Scenario Based Interview Questions on EKS

You're asked to harden security on your EKS cluster. What are the best practices you would apply?

To harden security on an **Amazon EKS cluster**, I would apply best practices in **5 key areas**: **Access Control**, **Networking**, **Pod Security**, **Secrets Management**, and **Monitoring**.

---

1. Access Control

✅ **Use IAM + RBAC**

- Use **IAM Roles for Service Accounts (IRSA)** to grant pods fine-grained AWS access.
- Map IAM identities to Kubernetes RBAC roles using aws-auth ConfigMap.
- Use **least privilege** principle:
  - Give users and apps only the permissions they need.

✅ **Limit kubectl Access**

- Disable public access to EKS API server or restrict by IP CIDR.
- Create **read-only roles** for developers if full control isn't needed.

🌐 **2. Network Security**

✅ **Private Cluster**

- Disable public endpoint or restrict access via **VPC** and security groups.
- Use **VPC endpoints** to access AWS services privately.

✅ **Use Network Policies**

- Use **Calico** or **Cilium** to enforce **Kubernetes NetworkPolicies**:
  - Restrict pod-to-pod traffic to only what's necessary.

✅ **Security Groups for Pods (SGP)**

- Assign **EC2-style security groups to individual pods** if using EC2 nodes.

📦 **3. Pod & Workload Security**

✅ **Enable Pod Security Standards (PSS)**

- Enforce **restricted policies** via PodSecurity admission controller.
- Prevent privilege escalation, root containers, hostPath mounts.

✅ **Use Read-Only Filesystems**

- Run containers with readOnlyRootFilesystem: true where possible.

✅ **Non-Root Users**

- Run containers as non-root by setting:

yaml

```
securityContext:
 runAsUser: 1000
 runAsNonRoot: true
```

🔑 **4. Secrets Management**

✅ **Use AWS Secrets Manager or Parameter Store**

- Store sensitive data like DB passwords and API keys in encrypted stores.
- Mount secrets into pods using **IRSA + external-secrets operator**.

✅ **Avoid Plaintext Secrets in YAML**

- Never store credentials in ConfigMaps or environment variables in code repos.

---

5. Observability & Audit

✅ **Enable Logging**

- Send **EKS control plane logs** to **CloudWatch Logs**.
  - Audit, Authenticator, Scheduler, etc.

✅ **Enable Audit Logging**

- Monitor changes to roles, pods, services, and network policies.

✅ **Use Tools Like:**

- **Falco** for runtime security monitoring
- **Kube-bench** to scan against CIS benchmarks
- **Kube-hunter** to simulate attacks and find vulnerabilities

---

Additional Hardening

Area	Practice
Image Security	Use **scanned, signed images** from trusted registries
Auto Scaling	Prevent overprovisioning and DoS using **limits/requests**
CI/CD	Integrate security scanning tools (e.g. Trivy) in pipeline
Node Groups	Use **managed node groups** and enable **auto-upgrade**

2. Traffic to your app increases 10x during peak hours. How will you scale your EKS workloads to handle it?

To handle 10x increase in traffic, I would set up **automatic horizontal and vertical scaling** at both the **pod level** and the **cluster/node level** in EKS.

🔁 1. Enable Horizontal Pod Autoscaler (HPA)

- Scales pods based on CPU, memory, or custom metrics.
- Example: if CPU > 70%, increase replicas.

YAML snippet:

yaml

```
---------
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
 name: backend-hpa
spec:
 scaleTargetRef:
  apiVersion: apps/v1
  kind: Deployment
  name: backend
 minReplicas: 2
 maxReplicas: 20
 metrics:
  - type: Resource
   resource:
    name: cpu
    target:
     type: Utilization
     averageUtilization: 70
```

⚙️ 2. Enable Cluster Autoscaler

- Automatically adds or removes EC2 worker nodes based on pending pods.
- Works with **managed node groups** and **auto scaling groups**.
- Deploy using the Cluster Autoscaler Helm chart.

bash

```
---------
helm install cluster-autoscaler ...
```

- Ensure the ASG allows scaling up to handle 10x peak.

📦 3. Use Resource Requests & Limits

- Always define requests and limits to help HPA and Cluster Autoscaler make correct decisions.

yaml

```
---------
resources:
 requests:
  cpu: "200m"
  memory: "256Mi"
 limits:
```

cpu: "500m"
memory: "512Mi"

🌐 **4. Use Load Balancer with Auto Scaling**

- Deploy an **AWS ALB/NLB** using the **AWS Load Balancer Controller**.
- It will balance traffic evenly across pods in all AZs.

📊 **5. Use Prometheus + Metrics Server**

- Install **Metrics Server** to provide resource usage data for HPA.
- Use **Prometheus + Grafana** to monitor app scaling trends and adjust thresholds.

---

6. Optional: Use Karpenter for Node Autoscaling

- Karpenter is a modern autoscaler that provisions nodes faster and more cost-efficiently than Cluster Autoscaler.
- It can scale based on real-time pod specs and launch spot or on-demand nodes as needed.

3 Your application is intermittently failing in EKS, but you don't have logs or metrics configured. What would you do?

If the app is intermittently failing and no logging or monitoring is set up, I would take the following **immediate steps to troubleshoot** and then **set up observability for long-term fixes**.

🛠️ **1. Check Pod and Event Status**

Start by gathering basic info from Kubernetes:

bash

```
kubectl get pods -n <namespace>
kubectl describe pod <pod-name> -n <namespace>
kubectl get events -n <namespace> --sort-by=.metadata.creationTimestamp
```

- Look for **CrashLoopBackOff**, OOMKilled, failed mounts, etc.
- Events can show **restarts**, **probe failures**, or **scheduling issues**.

📜 **2. Check Container Logs (Even if Limited)**

Even without centralized logging, you can still access recent logs:

bash

```
kubectl logs <pod-name> -n <namespace> --previous
```

- Look for error messages, exceptions, or timeouts.
- Check logs from both containers (if sidecars exist) and for multiple replicas.

📦 **3. Deploy a Debug Pod**

If the app is networking-dependent or failing silently:

bash

```
kubectl run -it debug --image=busybox --restart=Never -- bash
```

- Use curl, telnet, or dig to test DNS, network, and external services.

🔁 **4. Increase Replicas Temporarily**
If failures are intermittent, scale up temporarily:
bash
---
kubectl scale deployment <name> --replicas=10
- Helps isolate whether it's a **node**, **pod**, or **traffic-level issue**.

🔍 **5. Manually Check Nodes**
Check if it's a **node issue**:
bash
---
kubectl get nodes
kubectl describe node <node-name>
- Spot issues like disk pressure, memory pressure, or network errors.

🛠️ **6. Set Up Minimal Observability Immediately**
✅ **a. Install Metrics Server**
Allows kubectl top to show CPU/memory usage.
bash
---
kubectl top pod
kubectl top node
✅ **b. Enable Container Logs to CloudWatch**
- Use **CloudWatch Container Insights** or install **FluentBit** DaemonSet to stream logs.
- Helps track logs cluster-wide in future.

✅ **c. Add Readiness/Liveness Probes**
Ensure failures get captured properly and don't silently pass.
yaml
---
livenessProbe:
  httpGet:
    path: /healthz
    port: 8080

Final Summary:
"In the short term, I'd use kubectl describe, logs, and event inspection to isolate the issue. I'd then install metrics server and basic logging (like FluentBit or Container Insights) so future issues are visible and traceable."

Which is the most secure way to give pods access to AWS services?
A. Store encrypted keys in ConfigMap
B. Use IAM Roles for Service Accounts
C. Give node IAM least privileges
D. Use environment variables to store encrypted password
✅ **Answer: B**

To restrict pod-to-pod communication, what should you implement?
A. Security Groups
B. Network Policies
C. NAT Gateway
D. VPC Peering
✅ **Answer: B**

Where should EKS API server access be restricted?
A. By IP CIDR or private endpoint
B. By adding liveness probe
C. By reducing node size
D. By deleting default namespace
✅ **Answer: A**

Which AWS service is recommended for storing sensitive credentials for EKS workloads?
A. S3
B. Secrets Manager
C. ConfigMap
D. DynamoDB
✅ **Answer: B**

Which feature scales the number of worker nodes automatically?
A. Horizontal Pod Autoscaler
B. Vertical Pod Autoscaler
C. Cluster Autoscaler
D. Metrics Server
✅ **Answer: C**

Which modern autoscaling tool for EKS launches nodes faster than Cluster Autoscaler?

A. Karpenter
B. Prometheus
C. Helm
D. Kubectl
✅ **Answer: A**

What's the first Kubernetes command to check pod restart causes?
A. kubectl top pod
B. kubectl describe pod
C. kubectl rollout restart
D. kubectl exec
✅ **Answer: B**

If you suspect DNS issues inside a pod, which test should you run?
A. nslookup
B. netstat
C. uptime
D. traceroute
✅ **Answer: A**

Which lightweight tool can immediately collect cluster CPU/memory usage?
A. FluentBit
B. Metrics Server
C. Prometheus
D. Karpenter
✅ **Answer: B**
