IAM roles for service accounts

Archived: 2026-04-06 00:41:13 UTC

Help improve this page

To contribute to this user guide, choose the **Edit this page on GitHub** link that is located in the right pane of every page.

Help improve this page

To contribute to this user guide, choose the **Edit this page on GitHub** link that is located in the right pane of every page.

Tip

Register for upcoming Amazon EKS workshops.

Applications in a Pod's containers can use an AWS SDK or the AWS CLI to make API requests to AWS services using AWS Identity and Access Management (IAM) permissions. Applications must sign their AWS API requests with AWS credentials. **IAM roles for service accounts (IRSA)** provide the ability to manage credentials for your applications, similar to the way that Amazon EC2 instance profiles provide credentials to Amazon EC2 instances. Instead of creating and distributing your AWS credentials to the containers or using the Amazon EC2 instance's role, you associate an IAM role with a Kubernetes service account and configure your Pods to use the service account. You can't use IAM roles for service accounts with local clusters for Amazon EKS on AWS Outposts.

IAM roles for service accounts provide the following benefits:

- **Least privilege** – You can scope IAM permissions to a service account, and only Pods that use that service account have access to those permissions. This feature also eliminates the need for third-party solutions such as `kiam` or `kube2iam`.
- **Credential isolation** – When access to the Amazon EC2 Instance Metadata Service (IMDS) is restricted, a Pod's containers can only retrieve credentials for the IAM role that's associated with the service account that the container uses. A container never has access to credentials that are used by other containers in other Pods. If IMDS is not restricted, the Pod's containers also have access to the Amazon EKS node IAM role and the containers may be able to gain access to credentials of IAM roles of other Pods on the same node. For more information, see Restrict access to the instance profile assigned to the worker node.

Note

Pods configured with `hostNetwork: true` will always have IMDS access, but the AWS SDKs and CLI will use IRSA credentials when enabled.

- **Auditability** – Access and event logging is available through AWS CloudTrail to help ensure retrospective auditing.

Important

Containers are not a security boundary, and the use of IAM roles for service accounts does not change this. Pods assigned to the same node will share a kernel and potentially other resources depending on your Pod configuration. While Pods running on separate nodes will be isolated at the compute layer, there are node applications that have additional permissions in the Kubernetes API beyond the scope of an individual instance. Some examples are `kubelet`, `kube-proxy`, CSI storage drivers, or your own Kubernetes applications.

Enable IAM roles for service accounts by completing the following procedures:

1. Create an IAM OIDC provider for your cluster – You only complete this procedure once for each cluster.

   Note

   If you enabled the EKS VPC endpoint, the EKS OIDC service endpoint couldn't be accessed from inside that VPC. Consequently, your operations such as creating an OIDC provider with `eksctl` in the VPC will not work and will result in a timeout when attempting to request `https://oidc.eks. region .amazonaws.com`. An example error message follows:

   ```
   server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
   ```

   To complete this step, you can run the command outside the VPC, for example in AWS CloudShell or on a computer connected to the internet. Alternatively, you can create a split-horizon conditional resolver in the VPC, such as Route 53 Resolver to use a different resolver for the OIDC Issuer URL and not use the VPC DNS for it. For an example of conditional forwarding in CoreDNS, see the Amazon EKS feature request on GitHub.

2. Assign IAM roles to Kubernetes service accounts – Complete this procedure for each unique set of permissions that you want an application to have.

3. Configure Pods to use a Kubernetes service account – Complete this procedure for each Pod that needs access to AWS services.

4. Use IRSA with the AWS SDK – Confirm that the workload uses an AWS SDK of a supported version and that the workload uses the default credential chain.

IAM, Kubernetes, and OpenID Connect (OIDC) background information

In 2014, AWS Identity and Access Management added support for federated identities using OpenID Connect (OIDC). This feature allows you to authenticate AWS API calls with supported identity providers and receive a valid OIDC JSON web token (JWT). You can pass this token to the AWS STS `AssumeRoleWithWebIdentity` API

operation and receive IAM temporary role credentials. You can use these credentials to interact with any AWS service, including Amazon S3 and DynamoDB.

Each JWT token is signed by a signing key pair. The keys are served on the OIDC provider managed by Amazon EKS and the private key rotates every 7 days. Amazon EKS keeps the public keys until they expire. If you connect external OIDC clients, be aware that you need to refresh the signing keys before the public key expires. Learn how to Fetch signing keys to validate OIDC tokens.

Kubernetes has long used service accounts as its own internal identity system. Pods can authenticate with the Kubernetes API server using an auto-mounted token (which was a non-OIDC JWT) that only the Kubernetes API server could validate. These legacy service account tokens don't expire, and rotating the signing key is a difficult process. In Kubernetes version `1.12`, support was added for a new `ProjectedServiceAccountToken` feature. This feature is an OIDC JSON web token that also contains the service account identity and supports a configurable audience.

Amazon EKS hosts a public OIDC discovery endpoint for each cluster that contains the signing keys for the `ProjectedServiceAccountToken` JSON web tokens so external systems, such as IAM, can validate and accept the OIDC tokens that are issued by Kubernetes.
