Managing access to AWS resources for your applications

Document Level Classification

200

Introduction

Your application (*Pods*) running in a Substrate cluster (Amazon EKS) is assigned an identity known as a *ServiceAccount*. Each *Pod* uses a *ServiceAccount* named `default` unless configured otherwise. You can map the *ServiceAccount* to an AWS Identity and Access Management (IAM) role to grant access to AWS resources.

To provide your application access to AWS resources:

1. Create an IAM role with appropriate least-privilege IAM policy attached.
2. Configure a *ServiceAccount* and map it to the IAM role.

3. Use a supported AWS SDK in your application.

The recommended naming convention is:

- Use a lower kebab-case, with a `svc-` prefix (for example, `svc-my-service-name`).
- Use the same name for the IAM role and *ServiceAccount*.

Create an IAM role for your application

The IAM role you create needs to trust the OpenID Connect (OIDC) federated identity provider associated with your Substrate cluster. Your Substrate cluster is already provisioned and configured with an OIDC provider, typically in the format `oidc.eks.us-east-1.amazonaws.com/id/<UNIQUE_HEX_ID>`. The OIDC provider for your Substrate cluster is visible in the Overview tab when browsing using the AWS Management Console.

Use the following snippet as an example template for the trust policy when creating your IAM role, making sure to replace the placeholders appropriately:

Example IAM trust policy snippet

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<AWS_ACCOUNT_ID>:oidc-provider/oidc.
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.<AWS_REGION>.amazonaws.com/id/<UNIQUE_HEX_ID>:sub":
          "oidc.eks.<AWS_REGION>.amazonaws.com/id/<UNIQUE_HEX_ID>:aud":
        }
      }
    }
  ]
}
```

You can attach either IAM Manage policies or a custom inline policy to the role, using principles of least-privilege.

Example terraform code to provision an IAM role

```
####################################################################################
# TERRAFORM SNIPPET TO PROVISION AN IAM ROLE FOR KUBERNETES SERVICE-ACC
####################################################################################

#
# Reference:
# * https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-serv
#

# ------------------------------------------------------------------------------------
# variables.tf
# ------------------------------------------------------------------------------------

variable "substrate_cluster_name" {
  type        = string
  description = "Name of the Substrate cluster. For example, abcd-dev"
}

variable "substrate_cluster_namespace" {
  type        = string
  description = "The Substrate cluster namespace. For example, team-xyz
}

variable "service_name" {
  type        = string
  description = "Your service name. For example, my-service"
}
```

```
# ------------------------------------------------------------------------
# main.tf
# ------------------------------------------------------------------------

provider "aws" {
  region = "us-east-1"
}

data "aws_caller_identity" "this" {}

data "aws_eks_cluster" "this" {
  name = var.substrate_cluster_name
}

locals {
  oidc_id = replace(
    data.aws_eks_cluster.this.identity[0].oidc[0].issuer,
    "https://",
    "",
  )

  iam_role_name = "svc-${var.service_name}"
}

data "aws_iam_policy_document" "trust_policy" {
  statement {
    effect = "Allow"

    principals {
      type = "Federated"
      identifiers = [
        "arn:aws:iam::${data.aws_caller_identity.this.account_id}:oidc-
      ]
    }

    actions = [
      "sts:AssumeRoleWithWebIdentity",
    ]

    condition {
      test     = "StringEquals"
      variable = "${local.oidc_id}:sub"
      values = [
        "system:serviceaccount:${var.substrate_cluster_namespace}:${loc
      ]
    }

    condition {
      test     = "StringEquals"
      variable = "${local.oidc_id}:aud"
      values = [
        "sts.amazonaws.com",
      ]
    }
  }
}

resource "aws_iam_role" "this" {
  name               = local.iam_role_name
  description        = "IAM Role for Kubernetes ServiceAccount ${var.su
  assume_role_policy = data.aws_iam_policy_document.trust_policy.json
}

data "aws_iam_policy_document" "inline_policy" {
  statement {
    effect    = "Allow"
    actions   = ["sts:GetCallerIdentity"]
    resources = ["*"]
  }
  # TODO: Add more statements
}

resource "aws_iam_role_policy" "inline_policy" {
  role   = aws_iam_role.this.id
  policy = data.aws_iam_policy_document.inline_policy.json
}

# ----------------------------------------------------------------------
# outputs.tf
# ----------------------------------------------------------------------

output "aws_iam_role" {
  depends_on = [
    aws_iam_role_policy.inline_policy
  ]

  description = "IAM Role object"
  value       = aws_iam_role.this
}

########################################################################
```

Configure the *ServiceAccount* for your application

To provide your application access to AWS resources, or to other resources outside the Substrate cluster (for example, Substrate Vault), it is recommended to create a *ServiceAccount* specific to your application and not use `default`. By convention, the name you choose for the *ServiceAccount* matches your application name.

To configure a ServiceAccount:

1. If you are using [epic-app](), see the example snippet below.
2. If you are not using epic-app, you will need to create a *ServiceAccount* object and [update your Pod specification]().

⌄ Example: configure a ServiceAccount using epic-app

```
epic-app:
  # ServiceAccount configurations
  serviceAccount:
    # Specifies whether a ServiceAccount should be created
    create: true
    # The name of the ServiceAccount to use.
    name: <SERVICE_ACCOUNT_NAME>
    # Annotations to add to the ServiceAccount
    annotations:
      # Amazon Resource Name (ARN) of the IAM role that you want the Se
      eks.amazonaws.com/role-arn: arn:aws:iam::<AWS_ACCOUNT_ID>:role/<I

  # NOTE:
  # If a custom ServiceAccount is used and your Pod is using Docker ima
  # Epic's Artifactory, you must enable imagePullSecrets
  imagePullSecrets:
    - name: epic-artifactory-registry
```

Use a supported AWS SDK

Newer versions of AWS SDKs use a default credential provider chain to discover and use the IAM role mapped to your *ServiceAccount*. It is recommended that you use the minimum supported AWS SDK version, and not hard-code credential configurations.

With a *ServiceAccount* and associated IAM role mapping, an `AWS_WEB_IDENTITY_TOKEN_FILE` environment variable is set for your Pod that points to an automatically mounted *ServiceAccount*'s API credentials file. AWS SDKs look for use this environment variable to exchange *ServiceAccount* tokens for temporary IAM credentials using `AssumeRoleWithWebIdentity`. This credential provider chain will handle in-memory caching as well as refreshing credentials, as needed.