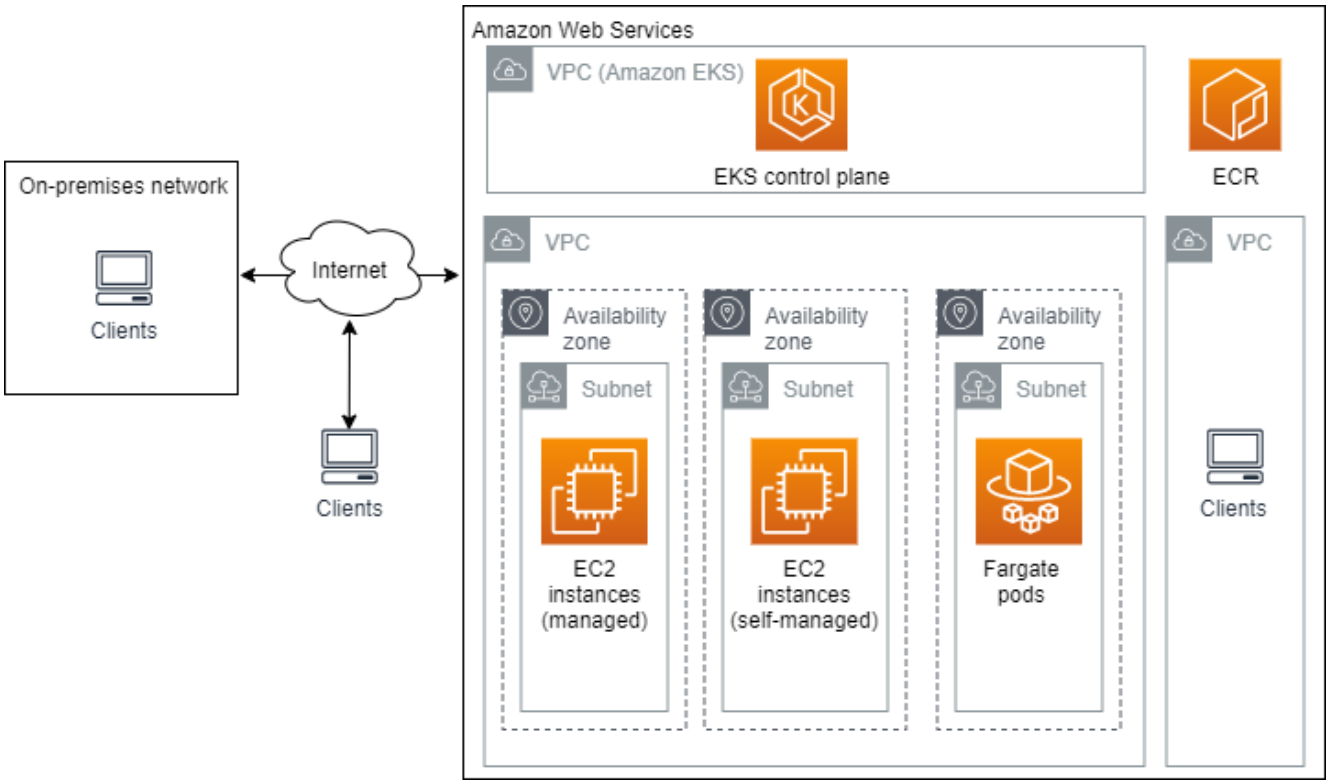
Amazon EKS networking

PDF (eks-ug.pdf#eks-networking) | RSS (doc-history.rss)

This chapter provides an overview of Amazon EKS networking. The following diagram shows key components of an Amazon EKS cluster, and the relationship of these components to a VPC.



The following explanations help you understand how components of the diagram relate to each other and which topics in this guide and other AWS guides that you can reference for more information.

- **Amazon VPC and subnets** – All Amazon EKS resources are deployed to one AWS Region in an existing subnet in an existing VPC. For more information, see VPCs and subnets (https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html) in the Amazon VPC User Guide. Each subnet exists in one Availability Zone. The VPC and subnets must meet requirements such as the following:
  - VPCs and subnets must be tagged appropriately, so that Kubernetes knows that it can use them for deploying resources, such as load balancers. For more information, see Subnet tagging (./network_reqs.html#vpc-subnet-tagging) . If you deploy the VPC using an

Amazon EKS provided AWS CloudFormation template (./creating-a-vpc.html#create-vpc) or using `eksctl`, then the VPC and subnets are tagged appropriately for you.

- A subnet may or may not have internet access. If a subnet does not have internet access, the pods deployed within it must be able to access other AWS services, such as Amazon ECR, to pull container images. For more information about using subnets that don't have internet access, see Private clusters (./private-clusters.html).
- Any public subnets that you use must be configured to auto-assign public IPv4 addresses or IPv6 addresses for Amazon EC2 instances launched within them. For more information, see VPC IP addressing (./network_reqs.html#vpc-cidr).
- If using IPv6, each subnet must be configured to auto-assign IPv6 addresses. For more information, see Modify the IPv6 addressing attribute for your subnet (https://docs.aws.amazon.com/vpc/latest/userguide/vpc-ip-addressing.html#vpc-working-with-ip-addresses) in the Amazon VPC User Guide.
- The nodes and control plane must be able to communicate over all ports through appropriately tagged security groups (https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html). For more information, see Amazon EKS security group considerations (./sec-group-reqs.html).
- You can implement a network segmentation and tenant isolation network policy. Network policies are similar to AWS security groups in that you can create network ingress and egress rules. Instead of assigning instances to a security group, you assign network policies to pods using pod selectors and labels. For more information, see Installing the Calico add-on (./calico.html).

You can deploy a VPC and subnets that meet the Amazon EKS requirements through manual configuration, or by deploying the VPC and subnets using eksctl (./eksctl.html), or an Amazon EKS provided AWS CloudFormation template. Both `eksctl` and the AWS CloudFormation template create the VPC and subnets with the required configuration. For more information, see Creating a VPC for your Amazon EKS cluster (./creating-a-vpc.html#create-vpc).

- **Amazon EKS control plane** – Deployed and managed by Amazon EKS in an Amazon EKS managed VPC. When you create the cluster, Amazon EKS creates and manages network interfaces in your account that have `Amazon EKS <cluster name>` in their description. These network interfaces allow AWS Fargate and Amazon EC2 instances to communicate with the control plane.

  By default, the control plane exposes a public endpoint so that clients and nodes can communicate with the cluster. You can limit the internet client source IP addresses that can communicate with the public endpoint. Alternatively, you can enable a private endpoint and disable the public endpoint or enable both the public and private endpoints. To learn more about cluster endpoints, see Amazon EKS cluster endpoint access control (./cluster-endpoint.html).

  Clients in your on-premises network or other VPCs can communicate with the public or private-only endpoint, if you've configured connectivity between the VPC that the cluster

is deployed to and the other networks. For more information about connecting your VPC to other networks, see the AWS Network-to-Amazon VPC connectivity options (https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/network-to-amazon-vpc-connectivity-options.html) and Amazon VPC-to-Amazon VPC connectivity options (https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/amazon-vpc-to-amazon-vpc-connectivity-options.html) technical papers.

- **Amazon EC2 instances** – Each Amazon EC2 node is deployed to one subnet. Each node is assigned a private IP address (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html#concepts-private-addresses) from a CIDR block assigned to the subnet. If the subnets were created using one of the Amazon EKS provided AWS CloudFormation templates (./creating-a-vpc.html#create-vpc) , then nodes deployed to public subnets are automatically assigned a public IPv4 address (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html#concepts-public-addresses) by the subnet. Each node is deployed with the Amazon VPC CNI add-on by default. The add-on assigns each pod a private IP address from the CIDR block assigned to the subnet that the node is in and adds an IPv4 address as a secondary IP address to one of the network interfaces (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html) attached to the instance. This AWS resource is referred to as a *network interface* in the AWS Management Console and the Amazon EC2 API. Therefore, we use "network interface" in this documentation instead of "elastic network interface". The term "network interface" in this documentation always means "elastic network interface".

  You can change this behavior by assigning additional IPv4 CIDR blocks to your VPC and enabling CNI custom networking (./cni-custom-network.html) , which assigns IP addresses to pods from different subnets than the node is deployed to. To use custom networking, you must enable it when you launch your nodes. You can also associate unique security groups with some of the pods running on many Amazon EC2 instance types. For more information, see Security groups for pods (./security-groups-for-pods.html) .

  By default, the source IPv4 address of each pod that communicates with resources outside of the VPC is translated through network address translation (NAT) to the primary IP address of the primary network interface attached to the node. You can change this behavior to instead have a NAT device in a private subnet translate each pod's IPv4 address to the NAT device's IPv4 address. For more information, see External source network address translation (SNAT) (./external-snat.html) .

  If your instance is deployed to a cluster that uses the IPv6 family, you must assign an IPv6 CIDR block to your VPC and subnets. Outbound IPv6 traffic is not network address translated. For more information about using IPv6 with your cluster, see Assigning IPv6 addresses to pods and services (./cni-ipv6.html) .

- **Fargate pods** – Deployed to private subnets only. Each pod is assigned a private IPv4 (and optionally, an IPv6) address from the CIDR block assigned to the subnet. Fargate does not support all pod networking options. For more information, see AWS Fargate considerations (./fargate.html#fargate-considerations) .