Kubernetes Authentication

```
# Configure the kubectl CLI (sec540.com/1575)

# Configure access to an AWS cluster
$ aws eks update-kubeconfig --region us-west-2
  --name aviata

# Configure access to an Azure cluster using
# the Entra ID auth plugin
$ az aks get-credentials -g ace135 -n aviata
$ kubelogin convert-kubeconfig -l azurecli

# Configure access to a Google cluster
$ gcloud container clusters get-credentials
  --region us-west2 --project ace135 aviata

# View Kubernetes authentication data
# including the user and group membership
$ kubectl auth whoami

# Check RBAC permissions to list the Kubernetes
# pods in the aviata namespace
$ kubectl auth can-i get pods -n aviata
```

Managing Kubernetes Resources

```
# List cluster resources
$ kubectl get nodes

# Describe an API resource and its fields
$ kubectl explain pods --recursive

# Creating or updating a resource
$ cat >> namespace.yml << EOF
apiVersion: v1
kind: Namespace
metadata:
  name: aviata
  annotations:
    ace135/owner: "aviata"
EOF
$ kubectl apply -f ./namespace.yml

# Viewing a resource in a namespace
$ kubectl describe pod -n aviata api

# Deleting a resource in a namespace
$ kubectl delete pod -n aviata api
```

Kubernetes RBAC

```
# ClusterRole resources define a list of cluster-wide
# permissions and ClusterRoleBinding resources grant
# those permissions to a user or group. The following
# ClusterRole grants read permissions to the cluster
# namespaces and customresourcedefinitions resources.
$ cat >> cluster_rbac.yml << EOF
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-security-auditor
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get", "list"]
  - apiGroups: ["apiextensions.k8s.io"]
    resources: ["customresourcedefinitions"]
    verbs: ["get", "list"]
---
kind: ClusterRoleBinding
metadata:
  name: cluster-security-auditor
subjects:
  - kind: Group
    name: aviata-security-auditor
    apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-security-auditor
  apiGroup: rbac.authorization.k8s.io
EOF
$ kubectl apply -f ./cluster_rbac.yml

# With the cluster role binding, members of the
# aviata-security-auditor group can view these
# cluster resources
$ kubectl get namespaces
$ kubectl get customresourcedefinitions

# Role resources define a list of namespaced
# permissions and RoleBinding resources grant
# those permissions to a user or group
$ cat >> rbac.yml << EOF
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
```

SANS CLOUD SECURITY

Kubernetes Cheat Sheet

Cloud Native Security and DevSecOps Automation

By Eric Johnson

Cheat Sheet v1.0.0

sans.org/cloud-security

Kubernetes RBAC (continued)

```
  name: aviata-security-auditor
  namespace: aviata
rules:
  - apiGroups: [""]
    resources: ["pods", "pods/log", "services", …]
    verbs: ["get", "list", "watch"]
---
kind: RoleBinding
metadata:
  name: aviata-security-auditor
  namespace: aviata
subjects:
  - kind: Group
    name: aviata-security-auditor
    apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: aviata-security-auditor
  apiGroup: rbac.authorization.k8s.io
EOF
$ kubectl apply -f ./rbac.yml

# With the role binding, members of the aviata-
# security-auditor group can view services, pods,
# and logs in the aviata namespace
$ kubectl get services -n aviata
$ kubectl logs -n aviata api

# Add the watch flag to see real time object updates
$ kubectl get pods -n aviata --watch
```

Kubernetes Services & Deployments

```
# List all resources in a namespace
$ kubectl api-resources --verbs=list
  --namespaced -o name | xargs -n 1 kubectl get
  --ignore-not-found --show-kind -n aviata

# List the ingress objects routing external
# traffic to a Kubernetes service
$ kubectl get ingress -n aviata

# List the service objects load balancing
# traffic to a group of pods
$ kubectl get services -n aviata

# List the deployment objects responsible for
# managing groups of pods (replicas)
$ kubectl get deployments -n aviata

# Scale a deployment's replicas (# of pods)
$ kubectl scale -n aviata --replicas 3
  deployment/api

# List all pods in the cluster with more detail
$ kubectl get pods -A -o wide

# List all pods in a namespace including labels
$ kubectl get pods -n aviata --show-labels

# List all pods with a given label
$ kubectl get pods -n aviata -l app=api

# Shell access to a pod running in the aviata
# namespace with the app=api label
$ kubectl exec --stdin --tty -n aviata
  $(kubectl get pods -n aviata -l app=api -o
  json | jq -r .items[0].metadata.name)
  -- /bin/bash

# Copy a file from a pod to a host directory
$ kubectl cp aviata/api:/www/app.jar
  /tmp/app.jar

# View the logs for a deployment
$ kubectl logs -n aviata deployments/api

# View the logs for pods with a given label
$ kubectl logs -n aviata -l app=api

# View the logs for a pod's previous
# instantiation of a container
$ kubectl logs -n aviata -l app=api --previous
```

OPA Gatekeeper Library

```
# The OPA Gatekeeper Library (sec540.com/2188) validates
# resources against constraint policies. Use Kustomize
# (sec540.com/1597) to install the cluster's gatekeeper
# library constraint templates
$ kubectl apply -k https://github.com/open-policy-
  agent/gatekeeper-library//library/general/?ref=master
$ kubectl get constrainttemplates

# List the gatekeeper library constraints applied to
# resources and any associated violations
$ kubectl get constraints

# Search the gatekeeper admission controller manager logs
# for denied resources
$ kubectl logs -n gatekeeper-system deployment/
  gatekeeper-controller-manager | grep "denied admission"
```

Calico Network Policy

```
# Tigera's Calico (sec540.com/1929) validates pod traffic
# against network policies.

# List the global network policies that apply cluster-
# wide rules to all namespaces
$ kubectl get globalnetworkpolicies.crd.projectcalico.org

# List the network policies in the aviata namespace
$ kubectl get networkpolicies.crd.projectcalico.org
  -n aviata

# Create a Calico NetworkPolicy blocking IMDS access
$ cat >> deny_imds.yml << EOF
apiVersion: crd.projectcalico.org/v1
kind: NetworkPolicy
metadata:
 name: deny-imds
 namespace: aviata
spec:
 types:
   - Egress
 egress:
   - action: Deny
     destination:
       nets: ["169.254.169.254/32", "fd00:ec2::254"]
   - action: Allow
     destination:
       nets: ["0.0.0.0/0", "::/0"]
EOF
$ kubectl apply -f ./deny_imds.yml
```

Sigstore Cosign Controller

```
# The Sigstore (sec540.com/1484) Cosign Policy
# (sec540.com/1768) controller verifies image
# supply chain metadata before creating a
# container

# List all namespaces with the sigstore
# validation label
$ kubectl get ns
  -l policy.sigstore.dev/include=true

# Add the sigstore label to a namespace to
# ensure all image signatures are verified
$ kubectl label ns aviata
  policy.sigstore.dev/include=true
```

OpenTelemetry Kubernetes Stack

```
# Use Helm (sec540.com/1600) to install the
# OpenTelemetry Kube Stack (sec540.com/2189)
# chart. The chart installs the OTel operator and
# collectors for cluster metrics, logs, and traces

# Add the helm repository and packages
$ helm repo add open-telemetry https://open-
  telemetry.github.io/opentelemetry-helm-charts
$ helm repo update

# Install the OTel Kube Stack helm chart
$ helm upgrade --install otel-kube-stack
  open-telemetry/opentelemetry-kube-stack
  --create-namespace --namespace otel-system
  --values ./otel-config.yaml
```