department-of-veterans-affairs / **devops** Internal

Code | Pull requests 8 | Discussions | Actions | Security 43 | Insights

We found potential security vulnerabilities in your dependencies.

You can see this message because you have been granted access to Dependabot alerts for this repository.

See Dependabot alerts

master

devops / terraform / environments / dsva-vagov-dev / **eks.tf**

 **mleclerc00** Test redirecting traffic to eks vets-api instance based on http head... ... ✓ History

9 contributors 

564 lines (507 sloc) | 16.8 KB

```
1  locals {
2    eks_cluster_name = "${var.base_tags.group}-${var.base_tags.project}-${var.base_tags["environment
3  }
4
5  module "dev_eks_cluster" {
6    source                       = "git::https://github.com/cloudposse/terraform-aws-eks-cluster.git?re
7    region                       = "us-gov-west-1"
8    vpc_id                       = aws_vpc.vpc.id
9    subnet_ids                   = [aws_subnet.subnet_1a.id, aws_subnet.subnet_1b.id, aws_subnet.subnet
10   kubernetes_version           = "1.19"
11   name                         = local.eks_cluster_name
12   oidc_provider_enabled        = true
13   enabled_cluster_log_types    = ["api", "audit", "authenticator", "controllerManager", "scheduler"]
14   endpoint_private_access      = true
15   endpoint_public_access       = true
16   # Temporarily disabled until a solution for changes via tf outside Ops team is determined - stev
17   apply_config_map_aws_auth    = false
18   map_additional_iam_users = [
19     {
20       userarn  = "arn:aws-us-gov:iam::008577686731:user/Jeremy.Britt",
21       username = "Jeremy.Britt"
```

```
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Demian.Ginther",
    username = "Demian.Ginther"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Srikanth.Valluru",
    username = "Srikanth.Valluru"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Matt.Leclerc",
    username = "Matt.Leclerc"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Eric.Oliver",
    username = "Eric.Oliver"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Elijah.Lynn",
    username = "Elijah.Lynn"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Nathan.Douglas",
    username = "Nathan.Douglas"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Cameron.Eagans",
    username = "Cameron.Eagans"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Neil.Hastings",
    username = "Neil.Hastings"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/John.Stevens-Garmon",
    username = "John.Stevens-Garmon"
    groups   = ["system:masters"]
  },
  {
    userarn  = "arn:aws-us-gov:iam::008577686731:user/Ryan.Beckwith",
```

```
      username = "Ryan.Beckwith"
      groups   = ["system:masters"]
    },
    {
      userarn  = "arn:aws-us-gov:iam::008577686731:user/Oseas.Moran",
      username = "Oseas.Moran"
      groups   = ["system:masters"]
    },
    {
      userarn  = "arn:aws-us-gov:iam::008577686731:user/Rebecca.Tolmach",
      username = "Rebecca.Tolmach"
      groups   = ["system:masters"]
    },
    {
      userarn  = "arn:aws-us-gov:iam::008577686731:user/service_account/dsva-eks-service-account",
      username = "dsva-eks-service-account"
      groups   = ["system:masters"]
    },
  ]
  map_additional_iam_roles = [
    {
      rolearn  = "arn:aws-us-gov:iam::008577686731:role/dsva-vagov-utility-cluster-argocd"
      username = "argocd-application-controller"
      groups   = []
    },
    {
      rolearn  = "arn:aws-us-gov:iam::008577686731:role/dsva-vagov-utility-cluster-argocd-server"
      username = "argocd-server"
      groups   = []
    },
  ]
}

module "eks_node_asg_group" {
  source                           = "github.com/department-of-veterans-affairs/terraform-aws-vsp
  region                           = "us-gov-west-1"
  vpc_id                           = aws_vpc.vpc.id
  desired_capacity                 = 5
  min_size                         = 3
  max_size                         = 20
  eks_cluster_security_group_id    = module.dev_eks_cluster.security_group_id
  cluster_name                     = module.dev_eks_cluster.eks_cluster_id
  eks_alb_ingress_security_group_id = module.internal-tools-alb.alb-sg-id
  name                             = "dsva-vagov-dev-eks-node-group-main"
  subnet_ids                       = [aws_subnet.subnet_1a.id, aws_subnet.subnet_1b.id, aws_subne
  instance_types                   = ["c5.4xlarge", "c5a.4xlarge", "c5n.4xlarge", "c5d.4xlarge"]
  ## After initial creation AMI ID is controlled by GHA https://github.com/department-of-veterans-
  ami_id         = "ami-0ce45355d016dc86b"
  bootstrap_args = "--use-max-pods false"
```

```
  target_group_arns = [
    aws_lb_target_group.traefik_eks_dev.arn,
    aws_lb_target_group.traefik_eks_dev_https.arn,
  ]
}

resource "aws_security_group_rule" "revproxy-to-eks-worker" {
  type                     = "ingress"
  from_port                = 0
  to_port                  = 0
  protocol                 = "-1"
  security_group_id        = module.eks_node_asg_group.asg_sg_id
  source_security_group_id = module.revproxy.revproxy-alb-security-group
  description              = "revproxy ALB"
}

## External Secrets Service Role
data "aws_iam_policy_document" "external_secrets_controller_dev" {
  statement {
    actions = ["sts:AssumeRoleWithWebIdentity"]
    effect  = "Allow"
    condition {
      test     = "StringEquals"
      variable = "${replace(module.dev_eks_cluster.eks_cluster_identity_oidc_issuer, "https://", "
      values = [
        "system:serviceaccount:external-secrets:external-secrets-controller",
      ] // IMPORTANT! This must match the serviceaccount in k8s 'system:serviceaccount:<namespace>
    }
    principals {
      identifiers = [module.dev_eks_cluster.eks_cluster_identity_oidc_issuer_arn]
      type        = "Federated"
    }
  }
}

data "aws_iam_policy_document" "external_secrets_controller_policy_dev" {
  statement {
    effect = "Allow"
    actions = [
      "ssm:DescribeParameters",
      "ssm:GetParameter",
      "ssm:GetParametersByPath"
    ]
    resources = ["*"]
  }
}

resource "aws_iam_role" "external_secrets_controller_dev" {
  assume_role_policy = data.aws_iam_policy_document.external_secrets_controller_dev.json
```

```
  name               = "${module.dev_eks_cluster.eks_cluster_id}-ext-secrets-controller-dev"
}

resource "aws_iam_role_policy" "external_secrets_controller_dev" {
  name   = "${module.dev_eks_cluster.eks_cluster_id}-External-Secrets-Controller-Policy-dev"
  role   = aws_iam_role.external_secrets_controller_dev.id
  policy = data.aws_iam_policy_document.external_secrets_controller_policy_dev.json
}

## Traefik target group and listener rules
resource "aws_lb_target_group" "traefik_eks_dev" {
  name        = "dsva-vagov-dev-traefik-tg"
  port        = 32112
  protocol    = "HTTP"
  vpc_id      = aws_vpc.vpc.id
  target_type = "instance"

  deregistration_delay = 10

  health_check {
    path     = "/ping"
    port     = 32112
    protocol = "HTTP"
  }
}

resource "aws_lb_target_group" "traefik_eks_dev_https" {
  # a temporary target group used for testing revproxy in EKS
  # since a target group can't be assigned to multiple listeners
  name        = "dsva-vagov-dev-https-traefik-tg"
  port        = 32494
  protocol    = "HTTPS"
  vpc_id      = aws_vpc.vpc.id
  target_type = "instance"

  deregistration_delay = 10

  health_check {
    path     = "/ping"
    port     = 32494
    protocol = "HTTPS"
  }
}

### VETS-API ALB to EKS ###
module "vets-api-elb-logs-bucket" {
  source             = "github.com/department-of-veterans-affairs/terraform-aws-vsp-s3-bucket"
  name               = "dsva-vagov-dev-vets-api-alb-logs"
  environment        = "dev"
```

```
  access_logs_policy = true
}

resource "aws_security_group_rule" "alb-to-worker" {
  type                     = "ingress"
  description              = "ALB to EKS worker" # allow new ALB to hit the EKS worker
  from_port                = 0
  to_port                  = 0
  protocol                 = "-1"
  source_security_group_id = module.vets-api.elb-security-group
  security_group_id        = module.eks_node_asg_group.asg_sg_id
}

##########################
## Temporary rule to test routing traffic based on http header to eks vets-api
resource "aws_lb_listener_rule" "vets_api_eks" {
  listener_arn = module.internal-tools-alb.listener-http-arn
  priority     = 49001

  action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.traefik_eks_dev.arn
  }

  condition {
    host_header {
      values = ["dev-api.va.gov"]
    }
  }
}

resource "aws_lb_listener_rule" "traefik_http" {
  listener_arn = module.internal-tools-alb.listener-http-arn
  priority     = 49000

  action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.traefik_eks_dev.arn
  }

  condition {
    host_header {
      values = ["*.vfs.va.gov"]
    }
  }
}

resource "aws_lb_listener_rule" "traefik_https" {
  listener_arn = module.internal-tools-alb.listener-https-arn
```

```
  priority     = 50000

  action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.traefik_eks_dev_https.arn
  }

  condition {
    host_header {
      values = ["*.vfs.va.gov"]
    }
  }
}

## External DNS Service Role
data "aws_iam_policy_document" "external_dns_controller_dev" {
  statement {
    actions = ["sts:AssumeRoleWithWebIdentity"]
    effect  = "Allow"
    condition {
      test     = "StringEquals"
      variable = "${replace(module.dev_eks_cluster.eks_cluster_identity_oidc_issuer, "https://", "
      values = [
        "system:serviceaccount:external-dns:external-dns",
      ] // IMPORTANT! This must match the serviceaccount in k8s 'system:serviceaccount:<namespace>
    }
    principals {
      identifiers = [module.dev_eks_cluster.eks_cluster_identity_oidc_issuer_arn]
      type        = "Federated"
    }
  }
}

data "aws_iam_policy_document" "external_dns_controller_policy_dev" {
  statement {
    effect = "Allow"
    actions = [
      "route53:ListHostedZones",
      "route53:ListResourceRecordSets"
    ]
    resources = ["*"]
  }

  statement {
    effect = "Allow"
    actions = [
      "route53:ChangeResourceRecordSets"
    ]
    resources = ["arn:aws-us-gov:route53:::hostedzone/Z10020332CIPNJYY8UL7O"]
```

```
316     }
317   }
318
319   resource "aws_iam_role" "external_dns_controller_dev" {
320     assume_role_policy = data.aws_iam_policy_document.external_dns_controller_dev.json
321     name               = "${module.dev_eks_cluster.eks_cluster_id}-ext-dns-controller-dev"
322   }
323
324   resource "aws_iam_role_policy" "external_dns_controller_dev" {
325     name   = "${module.dev_eks_cluster.eks_cluster_id}-External-DNS-Controller-Policy-dev"
326     role   = aws_iam_role.external_dns_controller_dev.id
327     policy = data.aws_iam_policy_document.external_dns_controller_policy_dev.json
328   }
329
330   ## Loki Service Role
331   data "aws_iam_policy_document" "loki" {
332     statement {
333       actions = ["sts:AssumeRoleWithWebIdentity"]
334       effect  = "Allow"
335
336       condition {
337         test     = "StringEquals"
338         variable = "${replace(module.dev_eks_cluster.eks_cluster_identity_oidc_issuer, "https://", "
339         values = [
340           "system:serviceaccount:observability:loki",
341         ] // IMPORTANT! This must match the serviceaccount in k8s 'system:serviceaccount:<namespace>
342       }
343
344       principals {
345         identifiers = [module.dev_eks_cluster.eks_cluster_identity_oidc_issuer_arn]
346         type        = "Federated"
347       }
348     }
349   }
350
351   data "aws_iam_policy_document" "loki_policy" {
352     statement {
353       sid    = "LokiAccessToS3"
354       effect = "Allow"
355       actions = [
356         "s3:PutObject",
357         "s3:GetObject",
358         "s3:ListBucket",
359         "s3:DeleteObject"
360       ]
361       resources = [
362         aws_s3_bucket.loki-dev-s3.arn,
363         "${aws_s3_bucket.loki-dev-s3.arn}/*"
364       ]
```

```
  }
}

resource "aws_iam_role" "loki_role" {
  assume_role_policy = data.aws_iam_policy_document.loki.json
  name               = "${module.dev_eks_cluster.eks_cluster_id}-loki"
}

resource "aws_iam_role_policy" "loki_policy" {
  name   = "${module.dev_eks_cluster.eks_cluster_id}-Loki-Policy"
  role   = aws_iam_role.loki_role.id
  policy = data.aws_iam_policy_document.loki_policy.json
}

## Vets-api Service Role
data "aws_iam_policy_document" "vets_api_dev" {
  statement {
    actions = ["sts:AssumeRoleWithWebIdentity"]
    effect  = "Allow"
    condition {
      test     = "StringEquals"
      variable = "${replace(module.dev_eks_cluster.eks_cluster_identity_oidc_issuer, "https://", "
      values = [
        "system:serviceaccount:vets-api:vets-api",
      ] // IMPORTANT! This must match the serviceaccount in k8s 'system:serviceaccount:<namespace>
    }
    principals {
      identifiers = [module.dev_eks_cluster.eks_cluster_identity_oidc_issuer_arn]
      type        = "Federated"
    }
  }
}

data "aws_iam_policy_document" "vets_api_policy_dev" {
  statement {
    effect = "Allow"
    actions = [
      "s3:ListAllMyBuckets",
      "s3:GetBucketLocation"
    ]
    resources = ["*"]
  }

  statement {
    effect = "Allow"
    actions = [
      "s3:ListBucket"
    ]
    resources = ["arn:aws-us-gov:s3:::dsva-vagov-dev-config-bucket"]
```

```
414     }
415
416     statement {
417       effect = "Allow"
418       actions = [
419         "s3:GetObject"
420       ]
421       resources = ["arn:aws-us-gov:s3:::dsva-vagov-dev-config-bucket/vets-api-server/*"]
422     }
423
424     statement {
425       effect = "Allow"
426       actions = [
427         "ecr:StartImageScan",
428         "ecr:DescribeImageReplicationStatus",
429         "ecr:ListTagsForResource",
430         "ecr:UploadLayerPart",
431         "ecr:ListImages",
432         "ecr:CompleteLayerUpload",
433         "ecr:TagResource",
434         "ecr:DescribeRepositories",
435         "ecr:BatchCheckLayerAvailability",
436         "ecr:ReplicateImage",
437         "ecr:GetLifecyclePolicy",
438         "ecr:DescribeImageScanFindings",
439         "ecr:GetLifecyclePolicyPreview",
440         "ecr:PutImageScanningConfiguration",
441         "ecr:GetDownloadUrlForLayer",
442         "ecr:PutImage",
443         "ecr:BatchGetImage",
444         "ecr:DescribeImages",
445         "ecr:InitiateLayerUpload",
446         "ecr:GetRepositoryPolicy",
447       ]
448       resources = ["arn:aws-us-gov:ecr:us-gov-west-1:008577686731:repository/dsva/vets-api*"]
449     }
450   }
451
452   resource "aws_iam_role" "vets_api_dev" {
453     assume_role_policy = data.aws_iam_policy_document.vets_api_dev.json
454     name               = "${module.dev_eks_cluster.eks_cluster_id}-vets-api"
455   }
456
457   resource "aws_iam_role_policy" "vets_api_dev" {
458     name   = "${module.dev_eks_cluster.eks_cluster_id}-vets-api-policy"
459     role   = aws_iam_role.vets_api_dev.id
460     policy = data.aws_iam_policy_document.vets_api_policy_dev.json
461   }
462
```

```
## Grafana Service Role
data "aws_iam_policy_document" "grafana" {
  statement {
    actions = ["sts:AssumeRoleWithWebIdentity"]
    effect  = "Allow"

    condition {
      test     = "StringEquals"
      variable = "${replace(module.dev_eks_cluster.eks_cluster_identity_oidc_issuer, "https://", "
      values = [
        "system:serviceaccount:observability:grafana",
      ] // IMPORTANT! This must match the serviceaccount in k8s 'system:serviceaccount:<namespace>
    }

    principals {
      identifiers = [module.dev_eks_cluster.eks_cluster_identity_oidc_issuer_arn]
      type        = "Federated"
    }
  }
}

data "aws_iam_policy_document" "grafana_policy" {
  statement {
    sid       = "AllowCloudWatchLogging"
    effect    = "Allow"
    resources = ["arn:aws-us-gov:logs:*:*:*"]

    actions = [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents",
      "logs:DecribeLogStreams",
    ]
  }

  statement {
    sid       = "AllowLifecycleAction"
    effect    = "Allow"
    resources = ["*"]
    actions   = ["autoscaling:CompleteLifecycleAction"]
  }

  statement {
    sid       = "AllowLogsReadAccess"
    effect    = "Allow"
    resources = ["*"]

    actions = [
      "logs:Describe*",
```

```
      "logs:Get*",
      "logs:List*",
      "logs:TestMetricFilter",
      "logs:FilterLogEvents",
    ]
  }

  statement {
    sid       = "AllowEventsReadAccess"
    effect    = "Allow"
    resources = ["*"]

    actions = [
      "events:DescribeRule",
      "events:ListRuleNamesByTarget",
      "events:ListRules",
      "events:ListTargetsByRule",
      "events:TestEventPattern",
      "events:DescribeEventBus",
    ]
  }

  statement {
    sid       = "AllowCloudwatchReadAccess"
    effect    = "Allow"
    resources = ["*"]

    actions = [
      "autoscaling:Describe*",
      "cloudwatch:Describe*",
      "cloudwatch:Get*",
      "cloudwatch:List*",
      "sns:Get*",
      "sns:List*",
    ]
  }
}

resource "aws_iam_role" "grafana_role" {
  assume_role_policy = data.aws_iam_policy_document.grafana.json
  name               = "${module.dev_eks_cluster.eks_cluster_id}-grafana-amt"
}

resource "aws_iam_role_policy" "grafana_policy" {
  name   = "${module.dev_eks_cluster.eks_cluster_id}-Grafana-Policy"
  role   = aws_iam_role.grafana_role.id
  policy = data.aws_iam_policy_document.grafana_policy.json
}

```

```
561 resource "aws_iam_role_policy_attachment" "grafana_cw_readonly_attach" {
562   role       = aws_iam_role.grafana_role.name
563   policy_arn = "arn:aws-us-gov:iam::aws:policy/CloudWatchReadOnlyAccess"
564 }
```