```
resource "aws_lb" "wikijs" {
  name               = "wikijs-alb"
  load_balancer_type = "application"
  subnets            = module.vpc.public_subnets
  security_groups    = [aws_security_group.alb_sg.id]

  tags = merge(local.common_tags, {
    Name = "wikijs-alb"
  })
}

resource "aws_lb_target_group" "wikijs" {
  name        = "wikijs-tg"
  port        = 3000
  protocol    = "HTTP"
  vpc_id      = module.vpc.vpc_id
  target_type = "ip"

  health_check {
    path                = "/"
    protocol            = "HTTP"
    matcher             = "200-399"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = merge(local.common_tags, {
    Name = "wikijs-tg"
  })
}

resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.wikijs.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.wikijs.arn
  }

  tags = merge(local.common_tags, {
    Name = "wikijs-http-listener"
  })
}resource "aws_cloudwatch_log_group" "wikijs" {
  name              = "/ecs/wikijs"
  retention_in_days = 14

  tags = {
    Name        = "wikijs-log-group"
  }
}########################################
# Task Definition
########################################
resource "aws_ecs_task_definition" "wikijs" {
  family                   = "wikijs-task"
  network_mode             = "awsvpc"
  requires_compatibilities = ["FARGATE"]
  cpu                      = "256" # .25 vCPU
  memory                   = "512" # 0.5 GB
  execution_role_arn       = aws_iam_role.ecs_task_execution_role.arn
  #task_role_arn            = aws_iam_role.ecs_task_role.arn

  container_definitions = jsonencode([
    {
      name       = "wikijs"
      image      = "643218715566.dkr.ecr.eu-west-1.amazonaws.com/wiki:2.5.312"
      essential  = true
      entryPoint = ["sh", "-c", "printenv && node server"]
      secrets = [
        {
          name      = "DB_PASS"
          # Extract the 'password' key from the RDS-generated secret
          valueFrom = "${aws_db_instance.wiki.master_user_secret[0].secret_arn}:pass
word::"
        }
      ]
      environmentFiles = [
        {
          value = "arn:aws:s3:::wikijs-conf/wikijs.env"
          type  = "s3"
        }
      ]
      environment = [
        {
          name  = "DB_HOST"
          # Reference the RDS instance address attribute
          value = aws_db_instance.wiki.address
        }
      ]
      portMappings = [
        {
          containerPort = 3000
          hostPort      = 3000
          protocol      = "tcp"
        }
      ]
      logConfiguration = {
        logDriver = "awslogs"
        options = {
          "awslogs-group"         = "/ecs/wikijs"
          "awslogs-region"        = "eu-west-1"
          "awslogs-stream-prefix" = "wikijs"
        }
      }
    }
  ])
}
########################################
# ECS Cluster (Container Insights Disabled)
########################################
resource "aws_ecs_cluster" "wikijs" {
  name = "wikijs-cluster"

  tags = {
    Name        = "wikijs-cluster"
  }
}
########################################
# ECS Service
########################################
resource "aws_ecs_service" "wikijs" {
  name                   = "wikijs-service"
  cluster                = aws_ecs_cluster.wikijs.name
  task_definition        = aws_ecs_task_definition.wikijs.arn
  desired_count          = 1
  launch_type            = "FARGATE"
  #enable_execute_command = true

  network_configuration {
    subnets          = module.vpc.private_subnets
    security_groups  = [aws_security_group.ecs_sg.id]
    assign_public_ip = false
  }

  load_balancer {
    target_group_arn = aws_lb_target_group.wikijs.arn
    container_name   = "wikijs"
    container_port   = 3000
  }

  deployment_minimum_healthy_percent = 50
  deployment_maximum_percent         = 200

  depends_on = [
    aws_lb_listener.http
  ]
  tags = merge(local.common_tags, {
    Name = "wikijs-service"
  })
}
############################
# Autoscaling Target
############################

resource "aws_appautoscaling_target" "ecs" {
  max_capacity       = var.ecs_max_capacity
  min_capacity       = var.ecs_min_capacity
  resource_id        = "service/${aws_ecs_cluster.wikijs.name}/${aws_ecs_service.wik
ijs.name}"
  scalable_dimension = "ecs:service:DesiredCount"
  service_namespace  = "ecs"
}

############################
# CPU Target Tracking (70%)
############################

resource "aws_appautoscaling_policy" "ecs_cpu" {
  name               = "ecs-cpu-scaling"
  policy_type        = "TargetTrackingScaling"
  resource_id        = aws_appautoscaling_target.ecs.resource_id
  scalable_dimension = aws_appautoscaling_target.ecs.scalable_dimension
  service_namespace  = aws_appautoscaling_target.ecs.service_namespace

  target_tracking_scaling_policy_configuration {
    target_value       = 70.0
    scale_in_cooldown  = 60
    scale_out_cooldown = 60

    predefined_metric_specification {
      predefined_metric_type = "ECSServiceAverageCPUUtilization"
    }
  }
}########################################
# IAM Role - Task Execution Role
########################################

resource "aws_iam_role" "ecs_task_execution_role" {
  name = "wikijs-ecs-task-execution-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Effect = "Allow"
      Principal = {
        Service = "ecs-tasks.amazonaws.com"
      }
      Action = "sts:AssumeRole"
    }]
  })

  tags = {
    Name = "wikijs-ecs-task-execution-role"
  }
}

# AWS managed policy for image pull and CloudWatch logs
resource "aws_iam_role_policy_attachment" "ecs_execution_policy" {
  role       = aws_iam_role.ecs_task_execution_role.name
  policy_arn = "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolic
y"
}

# Inline policy to read configuration from S3
resource "aws_iam_role_policy" "s3_read_bucket" {
  name = "wikijsS3ReadBucket"
  role = aws_iam_role.ecs_task_execution_role.id

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Sid    = "AllowBucketDiscovery"
        Effect = "Allow"
        Action = ["s3:GetBucketLocation", "s3:ListBucket"]
        Resource = ["arn:aws:s3:::wikijs-conf"]
      },
      {
        Sid    = "AllowReadConfigObjects"
        Effect = "Allow"
        Action = ["s3:GetObject"]
        Resource = ["arn:aws:s3:::wikijs-conf/*"]
      }
    ]
  })
}

# Inline policy to allow ECS to fetch the password from Secret Manager created by RD
S
resource "aws_iam_role_policy" "ecs_rds_secret_access" {
  role = aws_iam_role.ecs_task_execution_role.id
  name = "wikijsGetSecretValue"
  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Effect   = "Allow"
      Action   = ["secretsmanager:GetSecretValue"]
      Resource = [aws_db_instance.wiki.master_user_secret[0].secret_arn]
    }]
  })
}
####################################
# RDS Enhanced Monitoring IAM Role
####################################

resource "aws_iam_role" "rds_monitoring" {
  name = "rds-enhanced-monitoring-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Principal = {
          Service = "monitoring.rds.amazonaws.com"
        }
        Action = "sts:AssumeRole"
      }
    ]
  })
}

resource "aws_iam_role_policy_attachment" "rds_monitoring_attach" {
  role       = aws_iam_role.rds_monitoring.name
  policy_arn = "arn:aws:iam::aws:policy/service-role/AmazonRDSEnhancedMonitoringRole
"
}locals {
  common_tags = {
    Project     = "WikiJS"
    Environment = "Assessment"
    ManagedBy   = "Terraform"
  }
}##########################
# ALB Outputs
##########################
output "alb_dns_name" {
  value = aws_lb.wikijs.dns_name
}

output "alb_region" {
  value = aws_lb.wikijs.region
}
##########################
# Database Outputs
##########################
output "db_endpoint" {
  value = aws_db_instance.wiki.endpoint
}

output "db_availability_zone" {
  value = aws_db_instance.wiki.availability_zone
}

################################
# ECS Cluster & Service Outputs
################################
output "ecs_cluster_name" {
  value = aws_ecs_cluster.wikijs.name
}

output "ecs_service_name" {
  value = aws_ecs_service.wikijs.name
}

output "ecs_service_desired_count" {
  value = aws_ecs_service.wikijs.desired_count
}
##########################
# Autoscaling Outputs
##########################
output "ecs_autoscaling_min_capacity" {
  value = aws_appautoscaling_target.ecs.min_capacity
}

output "ecs_autoscaling_max_capacity" {
  value = aws_appautoscaling_target.ecs.max_capacity
}

output "ecs_autoscaling_resource_id" {
  value = aws_appautoscaling_target.ecs.resource_id
}

output "ecs_cpu_scaling_policy_name" {
  value = aws_appautoscaling_policy.ecs_cpu.name
}

output "ecs_cpu_scaling_target_value" {
  value = aws_appautoscaling_policy.ecs_cpu.target_tracking_scaling_policy_configura
tion[0].target_value
}

##########################
# VPC Endpoint Outputs
##########################
output "vpce_s3_state" {
  value = aws_vpc_endpoint.s3.state
}

output "vpce_ecr_api_state" {
  value = aws_vpc_endpoint.ecr_api.state
}

output "vpce_ecr_dkr_state" {
  value = aws_vpc_endpoint.ecr_dkr.state
}

output "vpce_logs_state" {
  value = aws_vpc_endpoint.logs.state
}

output "vpce_secretsmanager_state" {
  value = aws_vpc_endpoint.secretsmanager.state
}
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 6.0"
    }
  }
}
```

```
provider "aws" {
  region = "eu-west-1"

  default_tags {
    tags = {
      Project     = "WikiJS"
      Environment = "Assessment"
      ManagedBy   = "Terraform"
    }
  }
}
resource "aws_db_subnet_group" "wiki" {
  name       = "wikijs-db-subnet-group"
  subnet_ids = module.vpc.private_subnets

  tags = merge(local.common_tags, {
    Name = "wikijs-db-subnet-group"
  })
}

resource "aws_db_parameter_group" "wiki" {
  name   = "wikijs-postgres17"
  family = "postgres17"

  parameter {
    name  = "rds.force_ssl"
    value = "0"
  }

  tags = merge(local.common_tags, {
    Name = "wikijs-db-parameter-group"
  })
}
########################################
# DB Instance
########################################
resource "aws_db_instance" "wiki" {
  identifier = "wikijs-db"

  engine         = "postgres"
  engine_version = "17.6"
  instance_class = "db.t4g.micro"

  allocated_storage     = 20
  max_allocated_storage = 100
  storage_type          = "gp3"

  db_name  = "wikijs"
  username = var.db_username
  #Manage the master password with Secrets Manager.
  manage_master_user_password = true

  port                 = 5432
  parameter_group_name = aws_db_parameter_group.wiki.name

  db_subnet_group_name   = aws_db_subnet_group.wiki.name
  vpc_security_group_ids = [aws_security_group.db_sg.id]

  publicly_accessible = false
  multi_az            = true

  storage_encrypted = true

  backup_retention_period   = 7
  delete_automated_backups = false
  skip_final_snapshot       = false
  final_snapshot_identifier = "wikijs-final-snapshot-${formatdate("YYYYMMDDhhmmss",
timestamp())}"
  deletion_protection       = true

  auto_minor_version_upgrade = true
  apply_immediately           = true

  performance_insights_enabled = true
  performance_insights_retention_period = 7
  monitoring_interval          = 60
  monitoring_role_arn = aws_iam_role.rds_monitoring.arn

  tags = merge(local.common_tags, {
    Name = "wikijs-db"
  })
}
########################################
# ALB Security Group
########################################
resource "aws_security_group" "alb_sg" {
  name        = "wikijs-alb-sg"
  description = "Application Load Balancer"
  vpc_id      = module.vpc.vpc_id

  tags = merge(local.common_tags, {
    Name = "wikijs-alb-sg"
  })

}

# Ingress (from Internet)
resource "aws_vpc_security_group_ingress_rule" "alb_http" {
  security_group_id = aws_security_group.alb_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "tcp"
  from_port         = 80
  to_port           = 80
  description       = "HTTP from Internet"
}

resource "aws_vpc_security_group_ingress_rule" "alb_https" {
  security_group_id = aws_security_group.alb_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "tcp"
  from_port         = 443
  to_port           = 443
  description       = "HTTPS from Internet"
}

# Egress
resource "aws_vpc_security_group_egress_rule" "alb_to_ecs" {
  security_group_id            = aws_security_group.alb_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 3000
  to_port                      = 3000
  referenced_security_group_id = aws_security_group.ecs_sg.id
  description                  = "ALB can reach ECS tasks"
}

########################################
# ECS Security Group
########################################
resource "aws_security_group" "ecs_sg" {
  name        = "wikijs-ecs-sg"
  description = "ECS tasks"
  vpc_id      = module.vpc.vpc_id

  tags = merge(local.common_tags, {
    Name = "wikijs-ecs-sg"
  })
}

# Ingress (from ALB)
resource "aws_vpc_security_group_ingress_rule" "ecs_from_alb" {
  security_group_id            = aws_security_group.ecs_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 3000
  to_port                      = 3000
  referenced_security_group_id = aws_security_group.alb_sg.id
  description                  = "Allow ALB to reach ECS tasks"
}

# Egress (to DB)
```

```
resource "aws_vpc_security_group_egress_rule" "ecs_to_db" {
  security_group_id            = aws_security_group.ecs_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 5432
  to_port                      = 5432
  referenced_security_group_id = aws_security_group.db_sg.id
  description                  = "ECS tasks can reach RDS"
}

# Get AWS managed S3 prefix list
data "aws_prefix_list" "s3" {
  name = "com.amazonaws.${var.region}.s3"
}

# ECS egress to S3 over HTTPS
resource "aws_vpc_security_group_egress_rule" "ecs_to_s3" {
  security_group_id = aws_security_group.ecs_sg.id
  ip_protocol       = "tcp"
  from_port         = 443
  to_port           = 443
  prefix_list_id    = data.aws_prefix_list.s3.id
  description       = "ECS outbound to Gateway VPC Endpoint (S3)"
}

# Egress (to vpc endpoints)
resource "aws_vpc_security_group_egress_rule" "ecs_to_vpce" {
  security_group_id            = aws_security_group.ecs_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 443
  to_port                      = 443
  referenced_security_group_id = aws_security_group.vpce.id
  description                  = "ECS outbound to Interface VPC Endpoints"
}

########################################
# Interface VPC Endpoints Security Group
########################################
resource "aws_security_group" "vpce" {
  name        = "wikijs-vpce-sg"
  description = "Allow ECS to access VPC interface endpoints"
  vpc_id      = module.vpc.vpc_id

  tags = merge(local.common_tags, {
    Name = "wikijs-vpce-sg"
  })
}

# Ingress (from ecs)
resource "aws_vpc_security_group_ingress_rule" "vpce_from_ecs" {
  security_group_id            = aws_security_group.vpce.id
  ip_protocol                  = "tcp"
  from_port                    = 443
  to_port                      = 443
  referenced_security_group_id = aws_security_group.ecs_sg.id
  description                  = "Allow ECS tasks to access interface endpoints"
}

########################################
# RDS Security Group
########################################
resource "aws_security_group" "db_sg" {
  name        = "wikijs-db-sg"
  description = "Allow PostgreSQL access from ECS only"
  vpc_id      = module.vpc.vpc_id

  tags = merge(local.common_tags, {
    Name = "wikijs-db-sg"
  })
}

# Ingress (from ECS)
resource "aws_vpc_security_group_ingress_rule" "db_from_ecs" {
  security_group_id            = aws_security_group.db_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 5432
  to_port                      = 5432
  referenced_security_group_id = aws_security_group.ecs_sg.id
  description                  = "Allow ECS tasks to connect"
}
variable "region" {
  type        = string
  description = "Default region"
  default     = "eu-west-1"
}

variable "db_username" {
  type        = string
  description = "Database admin username"
  default     = "postgres"
}

variable "ecs_min_capacity" {
type = number
default = 1
}

variable "ecs_max_capacity" {
type = number
default = 3
}##########################
# Interface VPC Endpoints
##########################
resource "aws_vpc_endpoint" "ecr_api" {
  vpc_id              = module.vpc.vpc_id
  service_name        = "com.amazonaws.${var.region}.ecr.api"
  vpc_endpoint_type   = "Interface"
  subnet_ids          = module.vpc.private_subnets
  security_group_ids  = [aws_security_group.vpce.id]
  private_dns_enabled = true

  tags = merge(local.common_tags, {
    Name = "wikijs-ecr-api-endpoint"
  })
}

resource "aws_vpc_endpoint" "ecr_dkr" {
  vpc_id              = module.vpc.vpc_id
  service_name        = "com.amazonaws.${var.region}.ecr.dkr"
  vpc_endpoint_type   = "Interface"
  subnet_ids          = module.vpc.private_subnets
  security_group_ids  = [aws_security_group.vpce.id]
  private_dns_enabled = true

  tags = merge(local.common_tags, {
    Name = "wikijs-ecr-dkr-endpoint"
  })
}

resource "aws_vpc_endpoint" "logs" {
  vpc_id              = module.vpc.vpc_id
  service_name        = "com.amazonaws.${var.region}.logs"
  vpc_endpoint_type   = "Interface"
  subnet_ids          = module.vpc.private_subnets
  security_group_ids  = [aws_security_group.vpce.id]
  private_dns_enabled = true

  tags = merge(local.common_tags, {
    Name = "wikijs-logs-endpoint"
  })
}

resource "aws_vpc_endpoint" "secretsmanager" {
  vpc_id              = module.vpc.vpc_id
  service_name        = "com.amazonaws.${var.region}.secretsmanager"
  vpc_endpoint_type   = "Interface"
  subnet_ids          = module.vpc.private_subnets
  security_group_ids  = [aws_security_group.vpce.id]
  private_dns_enabled = true

  tags = { Name = "wikijs-secretsmanager-endpoint" }
```

```
}

#########################
# Gateway VPC Endpoint
#########################

resource "aws_vpc_endpoint" "s3" {
  vpc_id            = module.vpc.vpc_id
  service_name      = "com.amazonaws.${var.region}.s3"
  vpc_endpoint_type = "Gateway"
  route_table_ids   = module.vpc.private_route_table_ids
  tags = merge(local.common_tags, {
    Name = "wikijs-s3-endpoint"
  })
}
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "6.6.0"

  name = "wikijs-vpc"
  cidr = "10.0.0.0/16"

  azs = [
    "eu-west-1a",
    "eu-west-1b",
    #"eu-west-1c",
  ]

  # Public subnets
  public_subnets = [
    "10.0.1.0/24",
    "10.0.2.0/24",
    #"10.0.3.0/24",
  ]

  # Private subnets
  private_subnets = [
    "10.0.11.0/24",
    "10.0.12.0/24",
    #"10.0.13.0/24",
  ]

  create_igw         = true
  enable_nat_gateway = false
  #single_nat_gateway = true

  enable_dns_support   = true
  enable_dns_hostnames = true

  # Public subnet tags
  public_subnet_tags = {
    Name = "wikijs-public-subnet"
  }

  # Private subnet tags
  private_subnet_tags = {
    Name = "wikijs-private-subnet"
  }

  # Route table tags
  public_route_table_tags = {
    Name = "wikijs-public-rt"
  }

  private_route_table_tags = {
    Name = "wikijs-private-rt"
  }

  default_route_table_tags = {
    Name = "wikijs-default-rt"
  }

  #  # NAT Gateway tag
  #  nat_gateway_tags = {
  #    Name = "wikijs-nat-gw"
  #   }
  #
  # Internet Gateway tag
  igw_tags = {
    Name = "wikijs-igw"
  }

  # General VPC tags
  tags = merge(local.common_tags, {
    Name = "wikijs-vpc"
  })
}
```