```
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"
    Project     = "WikiJS"
    Environment = "Assessment"
  }
}########################################
# 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       = "arn:aws:iam::643218715566:role/WikijsTaskExecutionRole"
  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
      portMappings = [
        {
          containerPort = 3000
          hostPort      = 3000
          protocol      = "tcp"
        }
      ]
      environment = [
        {
          name  = "WIKIJS_ENV_FILE"
          value = "s3://wikijs-conf/wikijs.env"
        }
      ]
      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"
    Project     = "WikiJS"
    Environment = "Assessment"
  }
}
########################################
# 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"

  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"
  })
}#####################################
# 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"
    Project     = "WikiJS"
    Environment = "Assessment"
  }
}

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/AmazonECSTaskExecutionRolePolicy"
}

#####################################
# ECS Task Role for S3 access
#####################################

resource "aws_iam_role" "ecs_task_role" {
  name = "wikijs-ecs-task-role"

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

  tags = {
    Name        = "wikijs-ecs-task-role"
    Project     = "WikiJS"
    Environment = "Assessment"
    ManagedBy   = "Terraform"
  }
}
```

```
# Policy to allow ECS tasks to read the S3 .env file
resource "aws_iam_role_policy" "ecs_s3_access" {
  name = "ecs-s3-read-wikijs-env"
  role = aws_iam_role.ecs_task_role.id

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Effect   = "Allow",
        Action   = ["s3:GetObject"],
        Resource = "arn:aws:s3:::wikijs-conf/wikijs.env"
      }
    ]
  })
}locals {
  common_tags = {
    Project     = "WikiJS"
    Environment = "Assessment"
    ManagedBy  = "Terraform"
  }
}terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 6.0"
    }
  }
}

provider "aws" {
  region = "eu-west-1"
}
#######################################
# 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"
}

#resource "aws_vpc_security_group_egress_rule" "alb_egress_general" {
#  security_group_id = aws_security_group.alb_sg.id
#  ip_protocol       = "tcp"
#  from_port         = 0
#  to_port           = 0
#  cidr_ipv4         = "0.0.0.0/0"
#  description       = "ALB general outbound"
#}

#######################################
# 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 + general)
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"
}

# 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
  #cidr_ipv4         = "0.0.0.0/0"
  description       = "ECS general outbound traffic"
}

#######################################
# RDS Security Group
#######################################
```

```
resource "aws_security_group" "db_sg" {
  name        = "wikijs-db-sg"
  description = "RDS PostgreSQL"
  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"
}

# Egress (general)
resource "aws_vpc_security_group_egress_rule" "db_egress" {
  security_group_id = aws_security_group.db_sg.id
  ip_protocol       = "tcp"
  from_port         = 0
  to_port           = 0
  referenced_security_group_id = aws_security_group.ecs_sg.id
  #cidr_ipv4         = "0.0.0.0/0"
  description       = "RDS outbound traffic"
}
variable "region" {
  type        = string
  description = "Default region"
  default = "eu-west-1"
}

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

variable "db_password" {
  type        = string
  description = "Database admin password"
  sensitive   = true
  default = ""
}#######################################
# 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
  #referenced_security_group_id = aws_security_group.ecs_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 443
```

```
  to_port                      = 443
  cidr_ipv4         = "10.0.0.0/16"
  description                  = "Allow ECS tasks to access interface endpoints"
}

# Egress (to ecs)
resource "aws_vpc_security_group_egress_rule" "vpce_egress" {
  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
  cidr_ipv4         = "10.0.0.0/16"
  description       = "Endpoint outbound"
}

#########################
# 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"
  })
}

########################
# 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"
  })
}
```