Terraform - ECS Fargate + Aurora Serverless + EFS

```
# Terraform - ECS Fargate + Aurora Serverless + EFS
# Source: github.com/futurice/terraform-examples
#         aws/wordpress_fargate

provider "aws" {
  region = "ap-northeast-2"
}

locals {
  name = "webapp"
  env  = "production"
  tags = {
    Environment = local.env
    ManagedBy   = "terraform"
    Project     = "webapp-fargate"
  }
}

data "aws_availability_zones" "az" {
  state = "available"
}

# ========== VPC & Networking ==========

resource "aws_vpc" "main" {
  cidr_block           = "10.0.0.0/16"
  enable_dns_hostnames = true
  enable_dns_support   = true
  tags = merge(local.tags, { Name = "${local.name}-vpc" })
}

resource "aws_subnet" "public" {
  count             = 2
  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet("10.0.0.0/16", 8, count.index)
  availability_zone = data.aws_availability_zones.az.names[count.index]
  map_public_ip_on_launch = true
  tags = merge(local.tags, {
    Name = "${local.name}-pub-${count.index}"
  })
}

resource "aws_subnet" "private" {
  count             = 2
  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet("10.0.0.0/16", 8, count.index + 10)
  availability_zone = data.aws_availability_zones.az.names[count.index]
  tags = merge(local.tags, {
    Name = "${local.name}-priv-${count.index}"
  })
}

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
  tags   = merge(local.tags, { Name = "${local.name}-igw" })
}

resource "aws_eip" "nat" {
  domain = "vpc"
  tags   = merge(local.tags, { Name = "${local.name}-eip" })
}

resource "aws_nat_gateway" "main" {
```

```
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.public[0].id
  tags = merge(local.tags, { Name = "${local.name}-nat" })
}

# ========== Security Groups ==========

resource "aws_security_group" "alb" {
  name   = "${local.name}-alb-sg"
  vpc_id = aws_vpc.main.id
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = local.tags
}

resource "aws_security_group" "ecs" {
  name   = "${local.name}-ecs-sg"
  vpc_id = aws_vpc.main.id
  ingress {
    from_port       = 8080
    to_port         = 8080
    protocol        = "tcp"
    security_groups = [aws_security_group.alb.id]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = local.tags
}

resource "aws_security_group" "db" {
  name   = "${local.name}-db-sg"
  vpc_id = aws_vpc.main.id
  ingress {
    from_port       = 3306
    to_port         = 3306
    protocol        = "tcp"
    security_groups = [aws_security_group.ecs.id]
  }
  tags = local.tags
}

resource "aws_security_group" "efs" {
  name   = "${local.name}-efs-sg"
  vpc_id = aws_vpc.main.id
  ingress {
    from_port       = 2049
```

```
    to_port         = 2049
    protocol        = "tcp"
    security_groups = [aws_security_group.ecs.id]
  }
  tags = local.tags
}

# ========== EFS ==========

resource "aws_efs_file_system" "app" {
  creation_token = "${local.name}-efs"
  encrypted      = true
  lifecycle_policy {
    transition_to_ia = "AFTER_30_DAYS"
  }
  tags = merge(local.tags, { Name = "${local.name}-efs" })
}

resource "aws_efs_mount_target" "app" {
  count           = 2
  file_system_id  = aws_efs_file_system.app.id
  subnet_id       = aws_subnet.private[count.index].id
  security_groups = [aws_security_group.efs.id]
}

# ========== Aurora Serverless ==========

resource "aws_db_subnet_group" "main" {
  name       = "${local.name}-db-subnet"
  subnet_ids = aws_subnet.private[*].id
  tags       = local.tags
}

resource "aws_rds_cluster" "main" {
  cluster_identifier      = "${local.name}-aurora"
  engine                  = "aurora-mysql"
  engine_mode             = "serverless"
  database_name           = "webapp"
  master_username         = var.db_username
  master_password         = var.db_password
  db_subnet_group_name    = aws_db_subnet_group.main.name
  vpc_security_group_ids  = [aws_security_group.db.id]
  skip_final_snapshot     = false
  backup_retention_period = 7
  deletion_protection     = true

  scaling_configuration {
    auto_pause               = true
    min_capacity             = 1
    max_capacity             = 4
    seconds_until_auto_pause = 300
  }
  tags = local.tags
}

# ========== ECS Fargate ==========

resource "aws_ecs_cluster" "main" {
  name = "${local.name}-cluster"
  setting {
    name  = "containerInsights"
    value = "enabled"
  }
  tags = local.tags
}
```

```
resource "aws_iam_role" "ecs_exec" {
  name = "${local.name}-ecs-exec"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Action    = "sts:AssumeRole"
      Effect    = "Allow"
      Principal = {
        Service = "ecs-tasks.amazonaws.com"
      }
    }]
  })
  tags = local.tags
}

resource "aws_iam_role_policy_attachment" "ecs_exec" {
  role       = aws_iam_role.ecs_exec.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonECSTaskExecRolePolicy"
}

resource "aws_cloudwatch_log_group" "app" {
  name              = "/ecs/${local.name}"
  retention_in_days = 30
  tags              = local.tags
}

resource "aws_ecs_task_definition" "app" {
  family                   = "${local.name}-app"
  network_mode             = "awsvpc"
  requires_compatibilities = ["FARGATE"]
  cpu                      = "512"
  memory                   = "1024"
  execution_role_arn       = aws_iam_role.ecs_exec.arn

  container_definitions = jsonencode([{
    name  = "app"
    image = "wordpress:6.4-php8.2-fpm"
    portMappings = [{ containerPort = 8080 }]
    environment = [
      { name = "DB_HOST",
        value = aws_rds_cluster.main.endpoint },
      { name = "DB_NAME", value = "webapp" },
    ]
    secrets = [
      { name      = "DB_USER"
        valueFrom = aws_ssm_parameter.db_user.arn },
      { name      = "DB_PASS"
        valueFrom = aws_ssm_parameter.db_pass.arn },
    ]
    mountPoints = [{
      sourceVolume  = "efs-vol"
      containerPath = "/var/www/html"
    }]
    logConfiguration = {
      logDriver = "awslogs"
      options = {
        "awslogs-group"  = "/ecs/${local.name}"
        "awslogs-region" = "ap-northeast-2"
        "awslogs-stream-prefix" = "ecs"
      }
    }
    healthCheck = {
      command  = ["CMD-SHELL",
        "curl -f http://localhost:8080/ || exit 1"]
      interval = 30
      timeout  = 5
```

```
      retries  = 3
    }
  }])

  volume {
    name = "efs-vol"
    efs_volume_configuration {
      file_system_id = aws_efs_file_system.app.id
    }
  }
  tags = local.tags
}

resource "aws_ecs_service" "app" {
  name            = "${local.name}-svc"
  cluster         = aws_ecs_cluster.main.id
  task_definition = aws_ecs_task_definition.app.arn
  desired_count   = 2
  launch_type     = "FARGATE"

  network_configuration {
    subnets          = aws_subnet.private[*].id
    security_groups  = [aws_security_group.ecs.id]
    assign_public_ip = false
  }

  load_balancer {
    target_group_arn = aws_lb_target_group.app.arn
    container_name   = "app"
    container_port   = 8080
  }

  deployment_circuit_breaker {
    enable   = true
    rollback = true
  }
  tags = local.tags
}

# ========== Auto Scaling ==========

resource "aws_appautoscaling_target" "ecs" {
  max_capacity       = 6
  min_capacity       = 2
  resource_id        = "service/${aws_ecs_cluster.main.name}/${aws_ecs_service.app.name}"
  scalable_dimension = "ecs:service:DesiredCount"
  service_namespace  = "ecs"
}

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

  target_tracking_scaling_policy_configuration {
    predefined_metric_specification {
      predefined_metric_type = "ECSServiceAverageCPUUtilization"
    }
    target_value       = 70
    scale_in_cooldown  = 300
    scale_out_cooldown = 60
  }
}
```

```
# ========== Secrets (SSM) ==========

resource "aws_ssm_parameter" "db_user" {
  name  = "/${local.name}/db/username"
  type  = "SecureString"
  value = var.db_username
  tags  = local.tags
}

resource "aws_ssm_parameter" "db_pass" {
  name  = "/${local.name}/db/password"
  type  = "SecureString"
  value = var.db_password
  tags  = local.tags
}

# ========== Monitoring ==========

resource "aws_cloudwatch_metric_alarm" "cpu_high" {
  alarm_name          = "${local.name}-cpu-high"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/ECS"
  period              = 300
  statistic           = "Average"
  threshold           = 85
  alarm_description   = "ECS CPU > 85%"
  dimensions = {
    ClusterName = aws_ecs_cluster.main.name
    ServiceName = aws_ecs_service.app.name
  }
}

variable "db_username" {
  type      = string
  sensitive = true
}

variable "db_password" {
  type      = string
  sensitive = true
}

output "cluster_name" {
  value = aws_ecs_cluster.main.name
}

output "rds_endpoint" {
  value     = aws_rds_cluster.main.endpoint
  sensitive = true
}

output "efs_id" {
  value = aws_efs_file_system.app.id
}
```