

AWS Three-Tier Architecture Deployment with Terraform

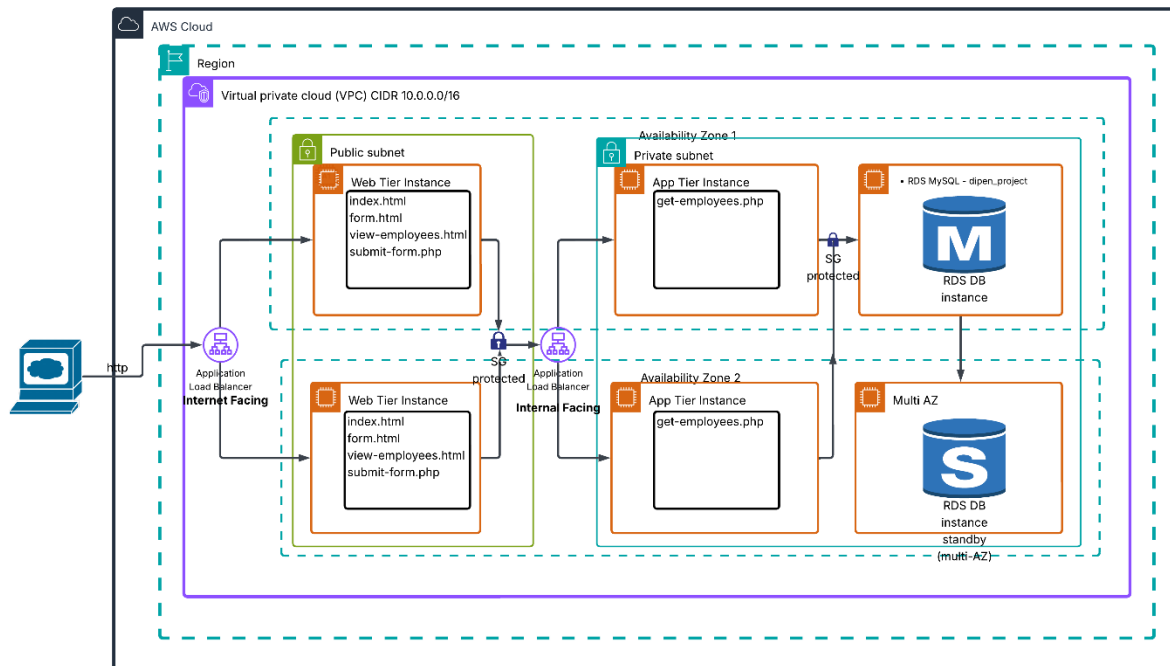
Architecture Layers:

1. **Web Tier** – Public-facing, handles incoming HTTP/HTTPS requests via a **Public ALB**.
2. **App Tier** – Internal application logic, accessible only via an **Internal ALB** from the Web tier.
3. **Database Tier** – Private **RDS MySQL** for persistent storage.

Key Features:

- Infrastructure fully automated with Terraform
- Multi-AZ deployment for HA (High Availability)
- Auto Scaling Groups for web and app tiers
- Secure networking with private subnets & SG rules
- Centralized static file hosting in S3
- Parameter Store for DB credentials & endpoints

Architecture Diagram



Implementation Steps

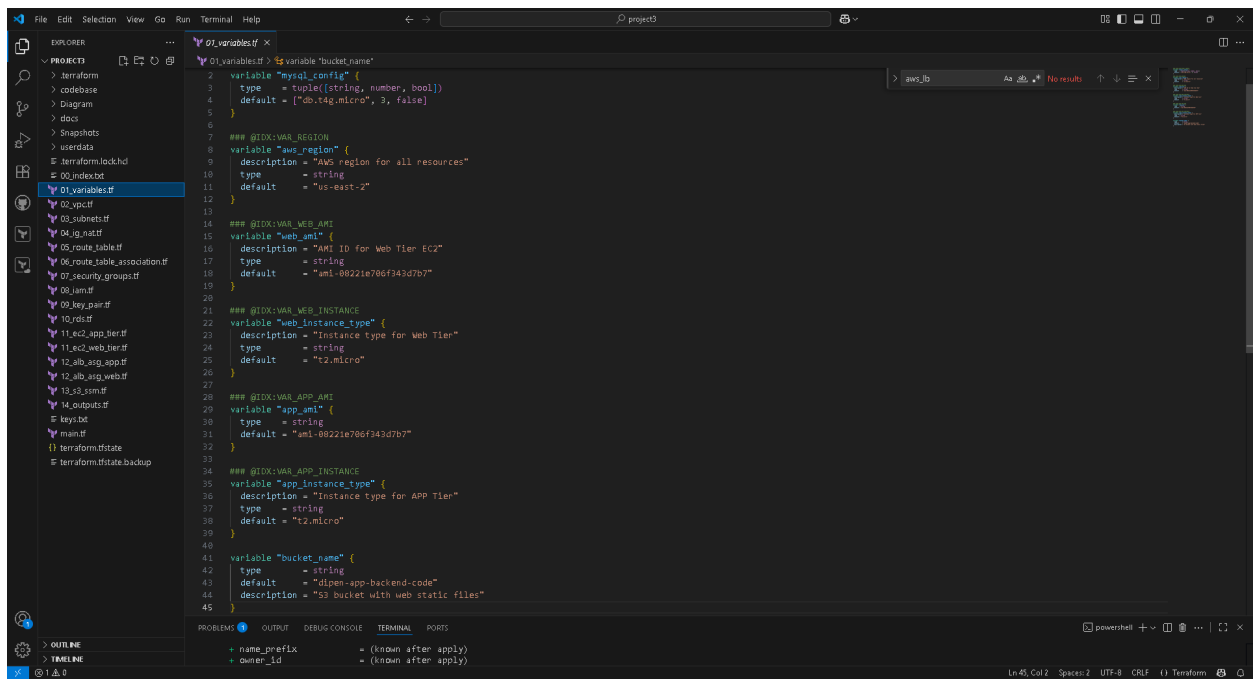
Phase 1: Defining variables and dependencies

```
# C:\Users\dipen\Downloads\project3\main.tf
terraform {
  required_version = ">=1.0.0"

  required_providers {
    # AWS provider by HashiCorp with version 5.0 or higher
    aws = {
      source  = "hashicorp/aws"
      version = ">=5.0"
    }

    # Random provider by HashiCorp with version 1.0 or higher
    random = {
      source  = "hashicorp/random"
      version = ">=1.0"
    }
  }
}

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

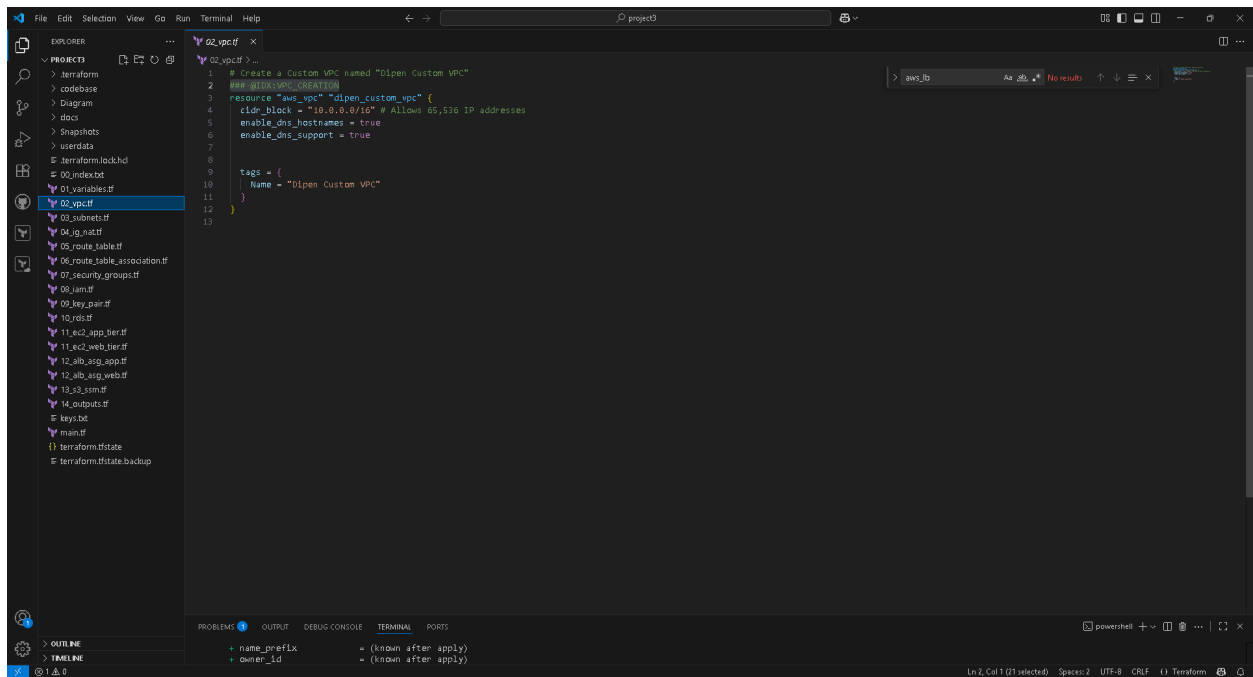


Phase 2: Custom VPC Creation

Provisioned a **custom VPC** with a CIDR block (e.g., 10.0.0.0/16) to host all resources.

The VPC provides:

- Logical network isolation.
- Custom IP addressing.
- Ability to define routing and security at the network level.

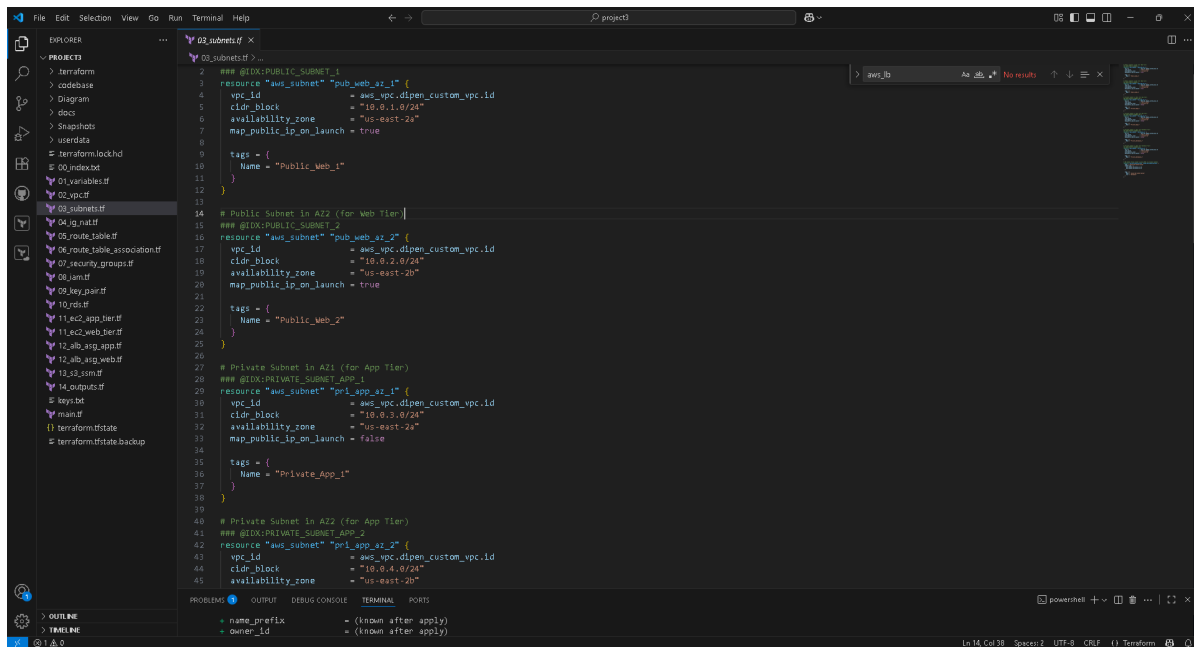


Phase 3: Subnet Creation

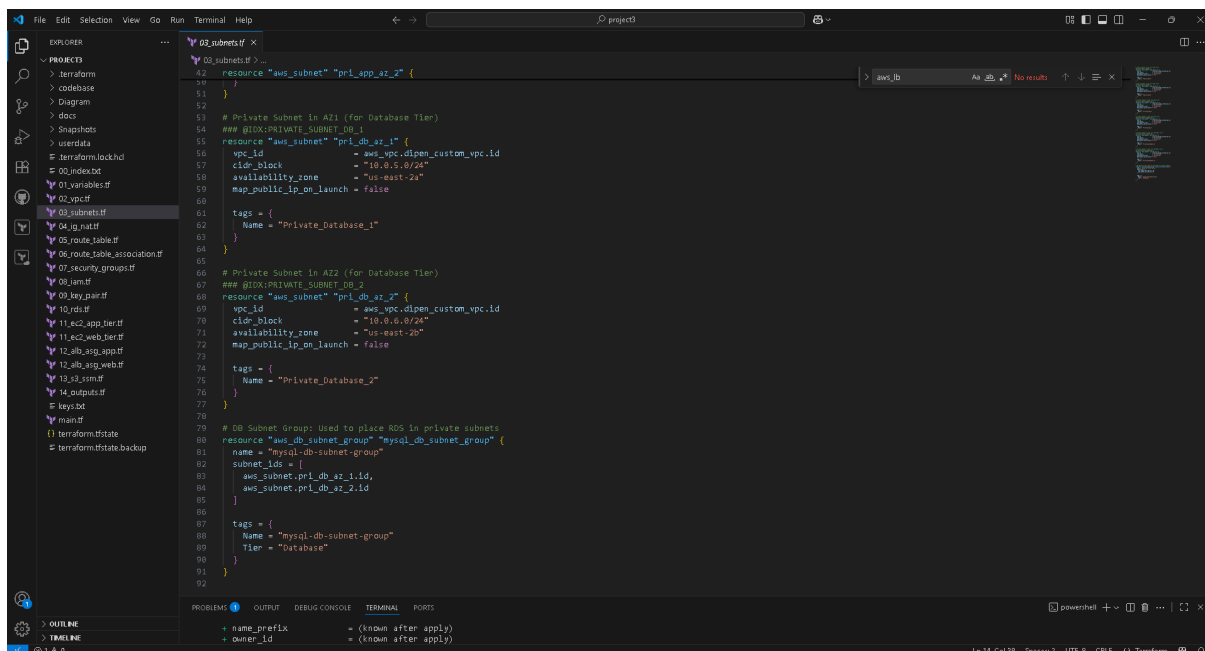
Created six subnets distributed across two Availability Zones:

- **Public Subnets (Web Tier):** Host public-facing ALB and Web EC2 instances.
- **Private Subnets (App Tier):** Host internal ALB and App EC2 instances.
- **Private Subnets (DB Tier):** Host RDS MySQL instance.

Each tier is isolated to enhance security and fault tolerance.



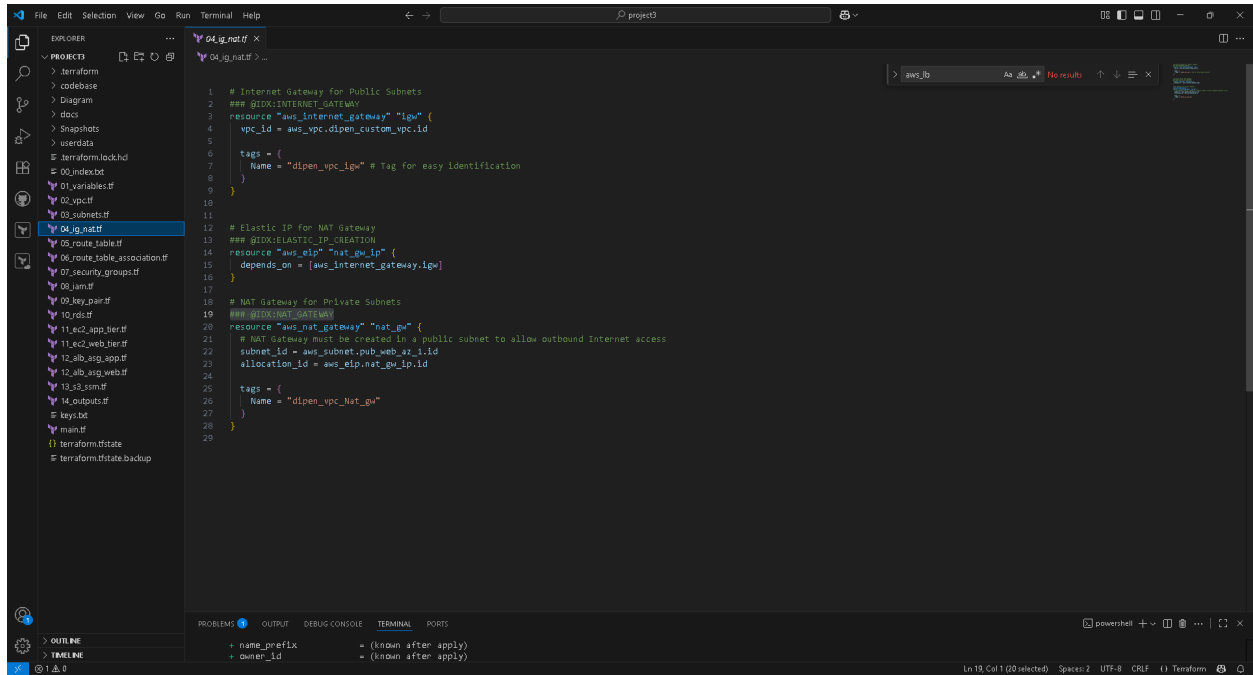
```
1 # Public Subnet in AZ1 (for Web Tier)
2 resource "aws_subnet" "pub_web_az_1" {
3   vpc_id            = aws_vpc.dipen_custom_vpc.id
4   cidr_block        = "10.0.1.0/24"
5   availability_zone  = "us-east-2a"
6   map_public_ip_on_launch = true
7
8   tags = {
9     Name = "Public_Web_1"
10  }
11 }
12
13 # Public Subnet in AZ2 (for Web Tier)
14 resource "aws_subnet" "pub_web_az_2" {
15   vpc_id            = aws_vpc.dipen_custom_vpc.id
16   cidr_block        = "10.0.2.0/24"
17   availability_zone  = "us-east-2b"
18   map_public_ip_on_launch = true
19
20   tags = {
21     Name = "Public_Web_2"
22   }
23 }
24
25 # Private Subnet in AZ1 (for App Tier)
26 resource "aws_subnet" "pri_app_az_1" {
27   vpc_id            = aws_vpc.dipen_custom_vpc.id
28   cidr_block        = "10.0.3.0/24"
29   availability_zone  = "us-east-2a"
30   map_public_ip_on_launch = false
31
32   tags = {
33     Name = "Private_App_1"
34   }
35 }
36
37 # Private Subnet in AZ2 (for App Tier)
38 resource "aws_subnet" "pri_app_az_2" {
39   vpc_id            = aws_vpc.dipen_custom_vpc.id
40   cidr_block        = "10.0.4.0/24"
41   availability_zone  = "us-east-2b"
42   map_public_ip_on_launch = false
43
44   tags = {
45     Name = "Private_App_2"
46   }
47 }
```



```
48 # Private Subnet in AZ1 (for Database Tier)
49 resource "aws_subnet" "pri_db_az_1" {
50   vpc_id            = aws_vpc.dipen_custom_vpc.id
51   cidr_block        = "10.0.5.0/24"
52   availability_zone  = "us-east-2a"
53   map_public_ip_on_launch = false
54
55   tags = {
56     Name = "Private_Database_1"
57   }
58 }
59
60 # Private Subnet in AZ2 (for Database Tier)
61 resource "aws_subnet" "pri_db_az_2" {
62   vpc_id            = aws_vpc.dipen_custom_vpc.id
63   cidr_block        = "10.0.6.0/24"
64   availability_zone  = "us-east-2b"
65   map_public_ip_on_launch = false
66
67   tags = {
68     Name = "Private_Database_2"
69   }
70 }
71
72 # DB Subnet Group: Used to place RDS in private subnets
73 resource "aws_db_subnet_group" "mysql_db_subnet_group" {
74   name = "mysql-db-subnet-group"
75   subnet_ids = [
76     aws_subnet.pri_db_az_1.id,
77     aws_subnet.pri_db_az_2.id
78   ]
79
80   tags = {
81     Name = "mysql-db-subnet-group"
82     Tier = "Database"
83   }
84 }
```

Phase 4: Internet Gateway and NAT Gateway

- **Internet Gateway (IGW):** Allows resources in public subnets to access the internet.
- **NAT Gateway:** Enables private subnet instances (App/DB tiers) to access the internet for updates without being directly exposed.



The screenshot shows a VS Code editor window with a Terraform configuration file named `04_ig.nat.tf`. The file is open in the editor, and the left sidebar shows the project structure. The code defines an Internet Gateway and a NAT Gateway for AWS.

```
1 # Internet Gateway for Public Subnets
2 ## @IX:INTERNET_GATEWAY
3 resource "aws_internet_gateway" "igw" {
4   vpc_id = aws_vpc.dipen_custom_vpc.id
5
6   tags = {
7     Name = "dipen_vpc_igw" # Tag for easy identification
8   }
9 }
10
11
12 # Elastic IP for NAT Gateway
13 ## @IX:ELASTIC_IP_CREATION
14 resource "aws_eip" "nat_gw_ip" {
15   depends_on = [aws_internet_gateway.igw]
16 }
17
18 # NAT Gateway for Private Subnets
19 ## @IX:NAT_GATEWAY
20 resource "aws_nat_gateway" "nat_gw" {
21   # NAT Gateway must be created in a public subnet to allow outbound Internet access
22   subnet_id = aws_subnet.pub_web_v2.id
23   allocation_id = aws_eip.nat_gw_ip.id
24
25   tags = {
26     Name = "dipen_vpc_nat_gw"
27   }
28 }
29
```

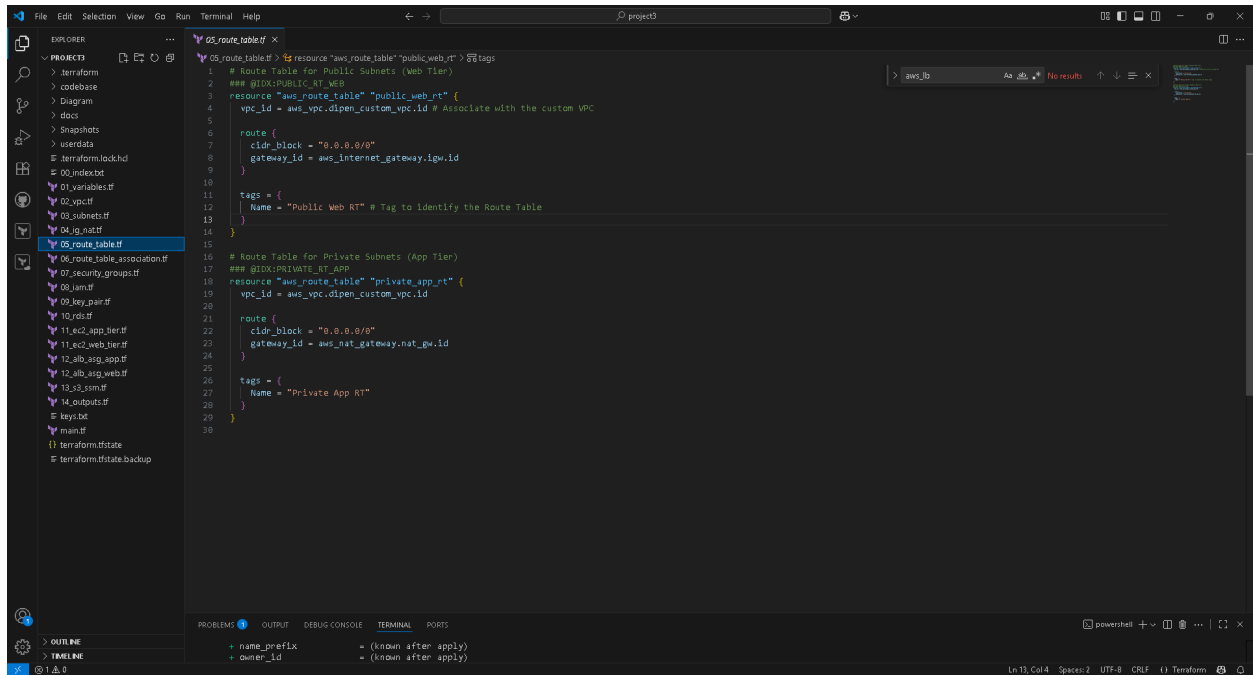
The bottom of the editor shows the `PROBLEMS` panel with two errors related to the `aws_eip` resource:

- `name_prefix` = (known after apply)
- `owner_id` = (known after apply)

Phase 5: Route Table Creation

Public Route Table: Routes outbound internet traffic from public subnets via IGW.

Private Route Tables: Routes outbound traffic from private subnets through NAT Gateway. This ensures correct routing for each tier.

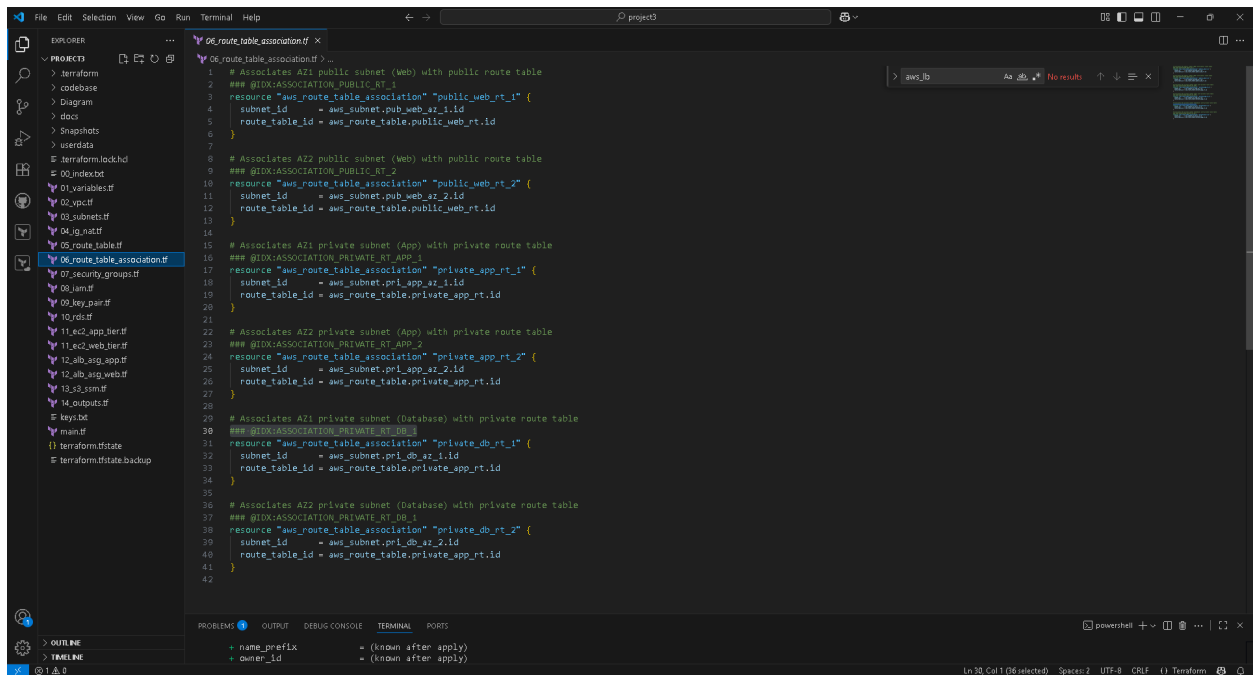


```
1 resource "aws_route_table" "public_web_rt" {
2   # Route Table for Public Subnets (Web Tier)
3   ## @IDX:PUBLIC_RT_WEB
4   resource "aws_route_table" "public_web_rt" {
5     vpc_id = aws_vpc.dipen_custom_vpc.id # Associate with the custom VPC
6
7     route {
8       cidr_block = "0.0.0.0/0"
9       gateway_id = aws_internet_gateway.igw.id
10    }
11
12    tags = {
13      Name = "Public Web RT" # Tag to identify the Route Table
14    }
15  }
16
17  # Route Table for Private Subnets (App Tier)
18  ## @IDX:PRIVATE_RT_APP
19  resource "aws_route_table" "private_app_rt" {
20    vpc_id = aws_vpc.dipen_custom_vpc.id
21
22    route {
23      cidr_block = "0.0.0.0/0"
24      gateway_id = aws_nat_gateway.nat_gw.id
25    }
26
27    tags = {
28      Name = "Private App RT"
29    }
30  }
```

Phase 6: Route Table Association

Linked subnets to their respective route tables:

- Public subnets → Public Route Table.
 - Private subnets (App/DB) → Private Route Tables.
- This enforces intended traffic flow and isolation.



```
1 # Associates AZ1 public subnet (Web) with public route table
2 ### @IDX:ASSOCIATION_PUBLIC_RT_1
3 resource "aws_route_table_association" "public_web_rt_1" {
4   subnet_id      = aws_subnet.pub_web_az_1.id
5   route_table_id = aws_route_table.public_web_rt.id
6 }
7
8 # Associates AZ2 public subnet (Web) with public route table
9 ### @IDX:ASSOCIATION_PUBLIC_RT_2
10 resource "aws_route_table_association" "public_web_rt_2" {
11   subnet_id      = aws_subnet.pub_web_az_2.id
12   route_table_id = aws_route_table.public_web_rt.id
13 }
14
15 # Associates AZ1 private subnet (App) with private route table
16 ### @IDX:ASSOCIATION_PRIVATE_RT_APP_1
17 resource "aws_route_table_association" "private_app_rt_1" {
18   subnet_id      = aws_subnet.pri_app_az_1.id
19   route_table_id = aws_route_table.private_app_rt.id
20 }
21
22 # Associates AZ2 private subnet (App) with private route table
23 ### @IDX:ASSOCIATION_PRIVATE_RT_APP_2
24 resource "aws_route_table_association" "private_app_rt_2" {
25   subnet_id      = aws_subnet.pri_app_az_2.id
26   route_table_id = aws_route_table.private_app_rt.id
27 }
28
29 # Associates AZ1 private subnet (Database) with private route table
30 ### @IDX:ASSOCIATION_PRIVATE_RT_DB_1
31 resource "aws_route_table_association" "private_db_rt_1" {
32   subnet_id      = aws_subnet.pri_db_az_1.id
33   route_table_id = aws_route_table.private_app_rt.id
34 }
35
36 # Associates AZ2 private subnet (Database) with private route table
37 ### @IDX:ASSOCIATION_PRIVATE_RT_DB_2
38 resource "aws_route_table_association" "private_db_rt_2" {
39   subnet_id      = aws_subnet.pri_db_az_2.id
40   route_table_id = aws_route_table.private_app_rt.id
41 }
42
```

PROBLEMS

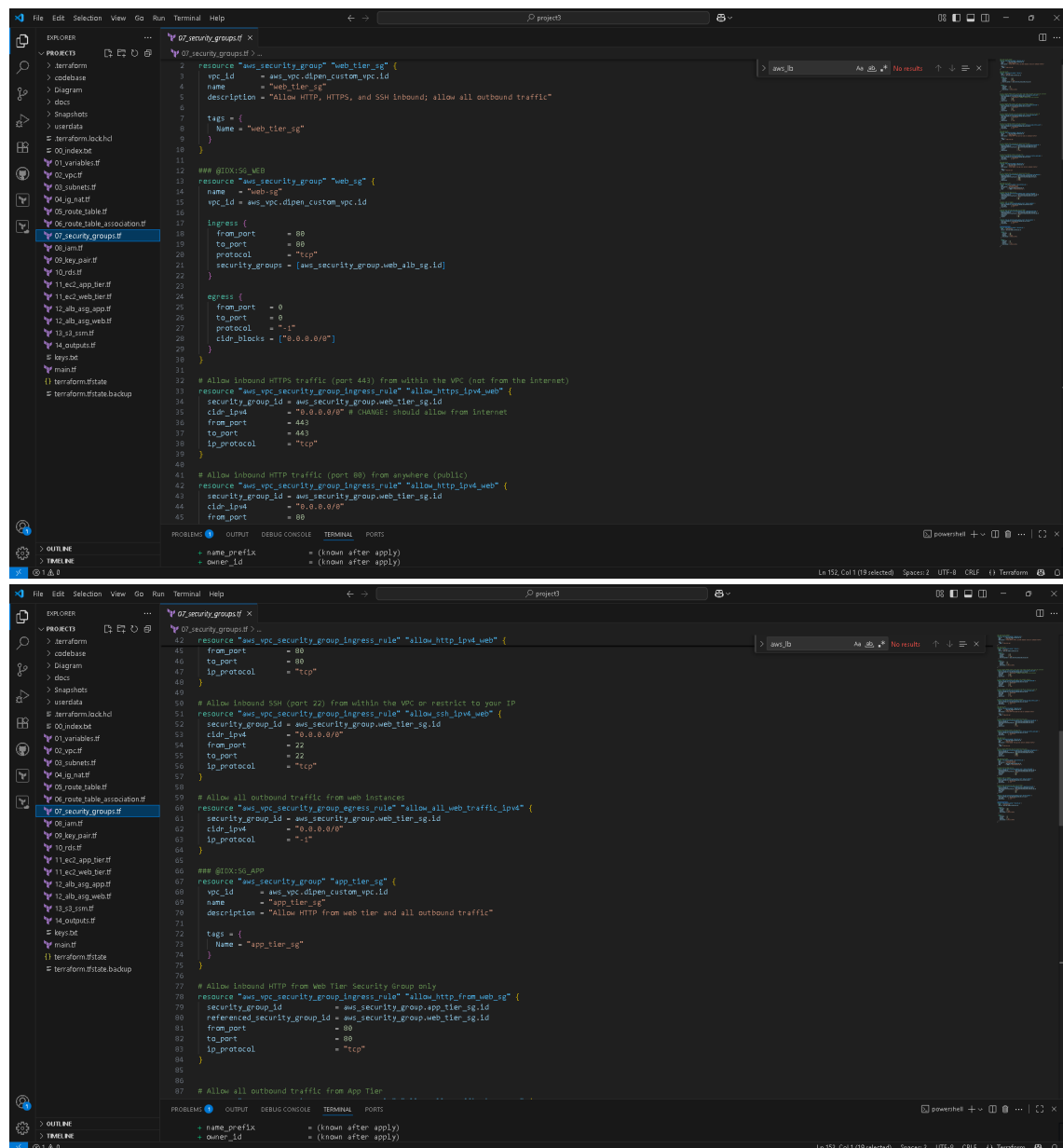
- name_prefix = (known after apply)
- owner_id = (known after apply)

Phase 7: Security Groups

Created tier-specific Security Groups:

- **Web Tier SG:** Allows inbound HTTP/HTTPS from 0.0.0.0/0.
- **App Tier SG:** Allows inbound HTTP from Web Tier SG only.
- **DB Tier SG:** Allows inbound MySQL (3306) from App Tier SG only.
- **ALB SGs:** Allow traffic from intended sources (public for Web ALB, Web SG for App ALB).

Applied **principle of least privilege**



```
resource "aws_security_group" "web_tier_sg" {
  vpc_id = aws_vpc.dipen_custom_vpc.id
  name   = "web_tier_sg"
  description = "Allow HTTP, HTTPS, and SSH inbound; allow all outbound traffic"
  tags = {
    Name = "web_tier_sg"
  }
}

### @IDK:SG_WEB
resource "aws_security_group" "web_sg" {
  name   = "web_sg"
  vpc_id = aws_vpc.dipen_custom_vpc.id

  ingress = [
    {
      from_port = 80
      to_port   = 80
      protocol  = "tcp"
      security_groups = [aws_security_group.web_alb_sg.id]
    }
  ]

  egress = [
    {
      from_port = 0
      to_port   = 0
      protocol  = "-1"
      cidr_blocks = ["0.0.0.0/0"]
    }
  ]

  # Allow Inbound HTTPS traffic (port 443) from within the VPC (not from the Internet)
  resource "aws_vpc_security_group_ingress_rule" "allow_https_ipv4_web" {
    security_group_id = aws_security_group.web_tier_sg.id
    cidr_ipv4         = "0.0.0.0/0" # CHANG: should allow from Internet
    from_port         = 443
    to_port           = 443
    ip_protocol       = "tcp"
  }

  # Allow Inbound HTTP traffic (port 80) from anywhere (public)
  resource "aws_vpc_security_group_ingress_rule" "allow_http_ipv4_web" {
    security_group_id = aws_security_group.web_tier_sg.id
    cidr_ipv4         = "0.0.0.0/0"
    from_port         = 80
    to_port           = 80
    ip_protocol       = "tcp"
  }

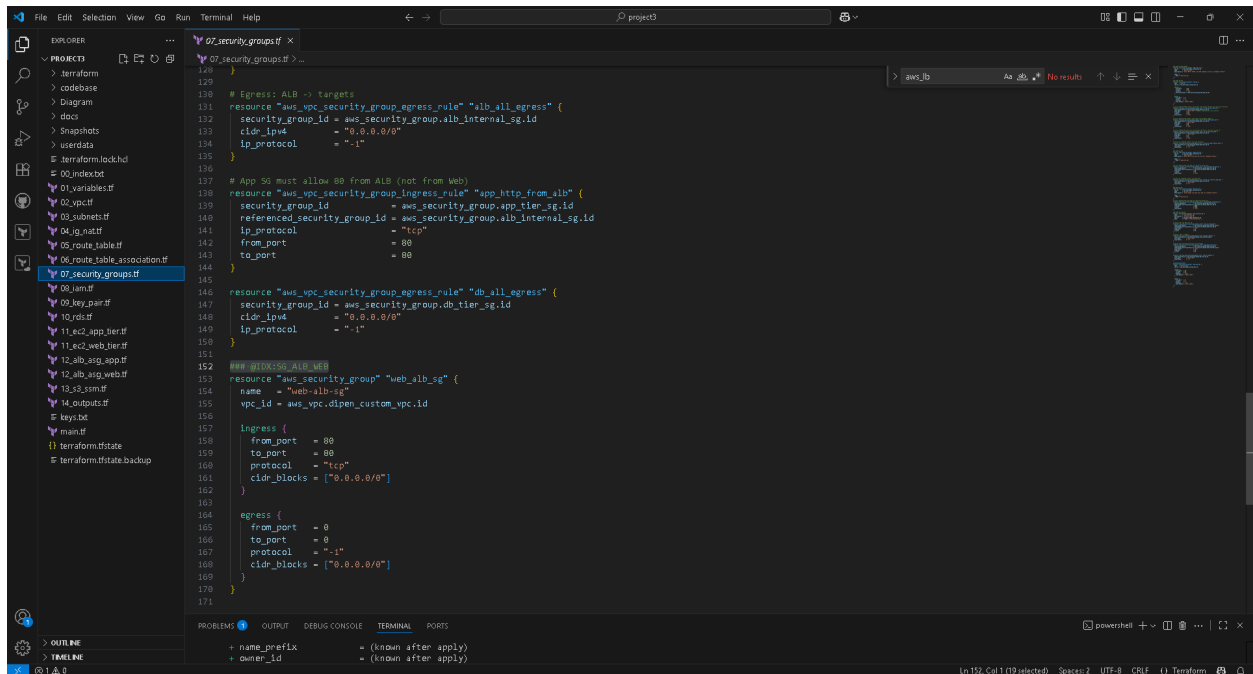
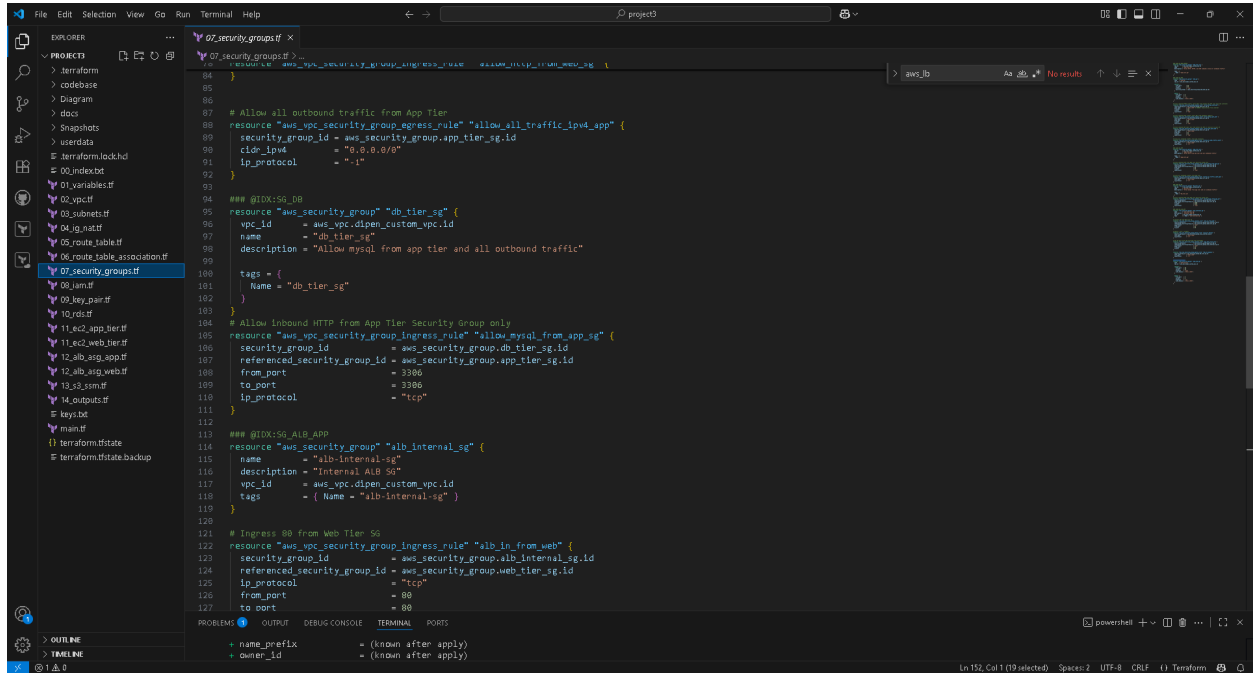
  # Allow Inbound SSH (port 22) from within the VPC or restrict to your IP
  resource "aws_vpc_security_group_ingress_rule" "allow_ssh_ipv4_web" {
    security_group_id = aws_security_group.web_tier_sg.id
    cidr_ipv4         = "0.0.0.0/0"
    from_port         = 22
    to_port           = 22
    ip_protocol       = "tcp"
  }

  # Allow all outbound traffic from web instances
  resource "aws_vpc_security_group_egress_rule" "allow_all_web_traffic_ipv4" {
    security_group_id = aws_security_group.web_tier_sg.id
    cidr_ipv4         = "0.0.0.0/0"
    ip_protocol       = "-1"
  }

  ### @IDK:SG_APP
  resource "aws_security_group" "app_tier_sg" {
    vpc_id = aws_vpc.dipen_custom_vpc.id
    name   = "app_tier_sg"
    description = "Allow HTTP from web tier and all outbound traffic"
    tags = {
      Name = "app_tier_sg"
    }
  }

  # Allow Inbound HTTP from Web Tier Security Group only
  resource "aws_vpc_security_group_ingress_rule" "allow_http_from_web_sg" {
    security_group_id = aws_security_group.app_tier_sg.id
    referenced_security_group_id = aws_security_group.web_tier_sg.id
    from_port         = 80
    to_port           = 80
    ip_protocol       = "tcp"
  }

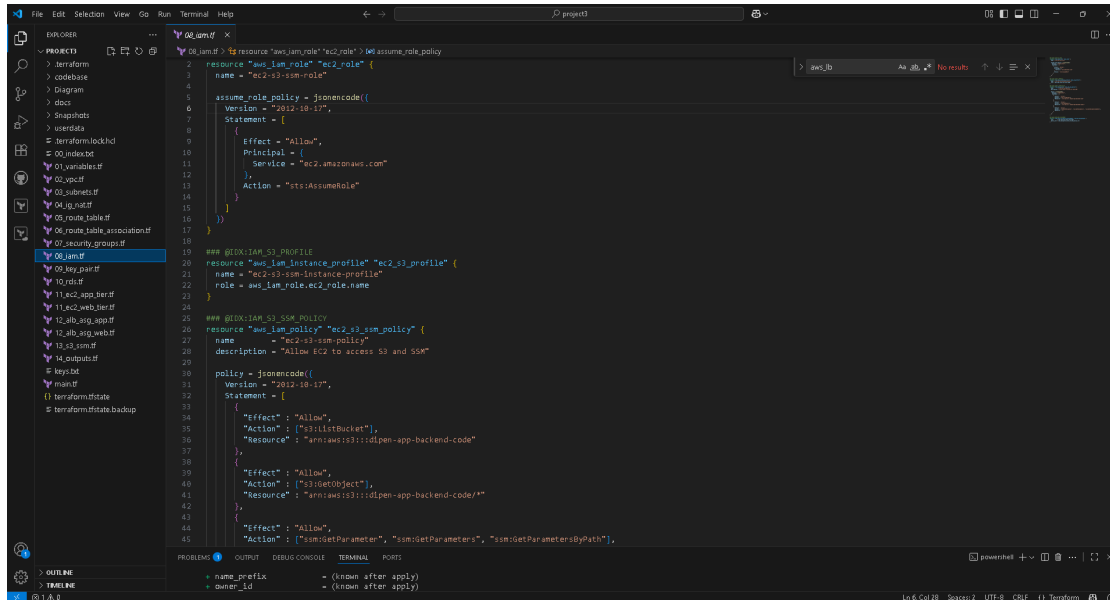
  # Allow all outbound traffic from App Tier
}
```



Phase 8: IAM

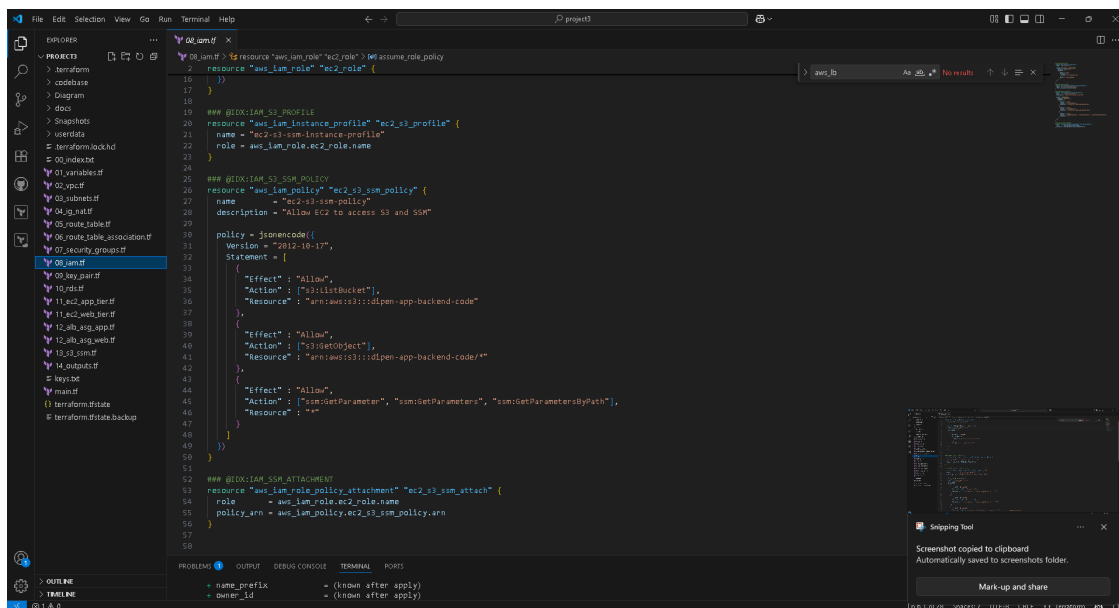
Provisioned IAM roles and instance profiles:

- **EC2 Role:** Grants access to S3 and SSM Parameter Store.
 - **Policies:** Attached managed and custom policies for required permissions.
- Attached IAM instance profile to EC2 instances to enable secure, credential-free AWS service access.



This screenshot shows the VS Code editor with a Terraform configuration file named `08_iam.tf`. The configuration defines an EC2 role, an instance profile, and a custom policy for S3 and SSM access.

```
08_iam.tf
1 resource "aws_iam_role" "ec2_role" {
2   name = "ec2-s3-ssm-role"
3 }
4
5 assume_role_policy = jsonencode({
6   Version = "2012-10-17",
7   Statement = [
8     {
9       Effect = "Allow",
10      Principal = {
11        Service = "ec2.amazonaws.com"
12      },
13      Action = "sts:AssumeRole"
14    }
15  ]
16 })
17
18 ## BDX: IAM S3 PROFILE
19
20 resource "aws_iam_instance_profile" "ec2_s3_profile" {
21   name = "ec2-s3-ssm-instance-profile"
22   role = aws_iam_role.ec2_role.name
23 }
24
25 ## BDX: IAM S3 SSM POLICY
26 resource "aws_iam_policy" "ec2_s3_ssm_policy" {
27   name = "ec2-s3-ssm-policy"
28   description = "Allow EC2 to access S3 and SSM"
29 }
30
31 policy = jsonencode({
32   Version = "2012-10-17",
33   Statement = [
34     {
35       "Effect": "Allow",
36       "Action": ["s3:ListBucket"],
37       "Resource": ["arn:aws:s3:::dlpen-app-backend-code"]
38     },
39     {
40       "Effect": "Allow",
41       "Action": ["s3:GetObject"],
42       "Resource": ["arn:aws:s3:::dlpen-app-backend-code/*"]
43     },
44     {
45       "Effect": "Allow",
46       "Action": ["ssm:GetParameter", "ssm:PutParameter", "ssm:DeleteParameter"],
47       "Resource": ["*"]
48     }
49   ]
50 })
51
52 ## BDX: IAM S3 SSM ATTACHMENT
53 resource "aws_iam_policy_attachment" "ec2_s3_ssm_attach" {
54   name = "ec2-s3-ssm-attach"
55   policy_arn = aws_iam_policy.ec2_s3_ssm_policy.arn
56 }
```

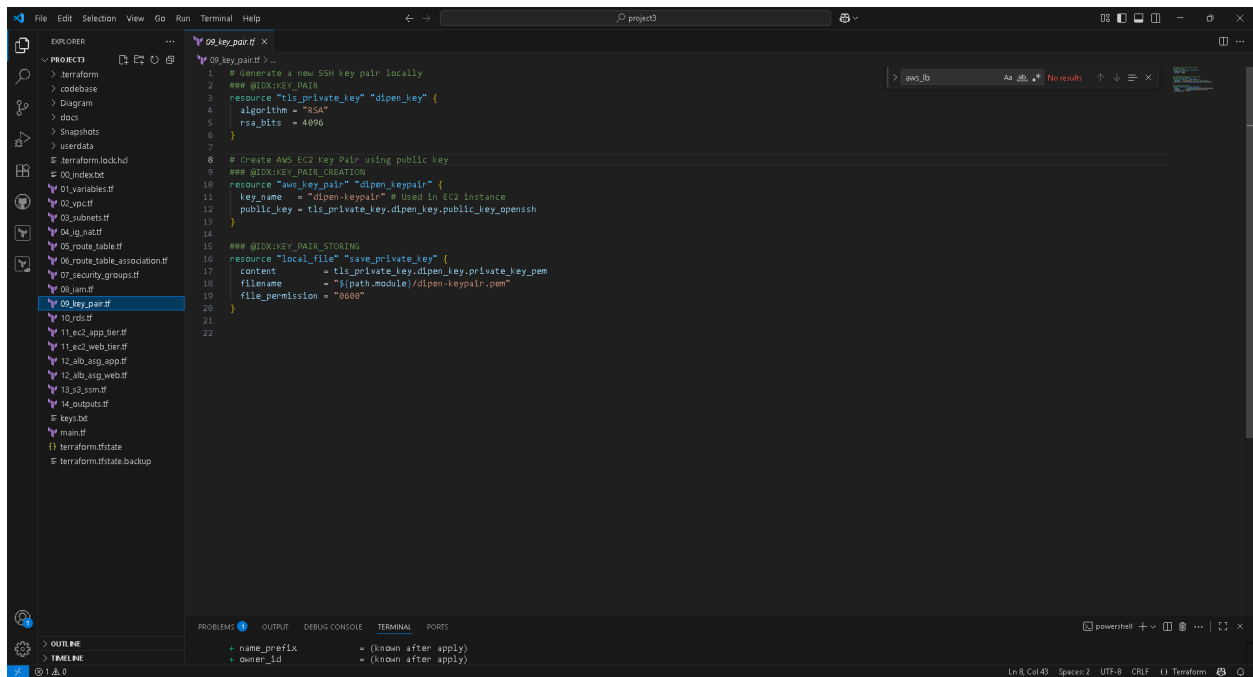


This screenshot shows the same VS Code editor with the Terraform configuration file `08_iam.tf`. The configuration is identical to the previous one, but it includes a screenshot of the AWS console in the bottom right corner, showing the IAM console interface. The screenshot is titled "Screenshot copied to clipboard" and "Automatically saved to screenshots folder.".

Phase 9: Key Pair Creation and Storing

Generated AWS Key Pair for SSH access:

- Stored .pem file securely.
- Associated with EC2 instances for secure administrative access.



The screenshot shows a VS Code editor window with a project named 'project3'. The file explorer on the left shows a directory structure for a Terraform project, including files like '01_variables.tf', '02_vpc.tf', '03_subnet.tf', '04_ig.tf', '05_route_table.tf', '06_route_table_association.tf', '07_security_group.tf', '08_key.tf', '09_key_pair.tf', '10_rds.tf', '11_ec2_app_server.tf', '12_alb.tf', '13_s3.tf', '14_outputs.tf', 'keys.tf', 'main.tf', 'terraform.tfstate', and 'terraform.tfstate.backup'. The file '09_key_pair.tf' is selected and open in the editor. The code in the editor is as follows:

```
1 # Generate a new SSH key pair locally
2 ### @IDX:KEY_PAIR
3 resource "tls_private_key" "dipen_key" {
4   algorithm = "RSA"
5   rsa_bits  = 4096
6 }
7
8 # Create AWS EC2 Key Pair using public key
9 ### @IDX:KEY_PAIR_CREATION
10 resource "aws_key_pair" "dipen_keypair" {
11   key_name   = "dipen-keypair" # Used in EC2 Instance
12   public_key = tls_private_key.dipen_key.public_key_openssh
13 }
14
15 ### @IDX:KEY_PAIR_STORAGE
16 resource "local_file" "save_private_key" {
17   content         = tls_private_key.dipen_key.private_key_pem
18   filename        = "[path.module]/dipen-keypair.pem"
19   file_permission = "600"
20 }
21
22
```

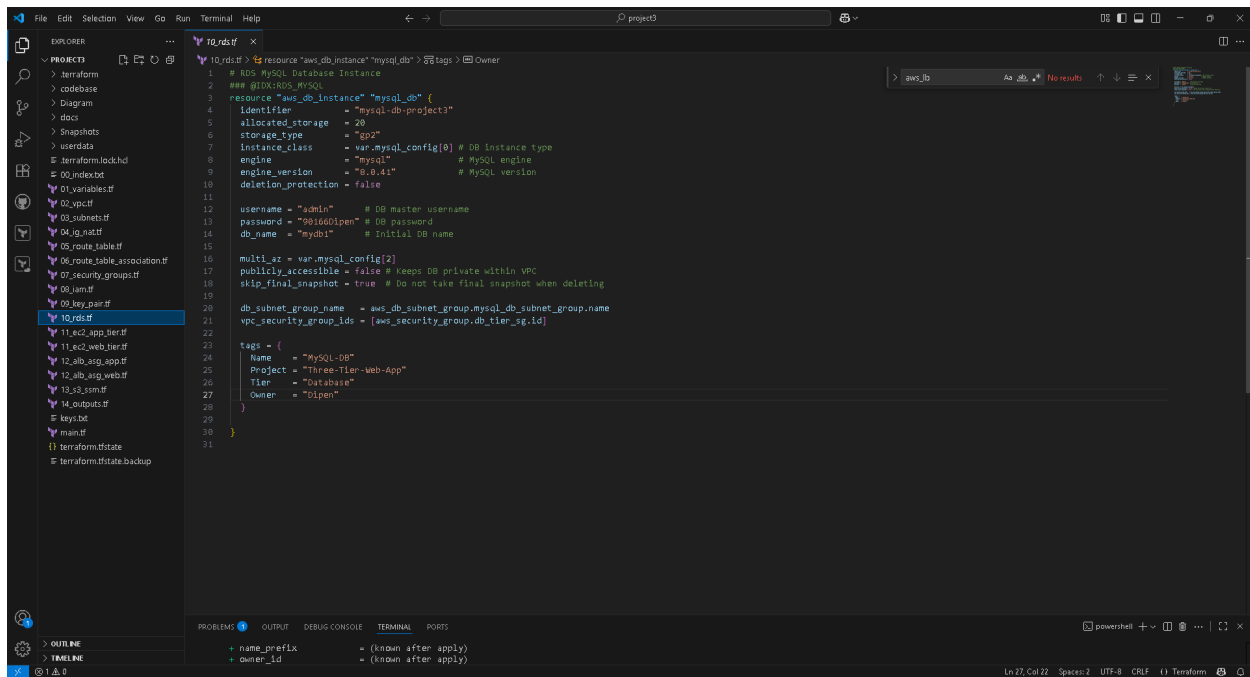
The bottom of the editor shows the 'PROBLEMS' panel with two items:

- name_prefix = (known after apply)
- owner_id = (known after apply)

Phase 10: RDS Creation

Deployed **Amazon RDS MySQL** instance:

- Multi-AZ deployment in private DB subnets.
- Security Group restricts access to App Tier only.

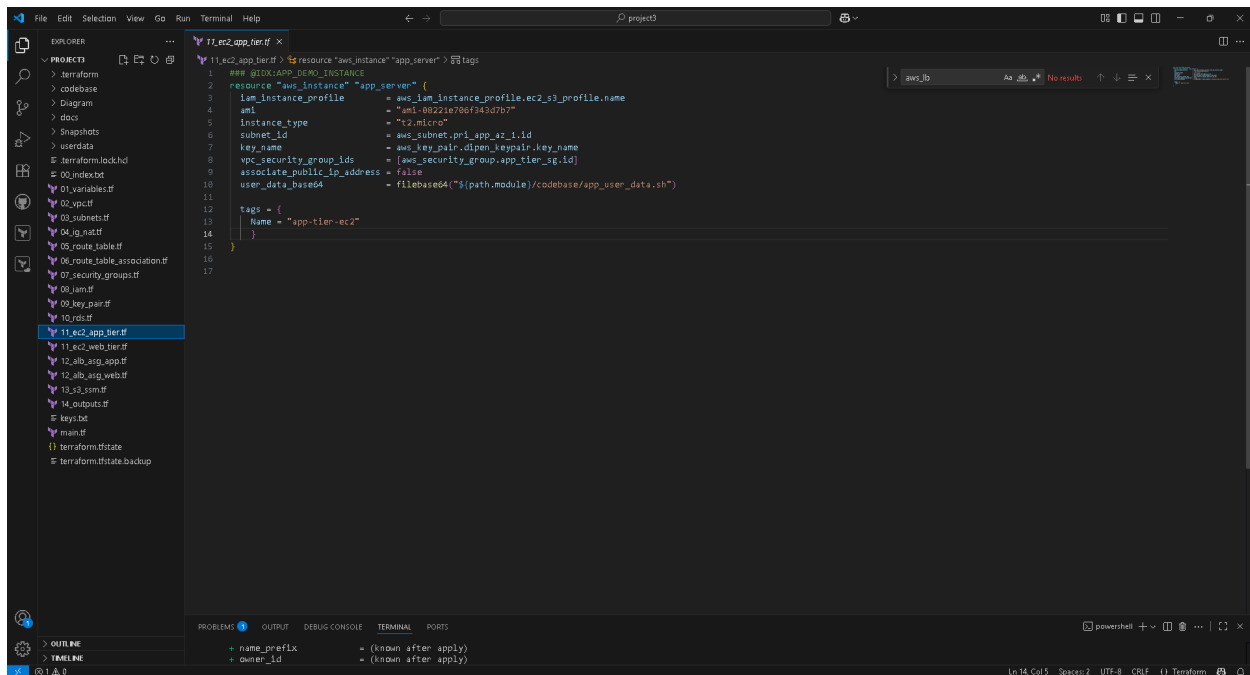


```
1 # AWS MySQL Database Instance
2 ## @ID:RDS_MYSQL
3 resource "aws_db_instance" "mysql_db" {
4   identifier      = "mysql-db-project3"
5   allocated_storage = 20
6   storage_type    = "gp2"
7   instance_class  = var.mysql_config[0] # DB Instance type
8   engine          = "mysql"             # MySQL engine
9   engine_version  = "8.0.41"           # MySQL version
10  deletion_protection = false
11
12  username = "admin" # DB master username
13  password = "091601pen" # DB password
14  db_name = "mydb1" # Initial DB name
15
16  multi_az = var.mysql_config[2]
17  publicly_accessible = false # Keeps DB private within VPC
18  skip_final_snapshot = true # Do not take final snapshot when deleting
19
20  db_subnet_group_name = aws_db_subnet_group.mysql_db_subnet_group.name
21  vpc_security_group_ids = [aws_security_group.db_tier_sg.id]
22
23  tags = {
24    Name     = "MySQL-DB"
25    Project  = "Three-Tier-Web-App"
26    Tier     = "Database"
27    Owner    = "Oliver"
28  }
29
30 }
31
```

Phase 11: Test Instance Creation

Launched temporary EC2 instances in Web and App tiers:

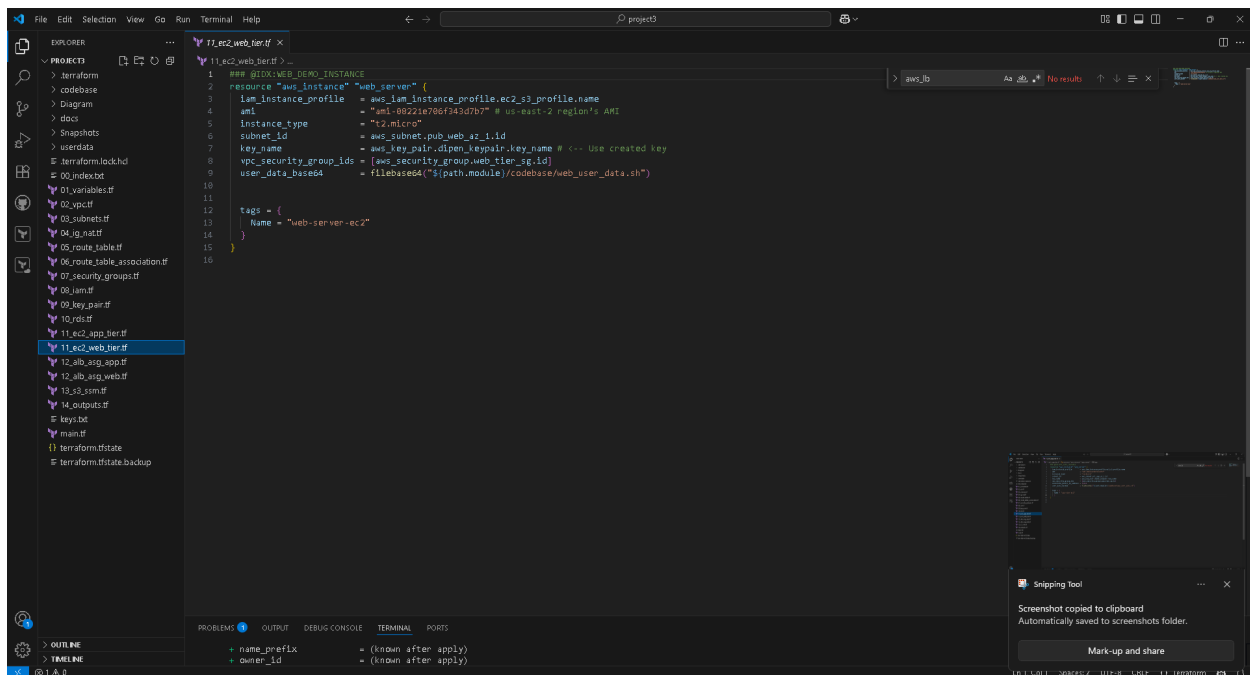
- Verified routing, internet access, and inter-tier communication.
- Tested DB connectivity from App tier to RDS MySQL.



The screenshot shows the VS Code editor with a Terraform configuration file named `11.ec2_app_server.tf`. The configuration defines an `aws_instance` resource for the `app_server`. The instance profile is `aws_iam_instance_profile.ec2_s3_profile.name`, the AMI is `ami-0822e706f343d7b77`, and the instance type is `t2.micro`. The subnet is `aws_subnet.pr1_app_az_1.id`, and the key pair is `aws_key_pair.dipen_keypair.key_name`. The VPC security group is `[aws_security_group.app_tier_sg.id]`. The `associate_public_ip_address` is set to `false`, and the `user_data` is a base64-encoded shell script. The tags include `app-tier-ec2`.

```
1 ## @ID:APP_SERVER_INSTANCE
2 resource "aws_instance" "app_server" {
3   iam_instance_profile = aws_iam_instance_profile.ec2_s3_profile.name
4   ami                  = "ami-0822e706f343d7b77"
5   instance_type        = "t2.micro"
6   subnet_id            = aws_subnet.pr1_app_az_1.id
7   key_name             = aws_key_pair.dipen_keypair.key_name
8   vpc_security_group_ids = [aws_security_group.app_tier_sg.id]
9   associate_public_ip_address = false
10  user_data_base64      = filebase64("${path.module}/codebase/app_user_data.sh")
11
12  tags = {
13    Name = "app-tier-ec2"
14  }
15
16
17
```

The bottom status bar shows the file is at line 14, column 5, with 2 spaces, UTF-8 encoding, and CRLF line endings.



The screenshot shows the VS Code editor with a Terraform configuration file named `11.ec2_web_server.tf`. The configuration defines an `aws_instance` resource for the `web_server`. The instance profile is `aws_iam_instance_profile.ec2_s3_profile.name`, the AMI is `ami-0822e706f343d7b77` (commented as `# us-east-2 region's AMI`), and the instance type is `t2.micro`. The subnet is `aws_subnet.pub_web_az_1.id`, and the key pair is `aws_key_pair.dipen_keypair.key_name` (commented as `# Use created key`). The VPC security group is `[aws_security_group.web_tier_sg.id]`, and the `user_data` is a base64-encoded shell script. The tags include `web-server-ec2`.

```
1 ## @ID:WEB_SERVER_INSTANCE
2 resource "aws_instance" "web_server" {
3   iam_instance_profile = aws_iam_instance_profile.ec2_s3_profile.name
4   ami                  = "ami-0822e706f343d7b77" # us-east-2 region's AMI
5   instance_type        = "t2.micro"
6   subnet_id            = aws_subnet.pub_web_az_1.id
7   key_name             = aws_key_pair.dipen_keypair.key_name # Use created key
8   vpc_security_group_ids = [aws_security_group.web_tier_sg.id]
9   user_data_base64     = filebase64("${path.module}/codebase/web_user_data.sh")
10
11  tags = {
12    Name = "web-server-ec2"
13  }
14
15
16
```

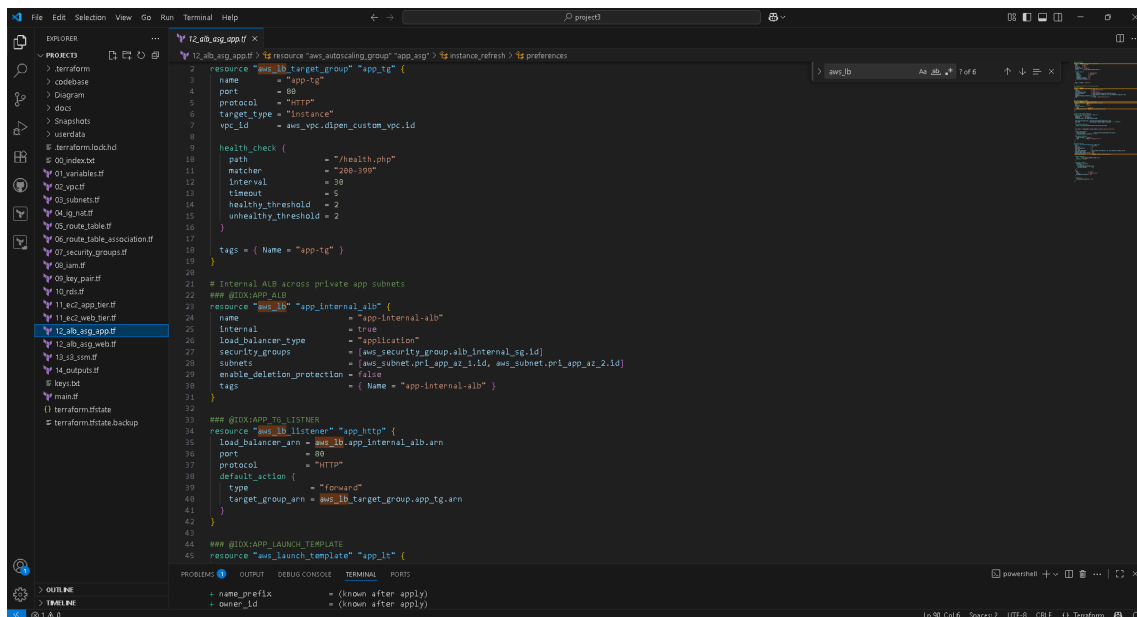
The bottom status bar shows the file is at line 1, column 1, with 2 spaces, UTF-8 encoding, and CRLF line endings.

A Snipping Tool window is open in the bottom right corner, displaying the message: "Screenshot copied to clipboard. Automatically saved to screenshots folder. Mark-up and share".

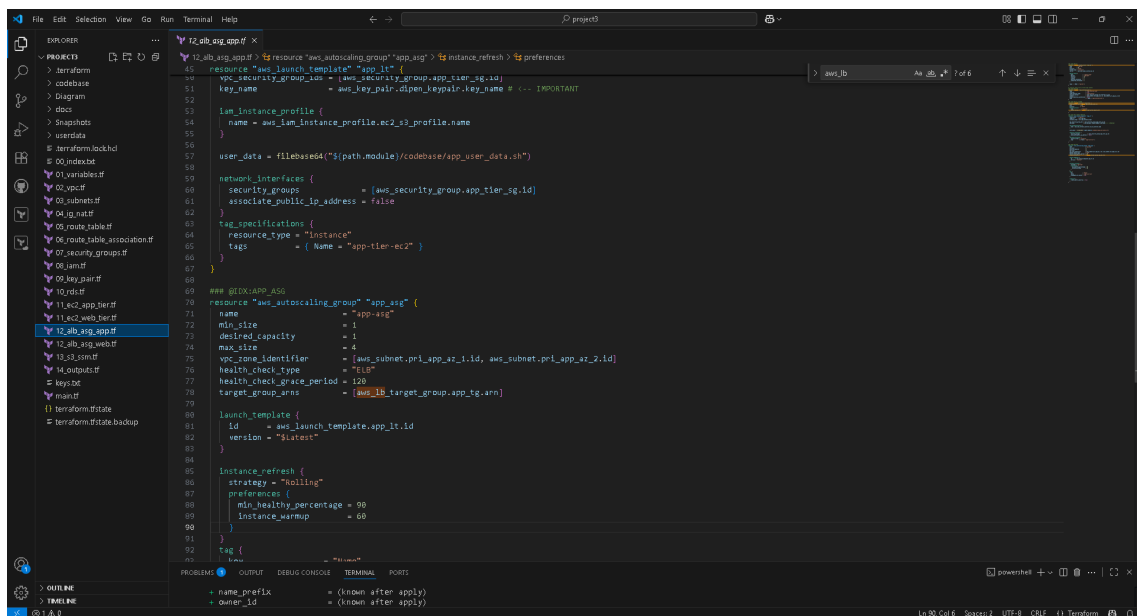
Phase 11: Application Load Balancer, Target Groups, Auto Scaling Group

App Tier

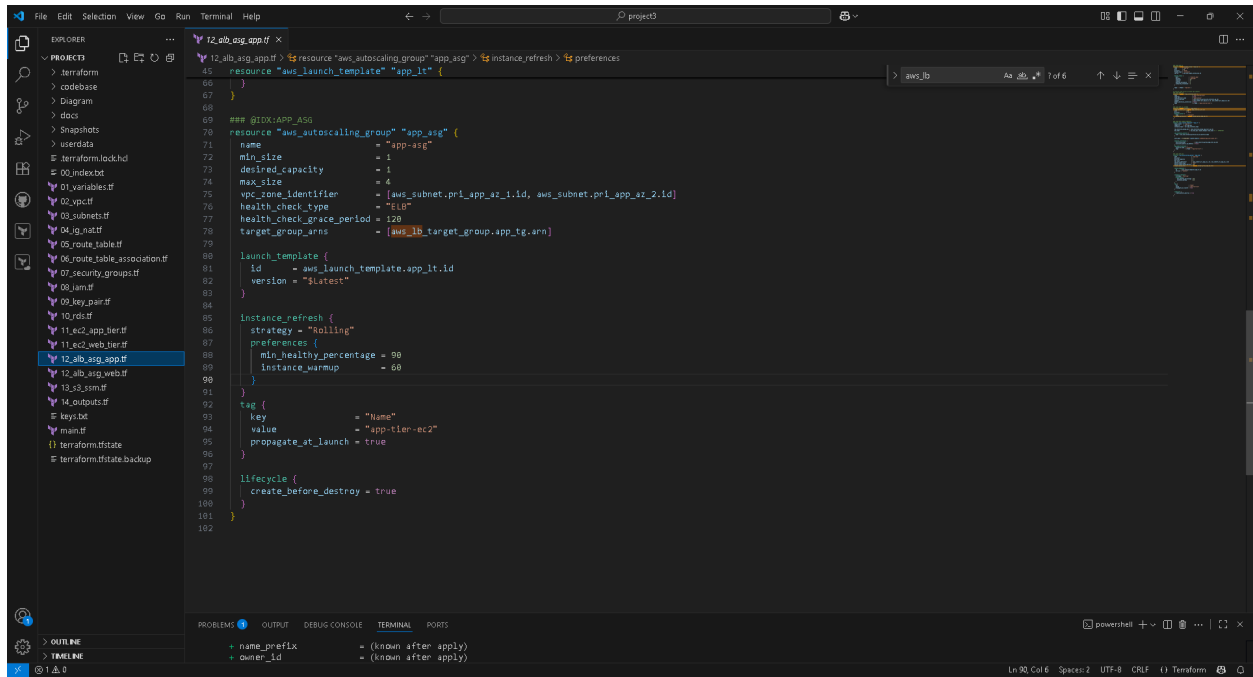
- **Internal ALB:** Routes traffic from Web Tier to App Tier.
- **Target Group:** Health checks on App EC2 instances.
- **Launch Template:** User data to configure App EC2 (PHP, DB connection).
- **ASG:** Ensures availability and scales based on demand.



```
1 resource "aws_target_group" "app_tg" {
2   name = "app-tg"
3   port = 80
4   protocol = "HTTP"
5   target_type = "Instance"
6   vpc_id = aws_vpc.dipen_custom_vpc.id
7
8   health_check {
9     path = "/health.php"
10    matcher = "200-399"
11    interval = 30
12    timeout = 5
13    healthy_threshold = 2
14    unhealthy_threshold = 2
15  }
16
17  tags = { Name = "app-tg" }
18}
19
20# Internal ALB across private app subnets
21aws_alb_target_group_target "app_tg_target" {
22  target_group_arn = aws_target_group.app_tg.arn
23  target_id = aws_subnet.pri_app_az_1.id
24}
25
26resource "aws_alb" "app_internal_alb" {
27  name = "app-internal-alb"
28  internal = true
29  load_balancer_type = "Application"
30  security_groups = [aws_security_group.alb_internal_sg.id]
31  subnets = [aws_subnet.pri_app_az_1.id, aws_subnet.pri_app_az_2.id]
32  enable_deletion_protection = false
33  tags = { Name = "app-internal-alb" }
34}
35
36# AWS ALB Listener
37resource "aws_alb_listener" "app_http" {
38  load_balancer_arn = aws_alb.app_internal_alb.arn
39  port = 80
40  protocol = "HTTP"
41  default_action {
42    type = "forward"
43    target_group_arn = aws_target_group.app_tg.arn
44  }
45}
46
47# AWS ASG Launch Template
48resource "aws_launch_template" "app_lt" {
49  name_prefix = "app-launch-template"
50  owner_id = "amazon"
51  image_id = "ami-0a72875f"
52  instance_type = "t3.micro"
53  key_name = "dipen_keypair"
54  user_data = file("${path.module}/codebase/app_user_data.sh")
55  network_interfaces = [aws_network_interface.app_tier_sg.id]
56  associate_public_ip_address = false
57  tag_specifications {
58    resource_type = "Instance"
59    tags = { Name = "app-tier-ec2" }
60  }
61}
62
63# AWS ASG
64resource "aws_autoscaling_group" "app_asg" {
65  name = "app-asg"
66  min_size = 1
67  desired_capacity = 1
68  max_size = 4
69  vpc_zone_identifier = [aws_subnet.pri_app_az_1.id, aws_subnet.pri_app_az_2.id]
70  health_check_type = "ELB"
71  health_check_grace_period = 120
72  target_group_arns = [aws_target_group.app_tg.arn]
73
74  launch_template {
75    id = aws_launch_template.app_lt.id
76    version = "Latest"
77  }
78
79  instance_refresh {
80    strategy = "Rolling"
81    preferences {
82      min_healthy_percentage = 90
83      instance_warmup = 60
84    }
85  }
86
87  tag {
88    key = "Name"
89    value = "app-asg"
90  }
91}
```

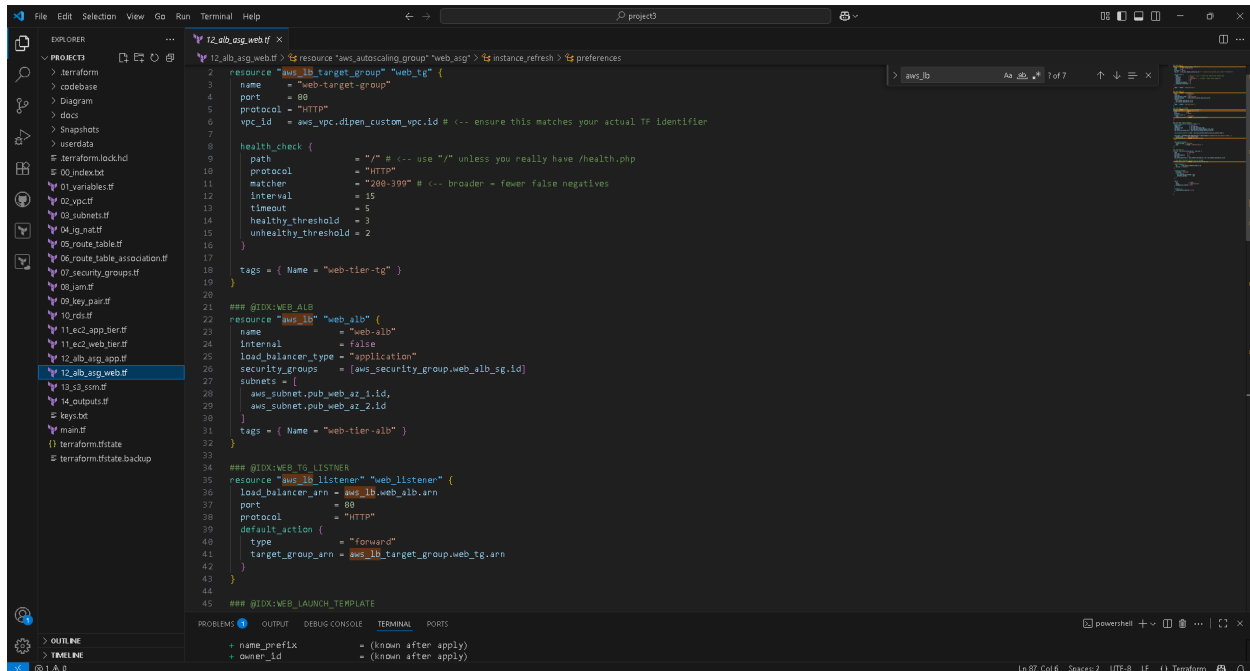


```
1 resource "aws_target_group" "app_tg" {
2   name = "app-tg"
3   port = 80
4   protocol = "HTTP"
5   target_type = "Instance"
6   vpc_id = aws_vpc.dipen_custom_vpc.id
7
8   health_check {
9     path = "/health.php"
10    matcher = "200-399"
11    interval = 30
12    timeout = 5
13    healthy_threshold = 2
14    unhealthy_threshold = 2
15  }
16
17  tags = { Name = "app-tg" }
18}
19
20# Internal ALB across private app subnets
21aws_alb_target_group_target "app_tg_target" {
22  target_group_arn = aws_target_group.app_tg.arn
23  target_id = aws_subnet.pri_app_az_1.id
24}
25
26resource "aws_alb" "app_internal_alb" {
27  name = "app-internal-alb"
28  internal = true
29  load_balancer_type = "Application"
30  security_groups = [aws_security_group.alb_internal_sg.id]
31  subnets = [aws_subnet.pri_app_az_1.id, aws_subnet.pri_app_az_2.id]
32  enable_deletion_protection = false
33  tags = { Name = "app-internal-alb" }
34}
35
36# AWS ALB Listener
37resource "aws_alb_listener" "app_http" {
38  load_balancer_arn = aws_alb.app_internal_alb.arn
39  port = 80
40  protocol = "HTTP"
41  default_action {
42    type = "forward"
43    target_group_arn = aws_target_group.app_tg.arn
44  }
45}
46
47# AWS ASG Launch Template
48resource "aws_launch_template" "app_lt" {
49  name_prefix = "app-launch-template"
50  owner_id = "amazon"
51  image_id = "ami-0a72875f"
52  instance_type = "t3.micro"
53  key_name = "dipen_keypair"
54  user_data = file("${path.module}/codebase/app_user_data.sh")
55  network_interfaces = [aws_network_interface.app_tier_sg.id]
56  associate_public_ip_address = false
57  tag_specifications {
58    resource_type = "Instance"
59    tags = { Name = "app-tier-ec2" }
60  }
61}
62
63# AWS ASG
64resource "aws_autoscaling_group" "app_asg" {
65  name = "app-asg"
66  min_size = 1
67  desired_capacity = 1
68  max_size = 4
69  vpc_zone_identifier = [aws_subnet.pri_app_az_1.id, aws_subnet.pri_app_az_2.id]
70  health_check_type = "ELB"
71  health_check_grace_period = 120
72  target_group_arns = [aws_target_group.app_tg.arn]
73
74  launch_template {
75    id = aws_launch_template.app_lt.id
76    version = "Latest"
77  }
78
79  instance_refresh {
80    strategy = "Rolling"
81    preferences {
82      min_healthy_percentage = 90
83      instance_warmup = 60
84    }
85  }
86
87  tag {
88    key = "Name"
89    value = "app-asg"
90  }
91}
```

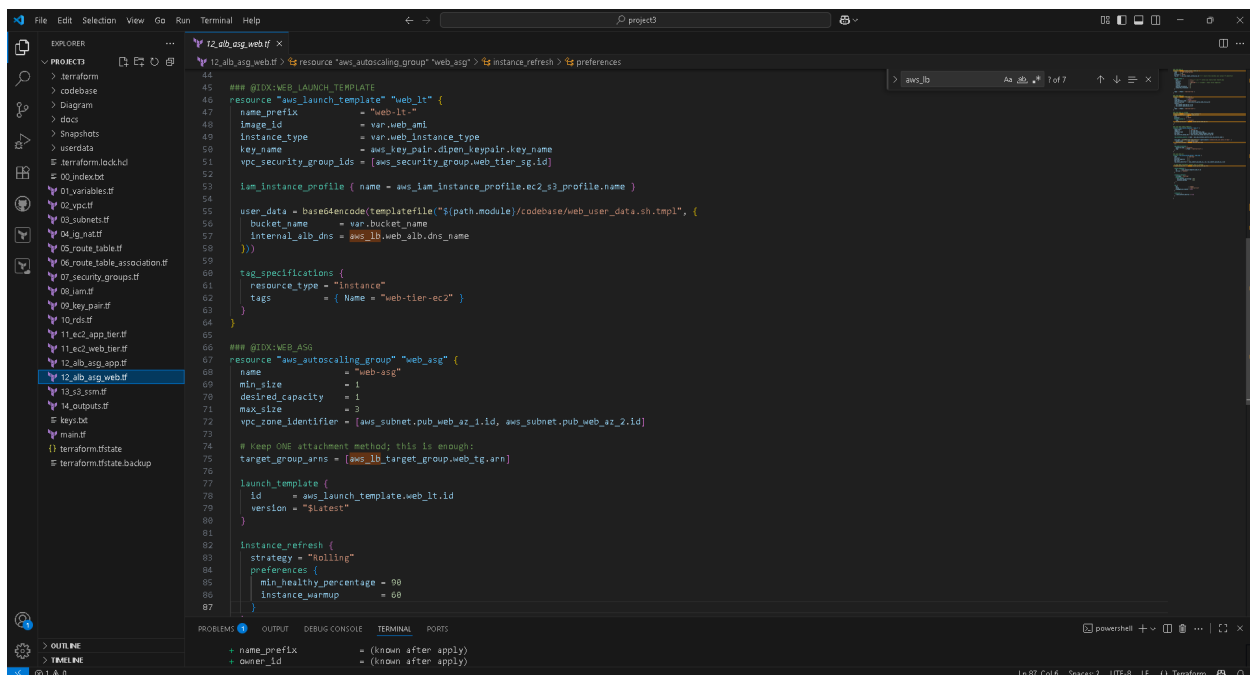


Web Tier

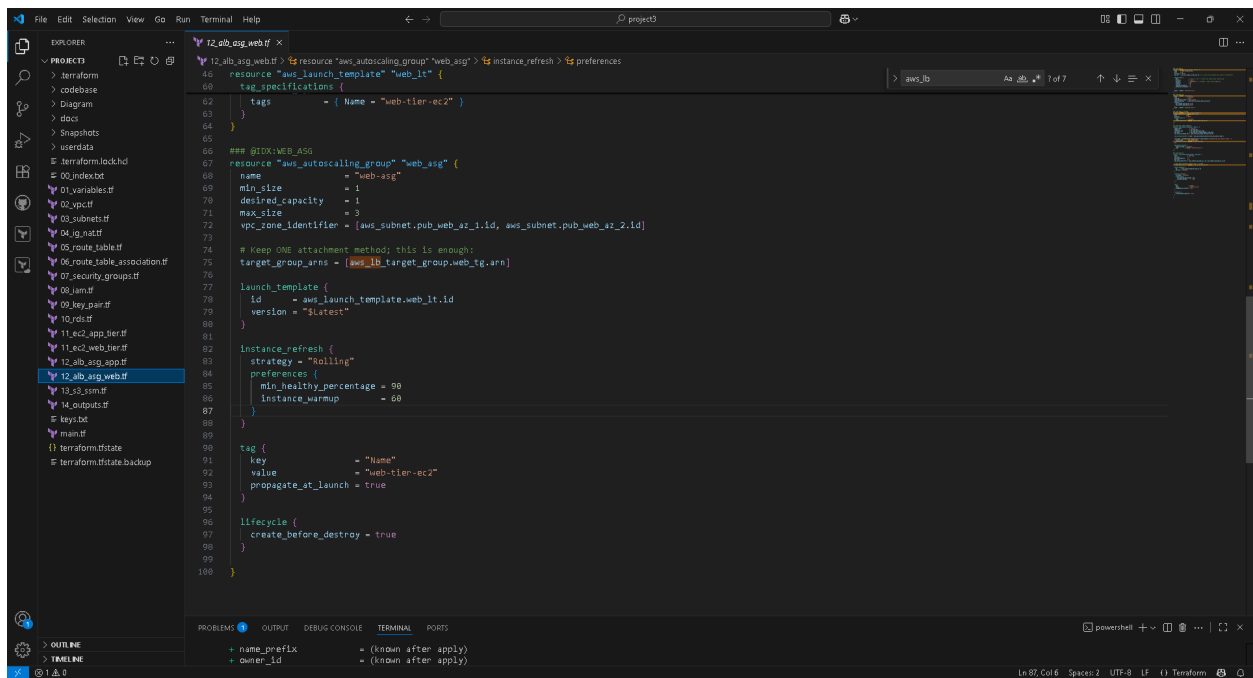
- **Public ALB:** Routes public HTTP/HTTPS requests.
- **Target Group:** Health checks on Web EC2 instances.
- **Launch Template:** User data to deploy static files from S3 and proxy /api calls to App ALB.
- **ASG:** Maintains desired capacity and handles scaling events.



```
1 resource "aws_alb_target_group" "web_tg" {
2   name = "web-target-group"
3   port = 80
4   protocol = "HTTP"
5   vpc_id = aws_vpc.dipen_custom_vpc.id # <-- ensure this matches your actual TF Identifier
6
7   health_check {
8     path = "/" # <-- use "/" unless you really have /health.php
9     protocol = "HTTP"
10    matcher = "200-399" # <-- broader = fewer false negatives
11    interval = 15
12    timeout = 5
13    healthy_threshold = 3
14    unhealthy_threshold = 2
15  }
16
17  tags = { Name = "web-tier-tg" }
18
19 }
20
21 ## @IDX:WEB_ALB
22 resource "aws_alb" "web_alb" {
23   name = "web-alb"
24   internal = false
25   load_balancer_type = "application"
26   security_groups = [aws_security_group.web_alb_sg.id]
27   subnets = [
28     aws_subnet.pub_web_az_1.id,
29     aws_subnet.pub_web_az_2.id
30   ]
31   tags = { Name = "web-tier-alb" }
32
33 }
34
35 ## @IDX:WEB_TO_LISTENER
36 resource "aws_lb_listener" "web_listener" {
37   load_balancer_arn = aws_lb.web_alb.arn
38   port = 80
39   protocol = "HTTP"
40   default_action {
41     type = "forward"
42     target_group_arn = aws_lb_target_group.web_tg.arn
43   }
44 }
45
46 ## @IDX:WEB_LAUNCH_TEMPLATE
47 resource "aws_launch_template" "web_lt" {
48   name_prefix = "web-lt-"
49   image_id = var.web_ami
50   instance_type = var.web_instance_type
51   key_name = aws_key_pair.dipen_tepair.key_name
52   vpc_security_group_ids = [aws_security_group.web_tier_sg.id]
53
54   iam_instance_profile { name = aws_iam_instance_profile.ec2_s3_profile.name }
55
56   user_data = base64encode(templatefile("${path.module}/codebase/web_user_data.sh.tpl", {
57     bucket_name = var.bucket_name
58     internal_alb_dns = aws_lb.web_alb.dns_name
59   }))
60
61   tag_specifications {
62     resource_type = "Instance"
63     tags = { Name = "web-tier-ec2" }
64   }
65 }
66
67 ## @IDX:WEB_ASG
68 resource "aws_autoscaling_group" "web_asg" {
69   name = "web-asg"
70   min_size = 1
71   desired_capacity = 1
72   max_size = 2
73   vpc_zone_identifier = [aws_subnet.pub_web_az_1.id, aws_subnet.pub_web_az_2.id]
74
75   # Keep ONE attachment method; this is enough:
76   target_group_arns = [aws_lb_target_group.web_tg.arn]
77
78   launch_template {
79     id = aws_launch_template.web_lt.id
80     version = "latest"
81   }
82
83   instance_refresh {
84     strategy = "Rolling"
85     preferences {
86       min_healthy_percentage = 90
87       instance_warmup = 60
88     }
89   }
90 }
```

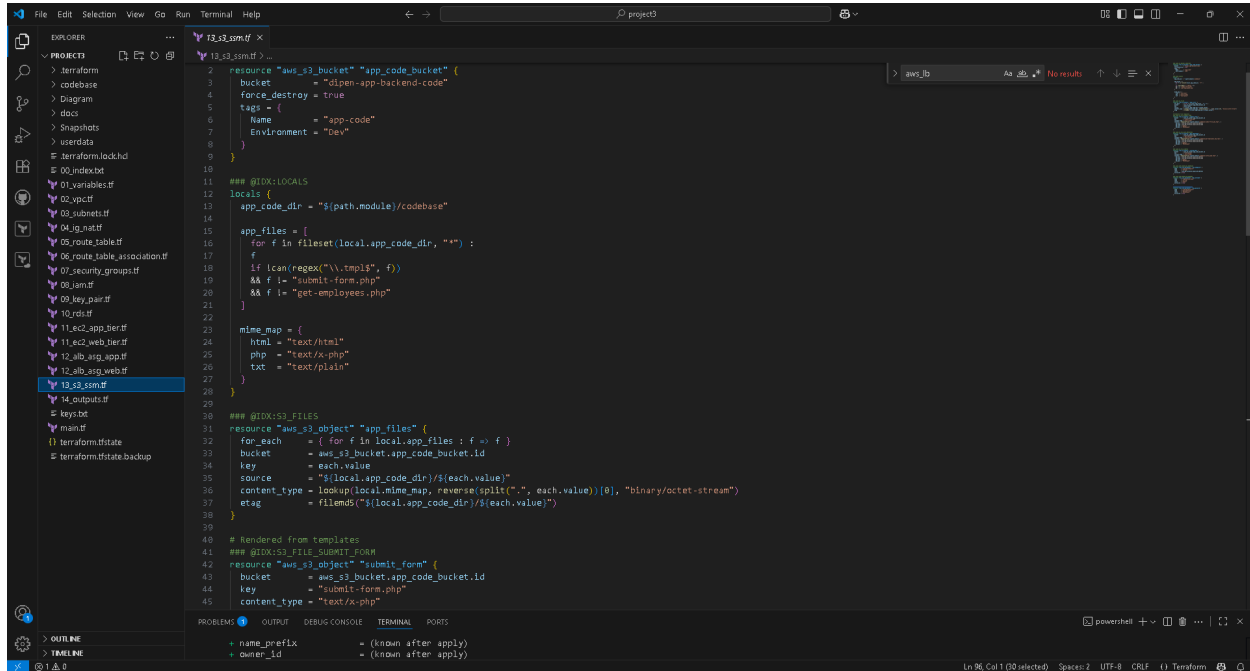


```
44
45 ## @IDX:WEB_LAUNCH_TEMPLATE
46 resource "aws_launch_template" "web_lt" {
47   name_prefix = "web-lt-"
48   image_id = var.web_ami
49   instance_type = var.web_instance_type
50   key_name = aws_key_pair.dipen_tepair.key_name
51   vpc_security_group_ids = [aws_security_group.web_tier_sg.id]
52
53   iam_instance_profile { name = aws_iam_instance_profile.ec2_s3_profile.name }
54
55   user_data = base64encode(templatefile("${path.module}/codebase/web_user_data.sh.tpl", {
56     bucket_name = var.bucket_name
57     internal_alb_dns = aws_lb.web_alb.dns_name
58   }))
59
60   tag_specifications {
61     resource_type = "Instance"
62     tags = { Name = "web-tier-ec2" }
63   }
64 }
65
66 ## @IDX:WEB_ASG
67 resource "aws_autoscaling_group" "web_asg" {
68   name = "web-asg"
69   min_size = 1
70   desired_capacity = 1
71   max_size = 2
72   vpc_zone_identifier = [aws_subnet.pub_web_az_1.id, aws_subnet.pub_web_az_2.id]
73
74   # Keep ONE attachment method; this is enough:
75   target_group_arns = [aws_lb_target_group.web_tg.arn]
76
77   launch_template {
78     id = aws_launch_template.web_lt.id
79     version = "latest"
80   }
81
82   instance_refresh {
83     strategy = "Rolling"
84     preferences {
85       min_healthy_percentage = 90
86       instance_warmup = 60
87     }
88   }
89 }
```

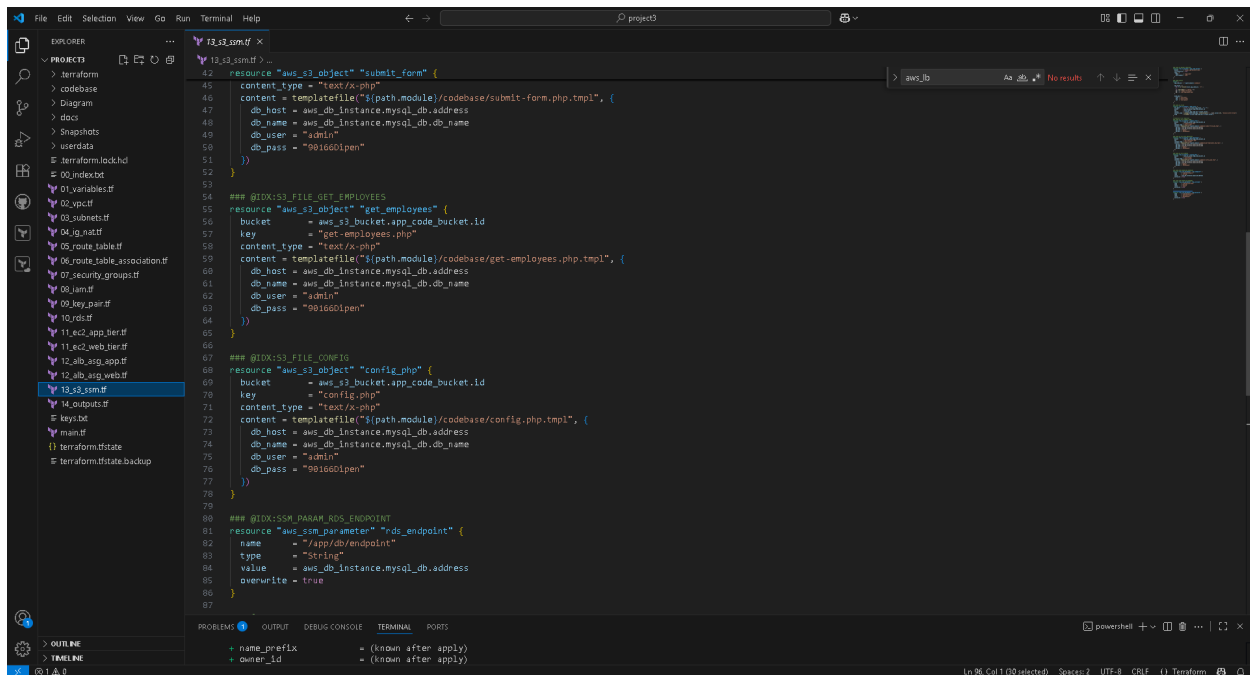


Phase 12: S3_SSM Implementation

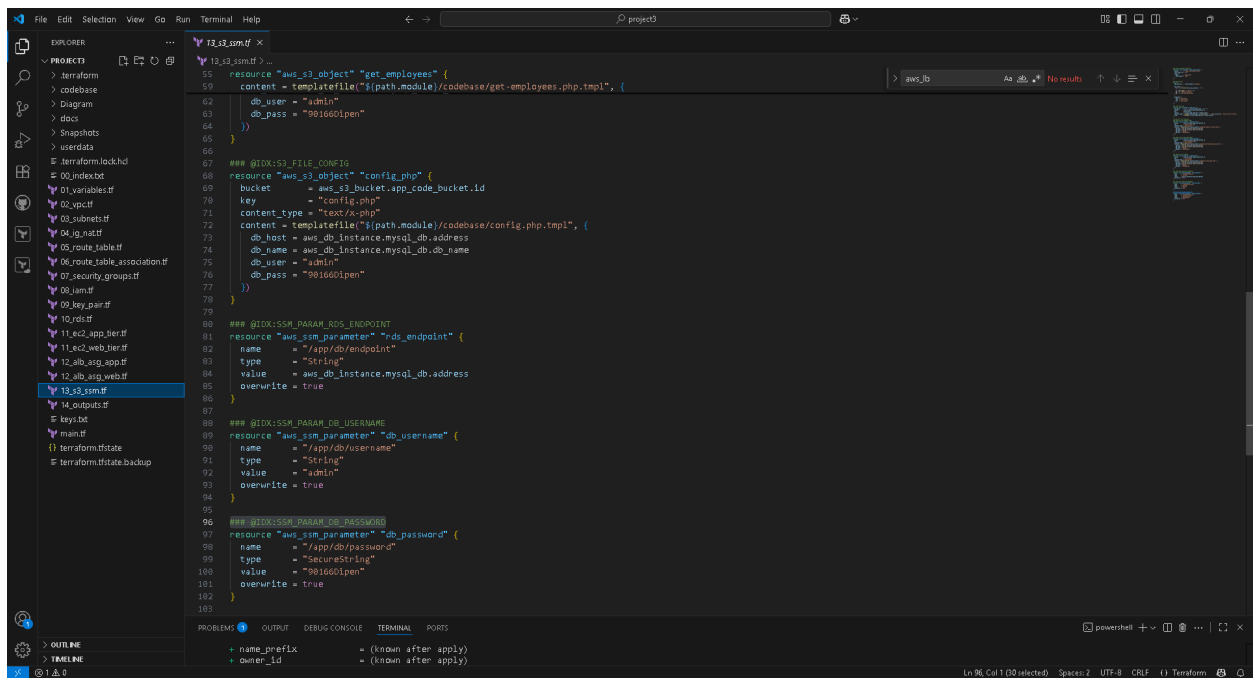
- **S3 Bucket:** Stores static assets and application code.
- **SSM Parameter Store:** Stores DB credentials and endpoint securely.
- EC2 instances retrieve configuration from Parameter Store at runtime.



```
1 resource "aws_s3_bucket" "app_code_bucket" {
2   bucket = "cdpen-app-backend-code"
3   force_destroy = true
4   tags = {
5     Name = "app-code"
6     Environment = "Dev"
7   }
8 }
9
10
11 ## @IDX:LOCALS
12 locals {
13   app_code_dir = "${path.module}/codebase"
14 }
15
16 app_files = [
17   for f in fileset(local.app_code_dir, "**") :
18   f
19   if !can(regex("\\.tmp$", f))
20   && f != "submit-form.php"
21   && f != "get-employees.php"
22 ]
23
24 mime_map = {
25   html = "text/html"
26   php = "text/x-php"
27   txt = "text/plain"
28 }
29
30 ## @IDX:S3_FILES
31 resource "aws_s3_object" "app_files" {
32   for_each = { for f in local.app_files : f => f }
33   bucket = aws_s3_bucket.app_code_bucket.id
34   key = each.value
35   source = "${local.app_code_dir}/${each.value}"
36   content_type = lookup(local.mime_map, reverse(split(".", each.value))[0], "binary/octet-stream")
37   etag = filemd5("${local.app_code_dir}/${each.value}")
38 }
39
40 # Rendered from templates
41 ## @IDX:S3_FILE_SUBMIT_FORM
42 resource "aws_s3_object" "submit_form" {
43   bucket = aws_s3_bucket.app_code_bucket.id
44   key = "submit-form.php"
45   content_type = "text/x-php"
46 }
```



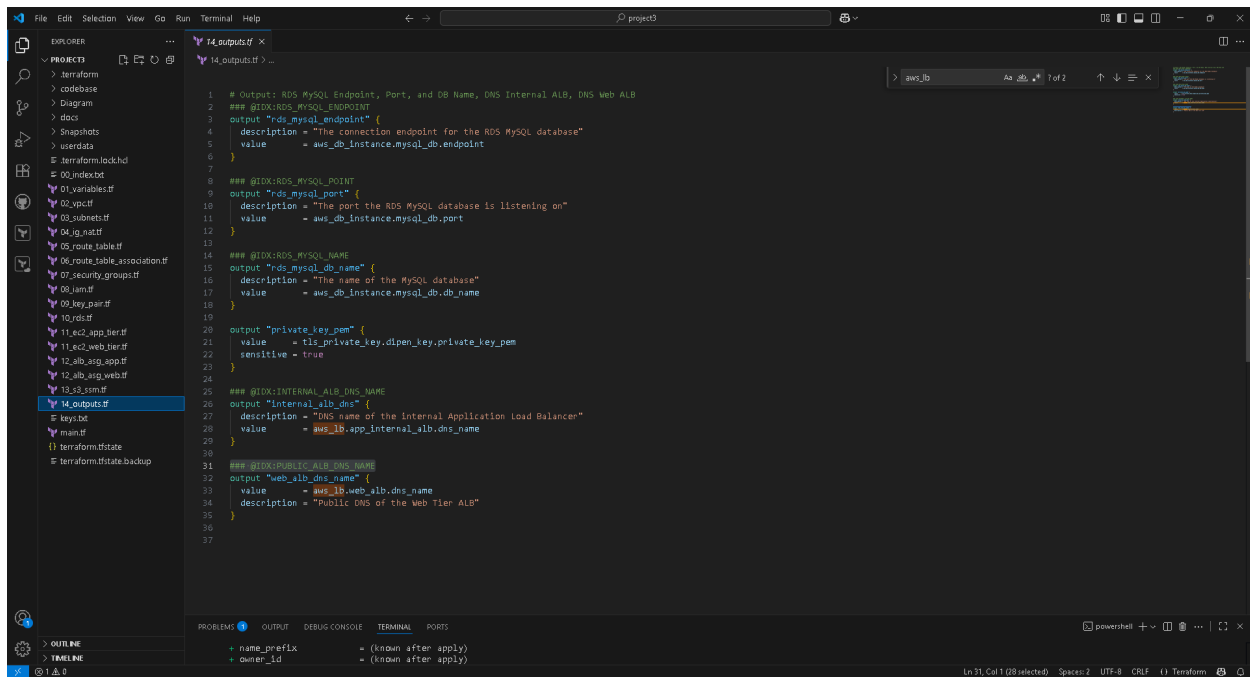
```
42 resource "aws_s3_object" "submit_form" {
43   content_type = "text/x-php"
44   content = templatefile("${path.module}/codebase/submit-form.php.tpl", {
45     db_host = aws_db_instance.mysql_db.address
46     db_name = aws_db_instance.mysql_db.db_name
47     db_user = "admin"
48     db_pass = "9916601pen"
49   })
50 }
51
52 ## @IDX:S3_FILE_GET_EMPLOYEES
53 resource "aws_s3_object" "get_employees" {
54   bucket = aws_s3_bucket.app_code_bucket.id
55   key = "get-employees.php"
56   content_type = "text/x-php"
57   content = templatefile("${path.module}/codebase/get-employees.php.tpl", {
58     db_host = aws_db_instance.mysql_db.address
59     db_name = aws_db_instance.mysql_db.db_name
60     db_user = "admin"
61     db_pass = "9916601pen"
62   })
63 }
64
65 ## @IDX:S3_FILE_CONFIG
66 resource "aws_s3_object" "config_php" {
67   bucket = aws_s3_bucket.app_code_bucket.id
68   key = "config.php"
69   content_type = "text/x-php"
70   content = templatefile("${path.module}/codebase/config.php.tpl", {
71     db_host = aws_db_instance.mysql_db.address
72     db_name = aws_db_instance.mysql_db.db_name
73     db_user = "admin"
74     db_pass = "9916601pen"
75   })
76 }
77
78 ## @IDX:SSM_PARAM_SQS_ENDPOINT
79 resource "aws_ssm_parameter" "rds_endpoint" {
80   name = "/app/db/endpoint"
81   type = "String"
82   value = aws_db_instance.mysql_db.address
83   overwrite = true
84 }
85
86
```



Phase 14: Outputs

Defined Terraform outputs:

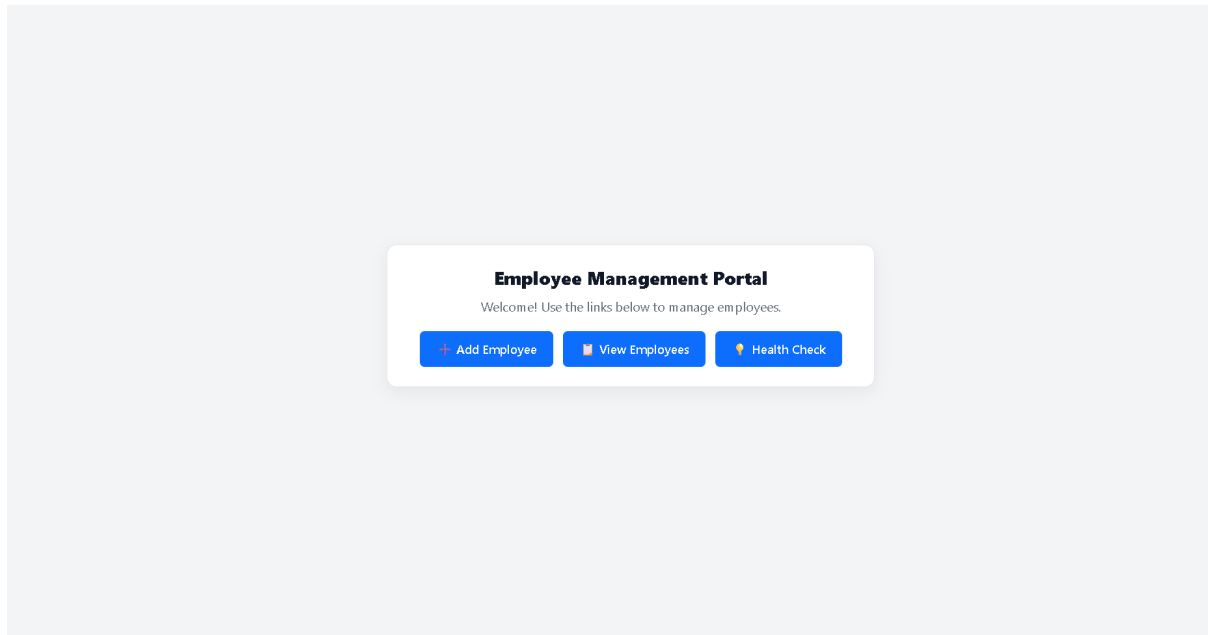
- Public ALB DNS.
- Internal ALB DNS.
- RDS MySQL endpoint.
- DB name.



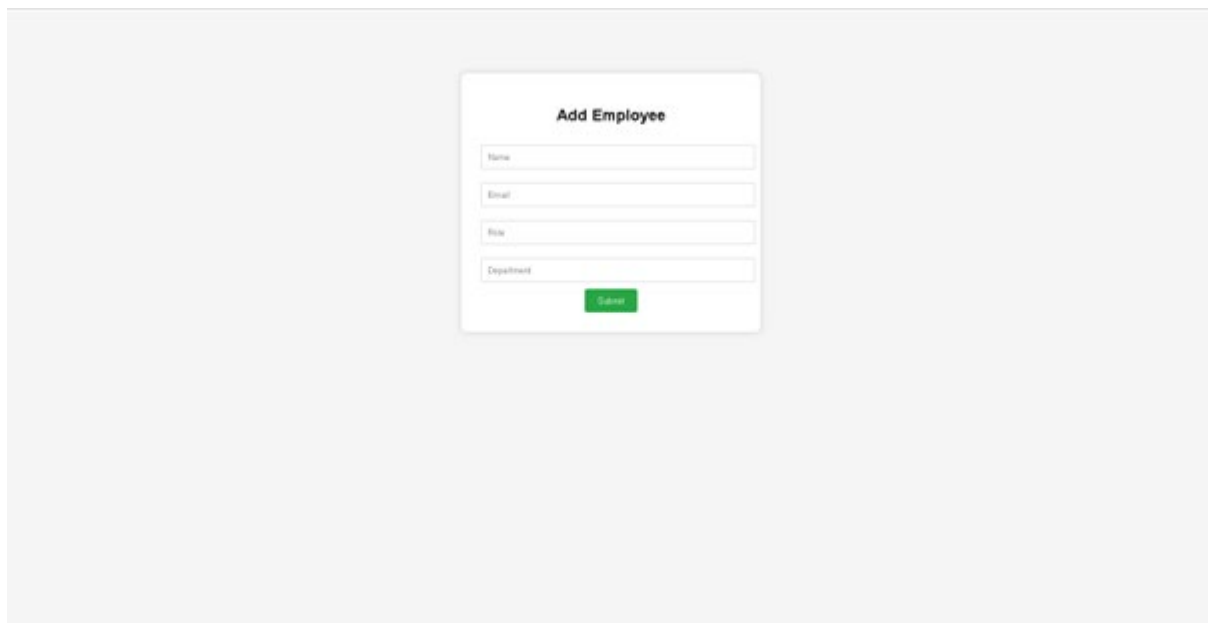
```
1 # Output: RDS MySQL Endpoint, Port, and DB Name, DNS Internal ALB, DNS Web ALB
2 ### @IDX:RDS_MYSQL_ENDPOINT
3 output "rds_mysql_endpoint" {
4   description = "The connection endpoint for the RDS MySQL database"
5   value       = aws_db_instance.mysql_db.endpoint
6 }
7
8 ### @IDX:RDS_MYSQL_PORT
9 output "rds_mysql_port" {
10  description = "The port the RDS MySQL database is listening on"
11  value       = aws_db_instance.mysql_db.port
12 }
13
14 ### @IDX:RDS_MYSQL_NAME
15 output "rds_mysql_db_name" {
16  description = "The name of the MySQL database"
17  value       = aws_db_instance.mysql_db.db_name
18 }
19
20 output "private_key_pem" {
21  value     = tls_private_key.dipen_key.private_key_pem
22  sensitive = true
23 }
24
25 ### @IDX:INTERNAL_ALB_DNS_NAME
26 output "internal_alb_dns" {
27  description = "DNS name of the Internal Application Load Balancer"
28  value       = aws_lb.app_internal_alb.dns_name
29 }
30
31 ### @IDX:PUBLIC_ALB_DNS_NAME
32 output "web_alb_dns_name" {
33  value     = aws_lb.web_alb.dns_name
34  description = "Public DNS of the Web Tier ALB"
35 }
36
37
```

User Interface

HomePage




Add Employee



The image shows the "Add Employee" form. It is a white card with a light gray border, centered on a light gray background. The card has the title "Add Employee" in bold. Below the title are four input fields: "Name", "Email", "Role", and "Department". At the bottom of the card is a green "Submit" button.

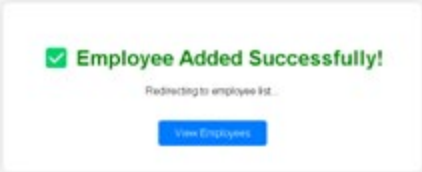
Data Input (form)




A form titled "Add Employee" with four input fields and a "Save" button. The fields are labeled "Name", "Email", "ID", and "Title". The "Name" field contains "John", the "Email" field contains "john@acme.com", and the "ID" field contains "101". The "Title" field is empty. The "Save" button is green.

Add Employee	
Name	John
Email	john@acme.com
ID	101
Title	
Save	

Form Submission



A success message box with a green checkmark icon, the text "Employee Added Successfully!", and a "View Employees" button. Below the message, it says "Redirecting to employee list...".

**Employee Added Successfully!**
Redirecting to employee list...
[View Employees](#)

View Employees (Data Fetch from RDS)

Employee Records					
ID	Name	Email	Role	Department	Created At
8	Vishnu	dipen@ex.com	RD	Sales	2025-07-23 05:36:25
7	abc	dipen@ex.com	Dev	IT	2025-07-23 05:34:26
6	Vishnu	vishnu@vishnu.com	RD	IT	2025-07-23 05:28:53
5	Vishnu	vishnu@vishnu.com	RD	IT	2025-07-23 05:28:50
4	John Doe	john@example.com	DevOps	Engineering	2025-07-23 05:28:37
1	Dipen Patel	dipen@example.com	Cloud Engineer	DevOps	2025-07-23 04:34:32
2	Aarav Shah	aarav-shah@example.com	Backend Developer	Engineering	2025-07-23 04:34:32
3	Mira Joshi	mira@example.com	Project Manager	Operations	2025-07-23 04:34:32