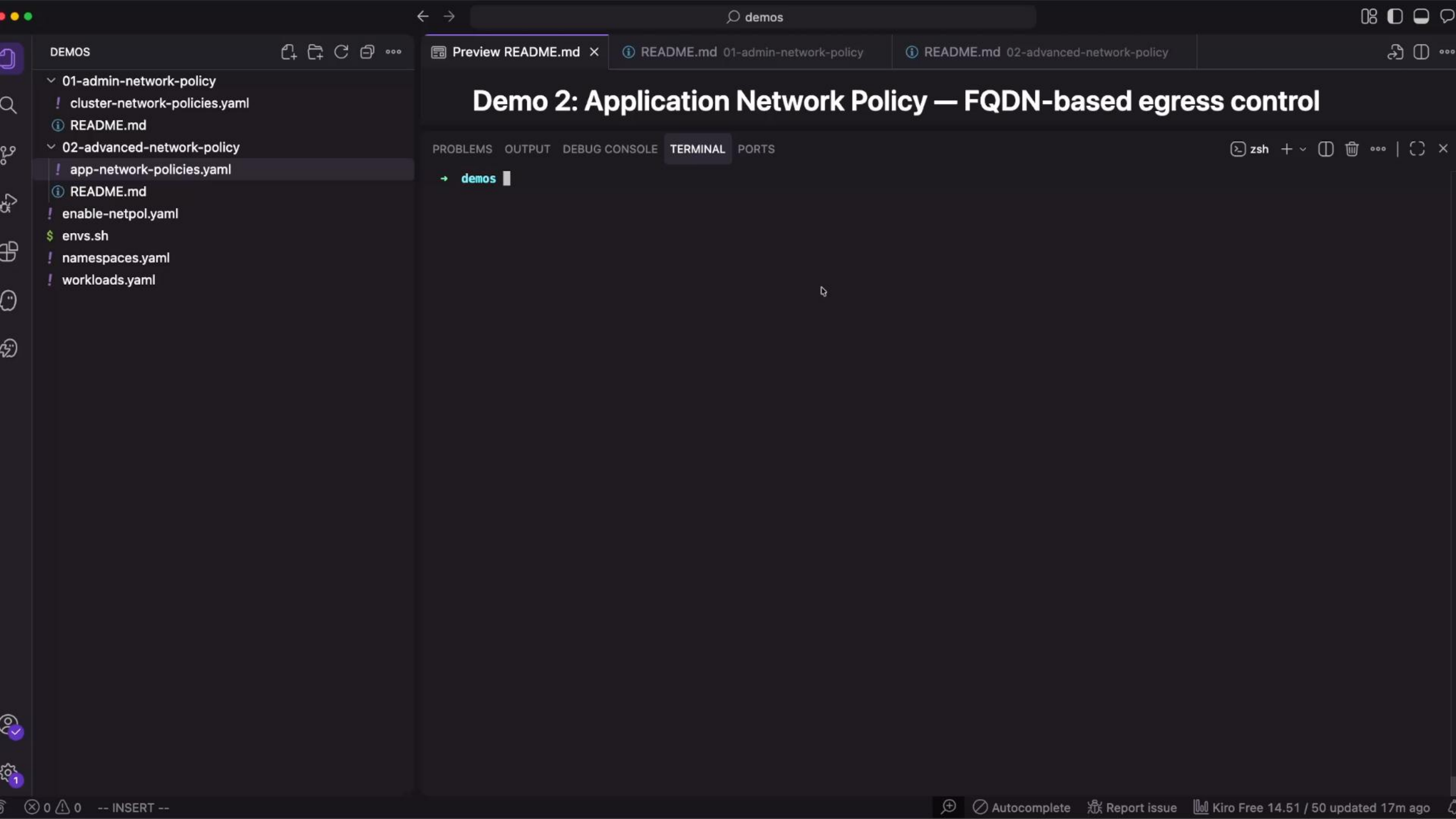
Advanced Network Policies

Or Layer 7/DNS Based Network Policies

Simplified policy management
with fine-grained control based
on DNS-based rules for external
endpoints.

Available Now for EKS Auto Mode!

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ApplicationNetworkPolicy
metadata:
  name: allowlist-egress-domains
  namespace: app
spec:
  podSelector: {}
  policyTypes:
    - Egress
  egress:
    - ports:
        - port: 443
          protocol: TCP
      to:
        - domainNames:
            # Legitimate destinations – only these are
            # reachable
            - "api.example.com"
            # - "s3.us-east-1.amazonaws.com" # Denied S3 region
            - "s3.us-west-2.amazonaws.com" # Allowed S3 region
            - "sts.amazonaws.com" # Global STS endpoint
```



DEMOS

01-admin-network-policy

cluster-network-policies.yaml

README.md

02-advanced-network-policy

app-network-policies.yaml

README.md

enable-netpol.yaml

envs.sh

namespaces.yaml

workloads.yaml

Demo 2: Application Network Policy — FQDN-based egress control

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

→ demos

zsh + - - - | x

0 0 -- INSERT --

Autocomplete Report issue Kiro Free 14.51 / 50 updated 17m ago

Key takeaways



Faster time to market
with reduced security
incidents



Improved security with
simplified management



Lower operational
overhead leading to
faster deployment

セッションをご視聴いただき有難うございました。
お近くのブーススタッフまで QR コードのご提示をお願いいたします。
Thank you for attending this session. Please show your QR code to the
nearest booth staff.

Thank you!

Toshikazu Ichikawa
kazuich@amazon.co.jp

