main.tf

```
provider "aws" {
 region = var.aws_region
}

# Recurso da instância EC2
resource "aws_instance" "ec2_instance" {
 ami         = aplicação
 instance_type = var.instance_type
 tags = {
  Name = "sciserviços"
 }
}


resource "aws_security_group" "instance_security_group" {
 name       = "instance-security-group"
 description = "grupo de acesso a porta 80"

 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"]
 }
}

# Recurso dos containers
resource "null_resource" "docker_deploy" {
 depends_on = [aws_instance.ec2_instance]

 provisioner "local-exec" {
  command = <<EOF
   ssh -o StrictHostKeyChecking=no ec2-user@${aws_instance.ec2_instance.public_ip} 'docker run -d -p 80:80 your-php-app'
  EOF
 }
}


resource "aws_lb" "load_balancer" {
 name            = "Load-Balancer"
 internal        = false
 load_balancer_type = "application"
 security_groups    = [aws_security_group.instance_security_group.id]
 subnets         = var.subnets

 tags = {
```

```
  Name = "Load-Balancer SCI"
 }
}


resource "aws_lb_target_group" "target_group" {
 name     = "target-loadbalance"
 port     = 80
 protocol = "HTTP"
 vpc_id   = var.vpc_id

 health_check {
  path                = "/"
  protocol            = "HTTP"
  matcher             = "200"
  timeout             = 5
  interval            = 30
  healthy_threshold   = 3
  unhealthy_threshold = 5
 }

 tags = {
  Name = "target-loadbalance"
 }
}

resource "aws_lb_listener" "http_listener" {
 load_balancer_arn = aws_lb.load_balancer.arn
```

```
  port            = 80
  protocol        = "HTTP"

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



resource "aws_ecs_cluster" "ecs_cluster" {
  name = "Clustersci"
}

resource "aws_ecs_task_definition" "task_definition" {
  family                = "task-sci"
  container_definitions    = file("${path.module}/task_definition.json")
  execution_role_arn      = "default"
  network_mode          = "awsvpc"
  requires_compatibilities = ["EC2"]

  tags = {
    Name = "task-sci"
  }
}



resource "aws_ecs_service" "ecs_service" {
```

```
 name            = "serviçosdegerenciamento"
 cluster         = aws_ecs_cluster.ecs_cluster.id
 task_definition = aws_ecs_task_definition.task_definition.arn
 desired_count   = 1
 launch_type     = "EC2"

 network_configuration {
  subnets         = var.subnets
  security_groups = [aws_security_group.instance_security_group.id]
  assign_public_ip = true
 }

 load_balancer {
  target_group_arn = aws_lb_target_group.target_group.arn
  container_name   = "containerphp"
  container_port   = 80
 }

 tags = {
  Name = "serviçosdegerenciamento"
 }
}
```

##################

variables.tf

```
variable "aws_region" {
  default = "us-east-1"
}

variable "ami_id" {
  description =
  default     =
}

variable "instance_type" {
  default = "t2.micro"
}

variable "vpc_id" {
  description =
  default     =
}

variable "subnets" {
  description =
  type        = list(string)
  default     = ["subnet-123", "subnet-456"]  # Substitua pelos IDs das subnets
}
```

###############

outputs.tf

```
output "ec2_instance_public_ip" {
```

```
  value = aws_instance.ec2_instance.public_ip
}

output "load_balancer_dns" {
  value = aws_lb.load_balancer.dns_sci

################

task_definition.json

{
  "family": "sci",
  "containerDefinitions": [
    {
      "name": "meus_containers",
      "image": "servidor-php",
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "memory": 512,
      "cpu": 256
    }
  ]
```

```
}


}
```