EKS-02

13 June 2025 15:35

🧠 Advanced Commentary: EKS Cluster Creation (Control Plane Only)

🧩 1. Control Plane (Managed by AWS)

When you run:

eksctl create cluster --without-nodegroup

You are provisioning only the **Kubernetes control plane** — i.e., the "brain" of your cluster — which includes:

Component	Description
kube-apiserver	Front-door for all cluster operations (authN/Z, kubelet, etc.)
etcd	State store for all objects: pods, deployments, configmaps
kube-scheduler	Assigns pods to available nodes (but nodes don't exist yet)
kube-controller-manager	Watches and reconciles resource states
cloud-controller-manager	Interacts with AWS for ELB, EBS, etc.

🔘 **Hosted by AWS**, this control plane is:

- HA by default (multi-AZ)
- Secure (managed TLS)
- API endpoint is internet-accessible unless explicitly locked down

⌨️ 2. API-Only Mode – What It Means

- You **can interact with the cluster** using kubectl, but:
- **No pods will be scheduled**, because:
  - No worker nodes exist
  - Addons like coredns, kube-proxy, metrics-server remain Pending

💡 **Useful for:**

- Attaching custom worker groups later
- Using Fargate or Spot capacity pools
- Enabling pod-level IAM or security controls first

🌐 3. Kubernetes Networking Inside EKS

a. Network Namespace Inside Pods

Every Kubernetes **Pod** runs in its own **network namespace**, isolated from other pods and host processes.

Inside this namespace:

- The Pod sees a single **eth0** interface (usually backed by an ENI in EKS).
- Each container shares this pod-level network namespace.
- DNS, routing, and loopback are pod-specific.

b. VPC-CNI Plugin (Installed Addon)

- Amazon's vpc-cni attaches **secondary IPs from VPC subnets** to EC2 ENIs.
- Each Pod is assigned a **real VPC IP address**.
- No overlay (like Calico, Flannel); everything is **routable natively**.

☑ **Advantages:**

- Native CloudWatch VPC Flow Logs
- NACLs and Security Groups can control Pod traffic
- No NAT required for east-west Pod communication

🔐 4. OIDC & IRSA (IAM Roles for Service Accounts)

a. Service Accounts

- Kubernetes concept: identities used by pods to authenticate within the cluster.
- Bound to a Pod via YAML (spec.serviceAccountName).

b. IRSA (IAM Roles for Service Accounts)

- Allows **binding an AWS IAM role to a Kubernetes service account**.
- EKS OIDC provider issues a token that can be assumed by IAM.
  This replaces assigning overly broad IAM roles to EC2 nodes.

☑ Benefits:

- Enables **least-privilege pod identity** (e.g., only S3 read for one pod).
- Follows **zero-trust** principle in multi-tenant clusters.
- Integrates cleanly with AWS CloudTrail.

c. You Saw This Warning:

OIDC is disabled; eksctl cannot configure the requested permissions
You must enable OIDC:

```
eksctl utils associate-iam-oidc-provider \
  --region us-east-1 \
  --cluster ep33-eks-01 \
  --approve
```

Then IRSA is possible:

```
eksctl create iamserviceaccount \
  --cluster ep33-eks-01 \
  --namespace kube-system \
  --name my-serviceaccount \
  --attach-policy-arn arn:aws:iam::ACCOUNT_ID:policy/MyScopedPolicy \
  --approve
```

📊 5. Metrics Server Addon

- This is the cluster component that **exposes CPU/memory usage metrics** for:
  - Horizontal Pod Autoscaler (HPA)
  - Vertical Pod Autoscaler (VPA)
- It scrapes metrics from kubelets on worker nodes.

🛑 Without nodes, **Metrics Server is stuck in Pending state**.

☑ Once nodes join:

- Metrics Server gets scheduled
- HPA can begin scaling decisions based on live metrics

☑ Expert Takeaways

Concept	Role in EKS
Control Plane	Managed by AWS, API-only until nodes are added
Network Namespaces	Isolated networking per Pod, shared across containers

vpc-cni	Assigns real VPC IPs to Pods, supports native routing & logging
Service Accounts	Kubernetes identity for a Pod
OIDC + IRSA	Grants IAM access to specific Pods via trusted service accounts
Least Privilege IAM	Encouraged via IRSA; avoid node-level full-access roles
Metrics Server	Enables auto-scaling logic but only works when nodes are present

==========

🧠 EKS Cluster Description (ep33-eks-01)

◇ 1. Cluster Identity & Basics

Field	Value/Implication
name	ep33-eks-01 – your logical cluster identifier
arn	Unique cluster ARN for IAM, CloudFormation, CloudTrail integration
createdAt	"2025-06-13T09:30:09.983Z" – timestamp of creation
version	1.32 – Kubernetes version (matches EKS-supported control plane + API)
platformVersion	eks.12 – Internal EKS platform version (affects patching, add-ons)
status	ACTIVE – Indicates cluster is fully initialized
endpoint	Public HTTPS endpoint used by kubectl / API clients
accessConfig.authenticationMode	API_AND_CONFIG_MAP – Traditional access via IAM + ConfigMap auth

🔐 **Note:** You must update the aws-auth ConfigMap to allow IAM roles/users to interact with the cluster.

◇ 2. IAM Role for Control Plane

Field	Value
roleArn	eksctl-ep33-eks-01-cluster-ServiceRole-...
Purpose	IAM role that EKS uses to manage networking, ELBs, cloud integrations

☑ This role must have permissions like ec2:DescribeSubnets, elasticloadbalancing:*, autoscaling:*, etc.

◇ 3. VPC & Subnet Configuration

Field	Details
vpcId	vpc-0fd0b051d644ac6b6 – Dedicated VPC for this EKS cluster
subnetIds	4 Subnets (likely 2 public + 2 private across AZs)
securityGroupIds	Security groups for worker nodes
clusterSecurityGroupId	Managed by EKS; used for inter-node pod communication
endpointPublicAccess	true – API endpoint is accessible over the internet
endpointPrivateAccess	false – No private access unless enabled
publicAccessCidrs	0.0.0.0/0 – API is open to all IPs (⚠ not secure in prod)

☑ For production, restrict publicAccessCidrs to your office IP or enable privateAccess.

◇ 4. Kubernetes Networking

Field	Description
serviceIpv4Cidr	10.100.0.0/16 – Cluster-wide service CIDR (used by kube-proxy/Service IPs)
ipFamily	ipv4 – Default (EKS supports dual-stack IPv6 in preview)
elasticLoadBalancing	"enabled": false – No managed LBs via EKS add-on (you can still create manually)

☑ Pod IPs are handled via the **Amazon VPC CNI plugin**, which assigns secondary ENI IPs from subnet CIDRs.

◇ 5. OIDC Configuration (for IRSA)

Field	Details
oidc.issuer	https://oidc.eks.us-east-1.amazonaws.com/id/5789FF2FD...
Enabled?	✗ Not yet enabled (alpha.eksctl.io/cluster-oidc-enabled: "false")

🔐 Without OIDC enabled:

- **IRSA (IAM Roles for Service Accounts)** won't work.
- You'll have to assign IAM permissions at the **EC2 node level**, which is less secure.

Enable OIDC with:
eksctl utils associate-iam-oidc-provider --region us-east-1 --cluster ep33-eks-01 --approve

◇ 6. Logging

Field	Value
Logging Types	api, audit, authenticator, controllerManager, scheduler
enabled	false

🔍 These logs are **not being sent to CloudWatch Logs** yet.

Enable with:
eksctl utils update-cluster-logging --enable-types all --region us-east-1 --cluster ep33-eks-01

◇ 7. Certificate Authority

Used for:

- TLS communication from kubectl and SDK clients
- Populates the CA section of your ~/.kube/config file

☑ You don't need to manage this manually — it's consumed automatically by Kubernetes clients.

◇ 8. Cluster Tags (via CloudFormation)

Tag Key	Purpose
aws:cloudformation:*	Tracks CloudFormation stack that built the cluster
alpha.eksctl.io/cluster-name	Used internally by eksctl
alpha.eksctl.io/cluster-oidc-enabled	"false" – No OIDC yet
eksctl.cluster.k8s.io/...	eksctl versioning and discovery
Name	Named logical resource in the CFN stack

🛠️ These are useful for automation, cost allocation, and tagging compliance.

☑ Summary

Area	Highlights
Control Plane	Managed by AWS; v1.32; active and public endpoint
IAM	Cluster role provided; IRSA not yet enabled
Network	Pod IPs from subnet ENIs (via vpc-cni); Services use 10.100.0.0/16
Security	Public access = 0.0.0.0/0 (⚠ wide open)
Observability	Logging disabled; metrics-server needs to be installed later
Certificates	TLS CA generated and stored in ~/.kube/config