resource "aws\_s3\_bucket" "terraform\_state" {

  count = length(var.bucket\_name)

  bucket = "${var.bucket\_name[count.index]}"

  tags = {

    Name        = var.name

    Environment = var.environment

    ProjectName= var.project\_name

    WorkOrder= var.work\_order

  }

}

resource "aws\_s3\_bucket\_versioning" "enabled" {

  count = length(var.bucket\_name)

  bucket = aws\_s3\_bucket.terraform\_state[count.index].id

  versioning\_configuration {

    status = "Enabled"

  }

}

resource "aws\_s3\_object" "folder" {

  # count = length(var.bucket\_name)

  bucket = aws\_s3\_bucket.terraform\_state[0].id

  # count = length(var.default\_s3\_content)

  for\_each = var.default\_s3\_content

  #key    = "sap/ausp\_function\_location/ausp/cabn/circuit/equipment" # This creates a folder-like object

  acl    = "private"

  key = "sap/${each.value}/"

  }

resource "aws\_s3\_bucket\_server\_side\_encryption\_configuration" "default" {

  count = length(var.bucket\_name)

  bucket = aws\_s3\_bucket.terraform\_state[count.index].id

  rule {

    apply\_server\_side\_encryption\_by\_default {

      kms\_master\_key\_id = "arn:aws:kms:us-west-2:420393866958:key/d81fc091-ef05-4a84-a08e-aaa0095262b4"

      sse\_algorithm     = "aws:kms"

    }

  }

}

resource "aws\_s3\_bucket\_public\_access\_block" "public\_access" {

  count = length(var.bucket\_name)

  bucket = aws\_s3\_bucket.terraform\_state[count.index].id

  block\_public\_acls       = true

  block\_public\_policy     = true

  ignore\_public\_acls      = true

  restrict\_public\_buckets = true

}

// resource "aws\_s3\_bucket\_notification" "bucket\_notification" {

//   depends\_on = [ aws\_s3\_bucket.terraform\_state ]

//   bucket      = aws\_s3\_bucket.terraform\_state[0].id

//   eventbridge = true

//   }

#   resource "aws\_sns\_topic\_policy" "example" {

#   arn = var.topic\_arn

#   policy = jsonencode({

#     Version = "2012-10-17",

#     Statement = [

#       {

#         Effect    = "Allow",

#         Principal = {

#           Service = "s3.amazonaws.com"

#         },

#         Action    = "sns:Publish",

#         Resource  = var.topic\_arn

#       }

#     ]

#   })

# }

//  resource "aws\_s3\_bucket\_notification" "bucket\_notification\_error" {

//    depends\_on = [ aws\_s3\_bucket.terraform\_state,aws\_sns\_topic\_policy.example ]

//    bucket      = aws\_s3\_bucket.terraform\_state[2].id

//    topic {

//       topic\_arn = var.topic\_arn

//       events = ["s3:ObjectCreated:\*"]

//    }

//   }

#   resource "aws\_s3\_object" "s3\_lambda\_deployment\_package" {

#   depends\_on = [ aws\_s3\_bucket.terraform\_state ]

#   count = length(var.lambda\_deployment\_package\_bucket\_key)

#   bucket      = aws\_s3\_bucket.terraform\_state[1].id

#   key    = var.lambda\_deployment\_package\_bucket\_key[count.index]

#   source = "${path.module}/${var.lambda\_deployment\_package\_bucket\_source\_path[count.index]}"

#   etag = filemd5("${path.module}/${var.lambda\_deployment\_package\_bucket\_source\_path[count.index]}")

# }

# Define IAM Role

resource "aws\_iam\_role" "s3\_access\_role" {

  name = "s3-access-role"

  assume\_role\_policy = jsonencode({

    Version = "2012-10-17",

    Statement = [

      {

        Effect = "Allow",

        Principal = {

          Service = "ec2.amazonaws.com" # Example service, change as needed

        },

        Action = "sts:AssumeRole"

      }

    ]

  })

}

# Define IAM Policy

resource "aws\_iam\_policy" "s3\_bucket\_management\_policy" {

  count = length(var.bucket\_name)

  name        = "S3BucketManagementPolicy"

  description = "IAM policy for managing S3 buckets"

  policy = jsonencode({

    Version = "2012-10-17",

    Statement = [

      {

        Effect = "Allow",

        Action = [

          "s3:CreateBucket",

          "s3:DeleteBucket",

          "s3:PutBucketPolicy",

          "s3:PutBucketTagging",

          "s3:GetBucketLocation",

          "s3:ListBucket",

          "s3:GetBucketAcl",

          "s3:PutBucketAcl",

          "s3:PutBucketLogging",

          "s3:PutBucketNotification",

          "s3:PutBucketVersioning",

          "s3:PutEncryptionConfiguration",

          "s3:GetEncryptionConfiguration",

          "s3:PutBucketPublicAccessBlock",

          "s3:GetBucketPublicAccessBlock"

        ],

        Resource = "arn:aws:s3:::${var.bucket\_name[count.index]}"

      },

      {

        Effect = "Allow",

        Action = [

          "s3:PutObject",

          "s3:GetObject",

          "s3:DeleteObject",

          "s3:ListBucketMultipartUploads",

          "s3:AbortMultipartUpload"

        ],

        Resource = "arn:aws:s3:::${var.bucket\_name[count.index]}/\*"

      }

    ]

  })

}

# Attach Policy to Role

resource "aws\_iam\_role\_policy\_attachment" "s3\_bucket\_management\_policy\_attachment" {

  count = length(var.bucket\_name)

  role       = aws\_iam\_role.s3\_access\_role.name

  policy\_arn =  aws\_iam\_policy.s3\_bucket\_management\_policy[count.index].arn

}

sqs data IAM policy

data "aws\_iam\_policy\_document" "queue\_policy" {

  statement {

    principals {

      type        = "AWS"

      identifiers = ["arn:aws:iam::${data.aws\_caller\_identity.current.account\_id}:root"]

    }

    actions = ["SQS:\*"]

    resources = [

      aws\_sqs\_queue.queue.arn

    ]

  }

}

resource "aws\_sqs\_queue\_policy" "queue\_policy" {

  queue\_url = aws\_sqs\_queue.queue.id

  policy    = data.aws\_iam\_policy\_document.queue\_policy.json

  depends\_on = [aws\_sqs\_queue.queue]

}

lambda data IAM policy

data "aws\_iam\_policy\_document" "lambda\_assume\_role" {

  statement {

    effect = "Allow"

    actions = ["sts:AssumeRole"]

    principals {

      type        = "Service"

      identifiers = ["lambda.amazonaws.com"]

    }

  }

}

resource "aws\_iam\_role" "lambda\_execution\_role" {

  count = length(var.lambda\_function\_role\_name)

  name  = var.lambda\_function\_role\_name[count.index]

  assume\_role\_policy = data.aws\_iam\_policy\_document.lambda\_assume\_role.json

}

resource "aws\_iam\_policy" "lambda\_s3\_policy" {

  count       = length(var.lambda\_function\_policy\_name)

  name        = var.lambda\_function\_policy\_name[count.index]

  description = "IAM policy for the Lambda function"

  policy      = jsonencode({

    Version = "2012-10-17",

    Statement = [

      {

        Effect   = "Allow",

        Action   = [

          "logs:CreateLogGroup",

          "logs:CreateLogStream",

          "logs:PutLogEvents"

        ],

        Resource = "arn:aws:logs:\*:\*:\*"

      },

      {

        Effect   = "Allow",

        Action   = [

          "sqs:SendMessage",

          "sqs:ReceiveMessage",

          "sqs:DeleteMessage",

          "sqs:GetQueueAttributes"

        ],

        Resource = "\*"

      },

      {

        Effect   = "Allow",

        Action   = [

          "s3:GetObject"

        ],

        Resource = "arn:aws:s3:::${var.bucket\_name}"

      },

      {

        Effect   = "Allow",

        Action   = [

          "sns:Publish"

        ],

        Resource = "\*"

      }

    ]

  })

}

resource "aws\_iam\_role\_policy\_attachment" "lambda\_s3\_policy\_attachment" {

  count      = length(var.lambda\_function\_role\_name) # Assuming same count as IAM role

  role       = aws\_iam\_role.lambda\_execution\_role[count.index].name

  policy\_arn = aws\_iam\_policy.lambda\_s3\_policy[count.index].arn

}

resource "aws\_iam\_role\_policy\_attachment" "lambda\_policy\_attachment" {

  count      = length(var.lambda\_function\_role\_name) # Assuming same count as IAM role

  role       = aws\_iam\_role.lambda\_execution\_role[count.index].name

  policy\_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"

}

policy for glue log groups

IAM Policies for Glue Jobs

resource "aws\_iam\_policy" "glue\_job\_policies" {

  count = length(var.glue\_job\_name)

  name        = "${var.glue\_job\_name[count.index]}-policy"

  description = "IAM policy for ${var.glue\_job\_name[count.index]}"

  policy = jsonencode({

    Version = "2012-10-17",

    Statement = [

      {

        Effect = "Allow",

        Action = [

          "s3:GetObject",

          "s3:PutObject",

          "s3:ListBucket"

        ],

        Resource = [

          "arn:aws:s3:::your-bucket-name",

          "arn:aws:s3:::your-bucket-name/\*"

        ]

      },

      {

        Effect = "Allow",

        Action = [

          "logs:CreateLogGroup",

          "logs:CreateLogStream",

          "logs:PutLogEvents"

        ],

        Resource = "arn:aws:logs:\*:\*:\*"

      },

      {

        Effect = "Allow",

        Action = [

          "glue:\*"

        ],

        Resource = "\*"

      }

    ]

  })

}

# Attach IAM Policy to the Role

resource "aws\_iam\_role\_policy\_attachment" "glue\_job\_policy\_attachments" {

  count = length(var.glue\_job\_name)

  role       = aws\_iam\_role.glue\_job\_roles[count.index].name

  policy\_arn = aws\_iam\_policy.glue\_job\_policies[count.index].arn

}

 =========================================

Step 1: Create a Glue Database

resource "aws\_glue\_catalog\_database" "my\_database" {

  count = length(var.glue\_database\_name)

   name = "${var.glue\_database\_name[count.index]}"

}

# Step 2: Create a Glue Crawler

resource "aws\_glue\_crawler" "my\_crawler" {

  count = length(var.glue\_crawler\_name)

  name = "${var.glue\_crawler\_name[count.index]}"

  database\_name = aws\_glue\_catalog\_database.my\_database.name

  role = "arn:aws:iam::${aws\_caller\_identity.current.id}:role/CI-DataEngineeringRole" # Replace with your IAM role ARN

  s3\_target {

    path = "${var.bucket\_name}" # Replace with your S3 path

  }

}

# Step 3: Create a Glue Job

resource "aws\_glue\_job" "glue\_job" {

  count = length(var.glue\_job\_name)

  name = "${var.glue\_job\_name[count.index]}"

  role\_arn = "arn:aws:iam::${aws\_caller\_identity.current.id}:role/CI-DataEngineeringRole" # Replace with your IAM role ARN

  command {

    name = "glueetl"

    script\_location = "${var.glue\_job\_script\_s3\_path}/${var.glue\_job\_name[count.index]}.py" # Replace with your script location

    python\_version = "3"

  }

  default\_arguments = {

    "--TempDir" = "s3://your-temp-dir/${var.glue\_job\_name[count.index]}/"  # Replace with your temporary directory

    "--job-bookmark-option" = "job-bookmark-enable"

  }

  tags = {

    Name = var.glue\_job\_name[count.index]

  }

  timeout           = 30

  number\_of\_workers = 4

  worker\_type       = "G.1X"

  max\_capacity = 2.0 # Adjust based on your needs

}

# IAM Role for Glue Jobs

resource "aws\_iam\_role" "glue\_job\_roles" {

  count = length(var.glue\_job\_name)

  name = "${var.glue\_job\_name[count.index]}-role"

  assume\_role\_policy = jsonencode({

    Version = "2012-10-17",

    Statement = [

      {

        Effect = "Allow",

        Principal = {

          Service = "glue.amazonaws.com"

        },

        Action = "sts:AssumeRole"

      }

    ]

  })

}

# IAM Policies for Glue Jobs

resource "aws\_iam\_policy" "glue\_job\_policies" {

  count = length(var.glue\_job\_name)

  name        = "${var.glue\_job\_name[count.index]}-policy"

  description = "IAM policy for ${var.glue\_job\_name[count.index]}"

  policy = jsonencode({

    Version = "2012-10-17",

    Statement = [

      {

        Effect = "Allow",

        Action = [

          "s3:GetObject",

          "s3:PutObject",

          "s3:ListBucket"

        ],

        Resource = [

          "arn:aws:s3:::your-bucket-name",

          "arn:aws:s3:::your-bucket-name/\*"

        ]

      },

      {

        Effect = "Allow",

        Action = [

          "logs:CreateLogGroup",

          "logs:CreateLogStream",

          "logs:PutLogEvents"

        ],

        Resource = "arn:aws:logs:\*:\*:\*"

      },

      {

        Effect = "Allow",

        Action = [

          "glue:\*"

        ],

        Resource = "\*"

      }

    ]

  })

}

# Attach IAM Policy to the Role

resource "aws\_iam\_role\_policy\_attachment" "glue\_job\_policy\_attachments" {

  count = length(var.glue\_job\_name)

  role       = aws\_iam\_role.glue\_job\_roles[count.index].name

  policy\_arn = aws\_iam\_policy.glue\_job\_policies[count.index].arn

}

data "aws\_iam\_policy\_document" "lambda\_assume\_role" {

  statement {

    effect = "Allow"

    actions = ["sts:AssumeRole"]

    principals {

      type        = "Service"

      identifiers = ["lambda.amazonaws.com"]

    }

  }

}

resource "aws\_iam\_role" "lambda\_execution\_role" {

  count = length(var.lambda\_function\_role\_name)

  name  = var.lambda\_function\_role\_name[count.index]

  assume\_role\_policy = data.aws\_iam\_policy\_document.lambda\_assume\_role.json

}

resource "aws\_iam\_policy" "lambda\_s3\_policy" {

  count       = length(var.lambda\_function\_policy\_name)

  name        = var.lambda\_function\_policy\_name[count.index]

  description = "IAM policy for the Lambda function"

  policy      = jsonencode({

    Version = "2012-10-17",

    Statement = [

      {

        Effect   = "Allow",

        Action   = [

          "logs:CreateLogGroup",

          "logs:CreateLogStream",

          "logs:PutLogEvents"

        ],

        Resource = "arn:aws:logs:\*:\*:\*"

      },

      {

        Effect   = "Allow",

        Action   = [

          "sqs:SendMessage",

          "sqs:ReceiveMessage",

          "sqs:DeleteMessage",

          "sqs:GetQueueAttributes"

        ],

        Resource = "\*"

      },

      {

        Effect   = "Allow",

        Action   = [

          "s3:GetObject"

        ],

        Resource = "arn:aws:s3:::${var.bucket\_name}"

      },

      {

        Effect   = "Allow",

        Action   = [

          "sns:Publish"

        ],

        Resource = "\*"

      }

    ]

  })

}

resource "aws\_iam\_role\_policy\_attachment" "lambda\_s3\_policy\_attachment" {

  count      = length(var.lambda\_function\_role\_name) # Assuming same count as IAM role

  role       = aws\_iam\_role.lambda\_execution\_role[count.index].name

  policy\_arn = aws\_iam\_policy.lambda\_s3\_policy[count.index].arn

}

resource "aws\_iam\_role\_policy\_attachment" "lambda\_policy\_attachment" {

  count      = length(var.lambda\_function\_role\_name) # Assuming same count as IAM role

  role       = aws\_iam\_role.lambda\_execution\_role[count.index].name

  policy\_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"

}

data "archive\_file" "lambda\_code" {

  type        = "zip"

  source\_file = "${path.module}/pse-sapcee-validate-dev.py"

  output\_path = "${path.module}/pse-sapcee-validate-dev.zip"

}

resource "aws\_lambda\_function" "pse\_lambda" {

  count = length(var.lambda\_function\_name)

  function\_name = var.lambda\_function\_name[count.index]

  role          = aws\_iam\_role.lambda\_execution\_role[count.index].arn

  handler       = var.lambda\_handler

  runtime       = var.lambda\_runtime

  # s3\_bucket     = var.bucket\_name

  filename      = data.archive\_file.lambda\_code.output\_path

  environment {

    variables = var.lambda\_environment\_variables

  }

  tags = {

    Name        = var.lambda\_function\_name[count.index]

    Environment = var.environment

  }

  depends\_on = [aws\_iam\_role.lambda\_execution\_role]

}

resource "aws\_lambda\_event\_source\_mapping" "event\_source\_mapping" {

  event\_source\_arn = var.sqs\_queue\_arn

  enabled          = true

  function\_name    = aws\_lambda\_function.pse\_lambda[0].arn

  batch\_size       = 10

}

resource "aws\_lambda\_permission" "example\_lambda\_sqs\_permission" {

  statement\_id  = "AllowSQSTrigger"

  action        = "lambda:InvokeFunction"

  function\_name = aws\_lambda\_function.pse\_lambda[0].function\_name

  principal     = "sqs.amazonaws.com"

  source\_arn    = var.sqs\_queue\_arn

}

Sqs

data "aws\_caller\_identity" "current" {}

data "aws\_region" "current" {}

resource "aws\_sqs\_queue" "queue" {

  name = var.sqs\_name

  visibility\_timeout\_seconds = var.timeout

  redrive\_policy = jsonencode({

    deadLetterTargetArn = aws\_sqs\_queue.dlq.arn

    maxReceiveCount     = var.max\_recievecount

  })

}

resource "aws\_sqs\_queue" "dlq" {

  name =  "${var.sqs\_name}-dlq"

}

data "aws\_iam\_policy\_document" "queue\_policy" {

  statement {

    principals {

      type        = "AWS"

      identifiers = ["arn:aws:iam::${data.aws\_caller\_identity.current.account\_id}:root"]

    }

    actions = ["SQS:\*"]

    resources = [

      aws\_sqs\_queue.queue.arn

    ]

  }

}

resource "aws\_sqs\_queue\_policy" "queue\_policy" {

  queue\_url = aws\_sqs\_queue.queue.id

  policy    = data.aws\_iam\_policy\_document.queue\_policy.json

  depends\_on = [aws\_sqs\_queue.queue]

}

**Sns**

resource "aws\_sns\_topic" "this" {

  name = var.sns\_topic\_name

  tags = {

    Name        = var.sns\_topic\_name

    #Environment = var.environment

  }

}

resource "aws\_sns\_topic\_subscription" "this" {

  topic\_arn = aws\_sns\_topic.this.arn

  protocol  = "email"

  endpoint  = var.subscription\_email

}

 =======================

data "aws\_iam\_policy\_document" "GlueCrawlerLanding" {

  #AWS Glue service role

  statement {

    actions = [

      "glue:\*",

      "s3:GetBucketLocation",

      "s3:ListBucket",

      "s3:ListAllMyBuckets",

      "s3:GetBucketAcl",

      "cloudwatch:PutMetricData",

    ]

    resources = [

      "\*",

    ]

  }

  statement {

    actions = [

      "logs:CreateLogGroup",

      "logs:CreateLogStream",

      "logs:PutLogEvents"

    ]

    resources = [

      "arn:aws:logs:${data.aws\_region.current.name}:${data.aws\_caller\_identity.current.account\_id}:log-group:/aws-glue/\*",

    ]

  }

  #additional s3 access

  statement {

    actions = [

      "s3:GetObject",

    ]

    resources = [

      "arn:aws:s3:::${local.landing\_s3\_path}/\*",

    ]

  }

}

resource "aws\_iam\_role" "GlueCrawlerLanding" {

  name                  = "glue-crawler-landing-${local.resource\_suffix}"

  description           = "This role and all its configurations are managed by Terraform. See role tag TFC\_WORKSPACE\_NAME for the management location."

  assume\_role\_policy    = local.trustPolicyJsonMap.glue

  force\_detach\_policies = true

  managed\_policy\_arns   = []

}

resource "aws\_iam\_role\_policy" "GlueCrawlerLanding" {

  name   = "glue-crawler-landing-${local.resource\_suffix}"

  role   = aws\_iam\_role.GlueCrawlerLanding.id

  policy = data.aws\_iam\_policy\_document.GlueCrawlerLanding.json

}

resource "aws\_glue\_crawler" "Landing" {

  database\_name          = aws\_glue\_catalog\_database.LandingZone.name

  name                   = "landing-${local.resource\_suffix}"

  role                   = aws\_iam\_role.GlueCrawlerLanding.arn

  security\_configuration = var.glue\_security\_configuraton\_name

  #   configuration = jsonencode(

  #     {

  #       #   Grouping = {

  #       #     TableGroupingPolicy = "CombineCompatibleSchemas" #Create a single schema for each S3 path

  #       #     TableLevelConfiguration = 4 #Table level - optional

  #       #   }

  #       CrawlerOutput = {

  #         # Partitions = { AddOrUpdateBehavior = "InheritFromTable" } #Update all new and existing partitions with metadata from the table

  #         # Tables = {

  #         #     AddOrUpdateBehavior = "MergeNewColumns" #Add new columns only

  #         #     TableThreshold      = 15 #Maximum table threshold - optional

  #         # }

  #       }

  #       Version = 1

  #     }

  #   )

  schema\_change\_policy {

    delete\_behavior = "DEPRECATE\_IN\_DATABASE"

    # update\_behavior = "LOG" #Ignore the change and don't update the table in the data catalog

    update\_behavior = "UPDATE\_IN\_DATABASE" #Update the table definition in the data catalog (OR Add new columns only override)

  }

  lake\_formation\_configuration {

    account\_id                     = data.aws\_caller\_identity.current.account\_id

    use\_lake\_formation\_credentials = false

  }

  lineage\_configuration {

    crawler\_lineage\_settings = "DISABLE"

  }

  recrawl\_policy {

    recrawl\_behavior = "CRAWL\_EVERYTHING"

  }

  s3\_target {

    path = "s3://${local.landing\_s3\_path}"

  }

}

data "aws\_iam\_policy\_document" "GlueJobLandingtoRaw" {

  #AWS Glue service role

  statement {

    actions = [

      "glue:\*",

      "s3:GetBucketLocation",

      "s3:ListBucket",

      "s3:ListAllMyBuckets",

      "s3:GetBucketAcl",

      "cloudwatch:PutMetricData",

    ]

    resources = [

      "\*",

    ]

  }

  statement {

    actions = [

      "logs:CreateLogGroup",

      "logs:CreateLogStream",

      "logs:PutLogEvents"

    ]

    resources = [

      "arn:aws:logs:${data.aws\_region.current.name}:${data.aws\_caller\_identity.current.account\_id}:log-group:/aws-glue/\*",

    ]

  }

  #additional s3 access

  statement {

    actions = [

      "s3:GetObject",

    ]

    resources = [

      "arn:aws:s3:::${local.landing\_s3\_path}/\*",

    ]

  }

}

resource "aws\_iam\_role" "GlueJobLandingtoRaw" {

  name                  = "glue-job-landing-to-raw-${local.resource\_suffix}"

  description           = "This role and all its configurations are managed by Terraform. See role tag TFC\_WORKSPACE\_NAME for the management location."

  assume\_role\_policy    = local.trustPolicyJsonMap.glue

  force\_detach\_policies = true

  managed\_policy\_arns   = []

}

resource "aws\_iam\_role\_policy" "GlueJobLandingtoRaw" {

  name   = "glue-job-landing-to-raw-${local.resource\_suffix}"

  role   = aws\_iam\_role.GlueJobLandingtoRaw.id

  policy = data.aws\_iam\_policy\_document.GlueJobLandingtoRaw.json

}

resource "aws\_s3\_object" "GlueJobLandingtoRaw" {

  bucket = var.terraform\_artifact\_s3\_bucket\_name

  key    = "/glue-jobs/landing\_to\_raw.py"

  source = "${path.module}/files/glue-jobs/landing\_to\_raw.py"

  acl    = "private"

}

resource "aws\_glue\_job" "GlueJobLandingtoRaw" {

  name                   = "landing-to-raw-${local.resource\_suffix}"

  role\_arn               = aws\_iam\_role.GlueJobLandingtoRaw.arn

  security\_configuration = var.glue\_security\_configuraton\_name

  command {

    name            = "glueetl"                                                                               # glueetl for spark

    script\_location = "s3://${var.terraform\_artifact\_s3\_bucket\_name}${aws\_s3\_object.GlueJobLandingtoRaw.key}" # Replace with your script location

    python\_version  = "3"                                                                                     #Allowed values are 2, 3 or 3.9. Version 3 refers to Python 3.6.

  }

  default\_arguments = {

    # "--TempDir"             = "${var.glue\_script\_s3}/temporary/${var.glue\_job\_script\_path[count.index]}/" #Need to update path to take scripts from terraform directory

    # "--job-bookmark-option" = "job-bookmark-enable"

    // "--BUCKET\_NAME"           = "pse-datalake-source-420393866958"

    // "--DQ\_BUCKET\_NAME"        = "pse-sapecc-dl-dqlogs-dev-tf"

    // "--DYNAMODB\_TABLE\_NAME"   = "pse\_dl\_glue\_job\_audits-tf"

    // "--PROCESSED\_BUCKET\_NAME" = "pse-sapecc-dl-processed-dev-tf"

  }

  # Choose either `worker\_type` and `number\_of\_workers` or `max\_capacity`

  # Option 1: Using `worker\_type` and `number\_of\_workers`

  worker\_type       = "G.1X"

  number\_of\_workers = 4

  # Option 2: Using `max\_capacity`

  # max\_capacity      = 2.0 # Adjust based on your needs

}

1)  Event bridge scheduler  
module "eventbridge" {  
  source = "terraform-aws-modules/eventbridge/aws"

  create\_bus = false

  rules = {  
    crons = {  
      description         = "Run state machine everyday 10:00 UTC"  
      schedule\_expression = "cron(0 10 \* \* ? \*)"  
    }  
  }  
#2)  Event bridge scheduler to trigger Step function  
  targets = {  
    crons = [  
      {  
        name            = "your-awesome-state-machine"  
        arn             = "arn:aws:states:us-east-1:123456789012:stateMachine:your-awesome-state-machine"  
        attach\_role\_arn = true  
      }  
    ]  
  }

  sfn\_target\_arns   = ["arn:aws:states:us-east-1:123456789012:stateMachine:your-awesome-state-machine"]  
  attach\_sfn\_policy = true  
}

tfvars

name                       = "dataengineering"

environment                = "dev"

bucket\_name                = ["pse-datalake-source-420393866958-us-west-2-tf","pse-sapecc-dl-processed-dev-tf",

"pse-sapecc-dl-analytics-dev-tf","pse-sapecc-dl-dqlogs-dev-tf",

"pse-esri-dl-dqlogs-dev-tf","pse-sapecc-glue-assets-tf"]

lambda\_function\_name       = ["pse-sapcee-validate-dev-tf","pse\_data\_quality\_error\_notification-tf"]

project\_name               = "datalake"

work\_order                 = "143006900"

log\_retention\_days = 30

sns\_topic\_arn      = "arn:aws:sns:us-west-2:420393866958:pse-error-notification-tf"

default\_s3\_content = ["ausp\_function\_location", "ausp", "cabn", "circuit", "equipment"]

sqs\_name = "pse-s3-event-sqs-tf"

resource   "aws\_s3\_object"   "folder"  {   ... by Goel, Ashish - AccentureGoel, Ashish - Accenture (External)4:02 PM

resource "aws\_s3\_object" "folder" {

  bucket = aws\_s3\_bucket.terraform\_state[0].id

  for\_each = var.default\_s3\_content

  acl    = "private"

  key = "sap/${each.value}/"

  }

  resource "aws\_s3\_object" "folder2" {

  bucket = aws\_s3\_bucket.terraform\_state[0].id

  key    = "esri/"  # Creates a "subfolder2" in the S3 bucket

  acl    = "private"

}

   variable   "default\_s3\_content"  {   desc... by Goel, Ashish - AccentureGoel, Ashish - Accenture (External)4:02 PM

variable "default\_s3\_content" {

  description = "The default content of the s3 bucket upon creation of the bucket"

  type = set(string)

}

 module   "s3"  {   source       =  "./terra... by Goel, Ashish - AccentureGoel, Ashish - Accenture (External)4:03 PM

module "s3" {

  source       = "./terraform-modules/s3"

  name         = var.name

  environment  = var.environment

  bucket\_name  = var.bucket\_name

  project\_name = var.project\_name

  work\_order   = var.work\_order

  glue\_script\_location\_source\_key = var.glue\_script\_location\_source\_key

  glue\_script\_location\_source\_path = var.glue\_script\_location\_source\_path

  default\_s3\_content = var.default\_s3\_content

}

module "sqs" {

  source           = "./terraform-modules/sqs"

  sqs\_name         = var.sqs\_name

  max\_recievecount = var.max\_recievecount

  timeout          = var.timeout

}

module "event\_bridge" {

  source          = "./terraform-modules/eventBridge"

  event\_rule\_name = var.event\_rule\_name

  sqs\_queue\_arn   = module.sqs.sqs\_queue\_arn

  sqs\_queue\_url   = var.sqs\_queue\_url

  bucket\_name     = var.bucket\_name

  region          = var.region

}

module "sns\_topic" {

  source             = "./terraform-modules/sns"

  sns\_topic\_name     = var.sns\_topic\_name

has context menu