AWS Three-Tier Architecture Deployment with Terraform

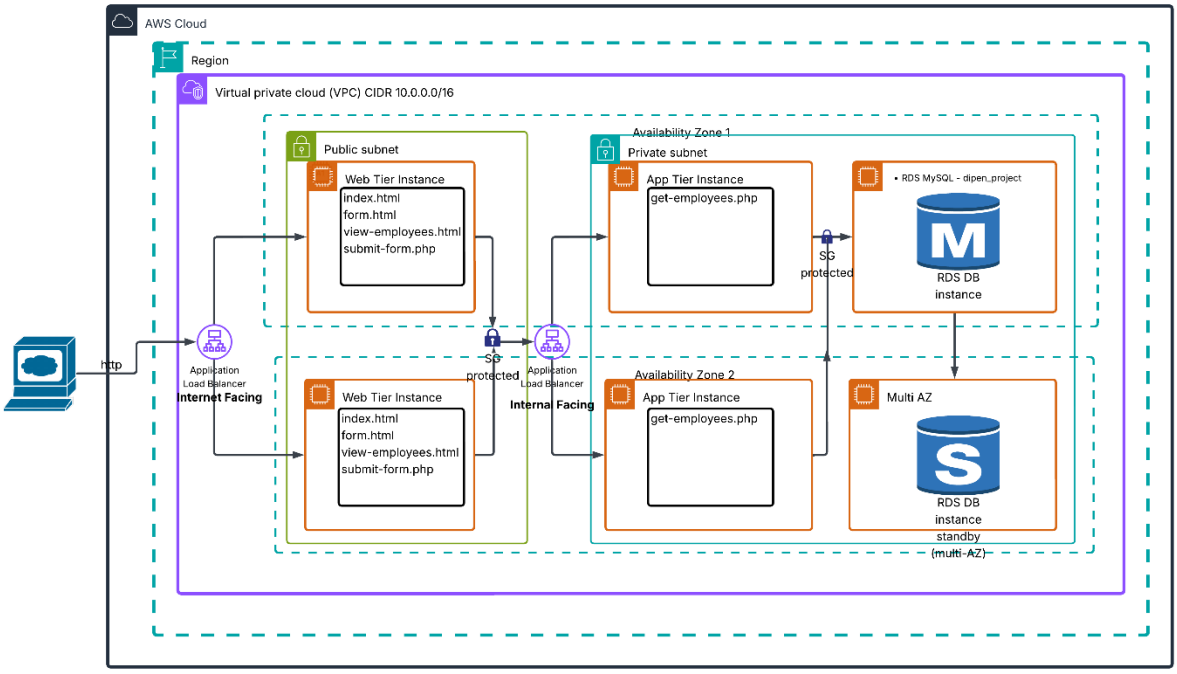
Architecture Layers:

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

```
# ions and Providers
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
```

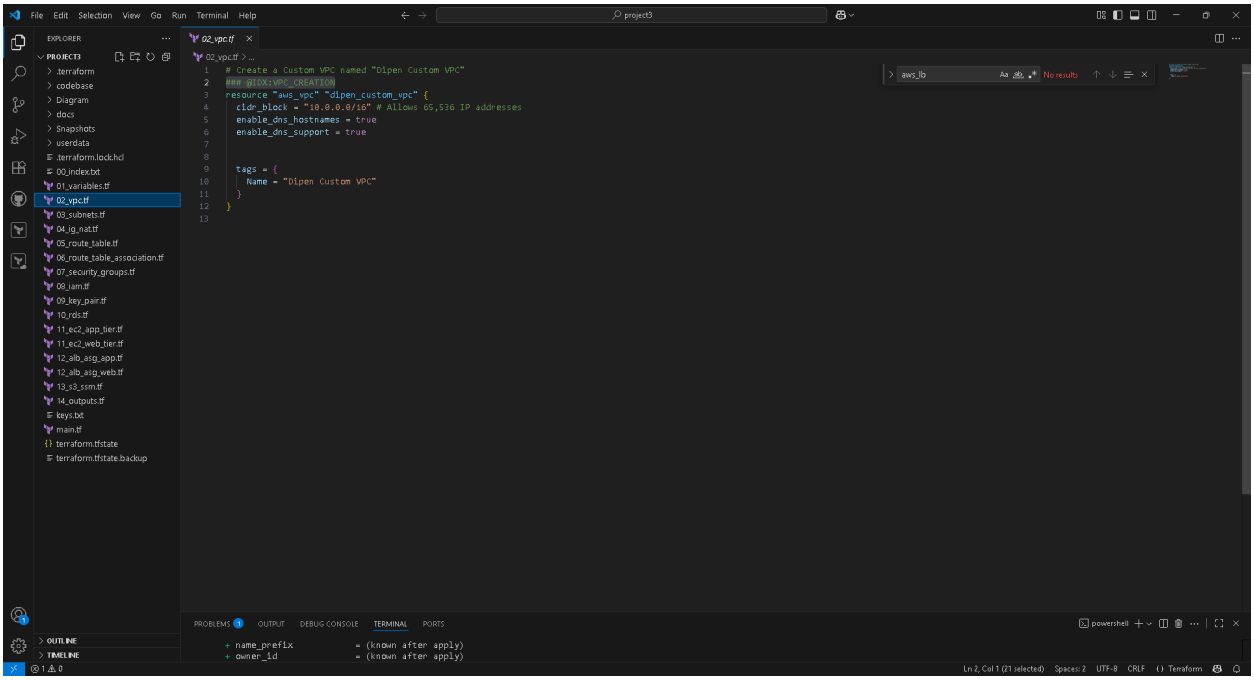
```
variable "mysql_config" {
  type    = tuple([string, number, bool])
  default = ["db.t4g.micro", 3, false]
}

### @IDX:VAR_REGION
variable "aws_region" {
  description = "AWS region for all resources"
  type        = string
  default     = "us-east-2"
}

### @IDX:VAR_WEB_AMI
variable "web_ami" {
  description = "AMI ID for Web Tier EC2"
  type        = string
  default     = "ami-08221e706f343d7b7"
}

### @IDX:VAR_WEB_INSTANCE
variable "web_instance_type" {
  description = "Instance type for Web Tier"
  type        = string
  default     = "t2.micro"
}

### @IDX:VAR_APP_AMI
variable "app_ami" {
  type    = string
  default = "ami-08221e706f343d7b7"
}

### @IDX:VAR_APP_INSTANCE
variable "app_instance_type" {
  description = "Instance type for APP Tier"
  type    = string
  default = "t2.micro"
}

variable "bucket_name" {
  type        = string
  default     = "dipen-app-backend-code"
  description = "S3 bucket with web static files"
}
```

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

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:

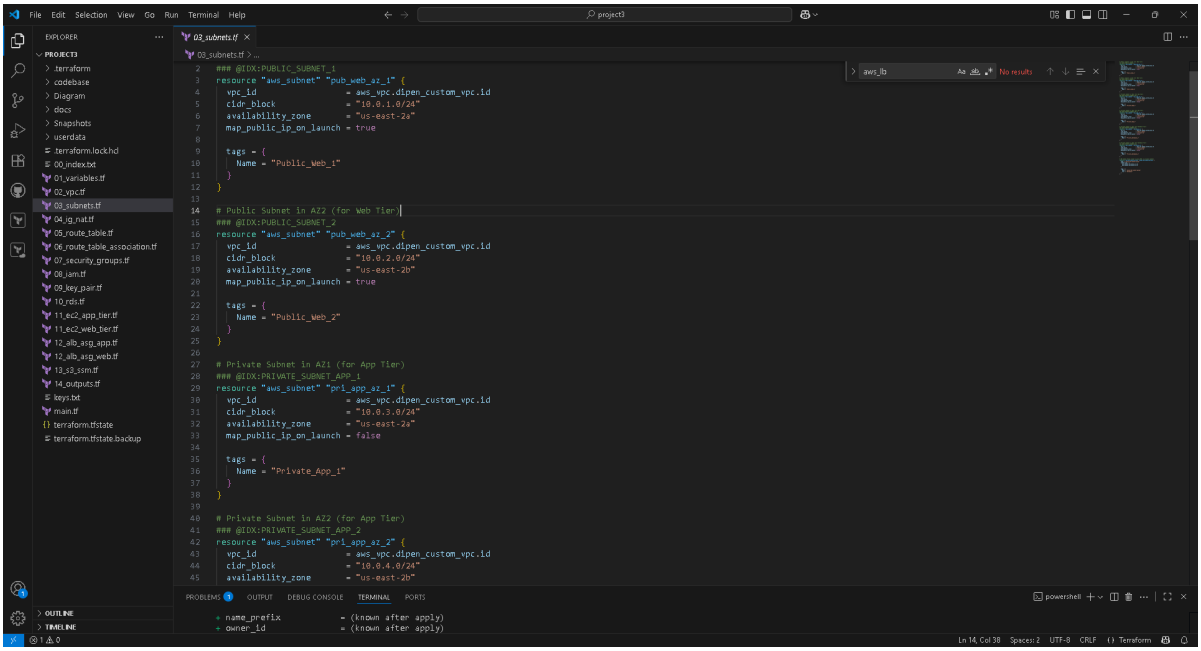
- Logical network isolation.
- Custom IP addressing.
- Ability to define routing and security at the network level.

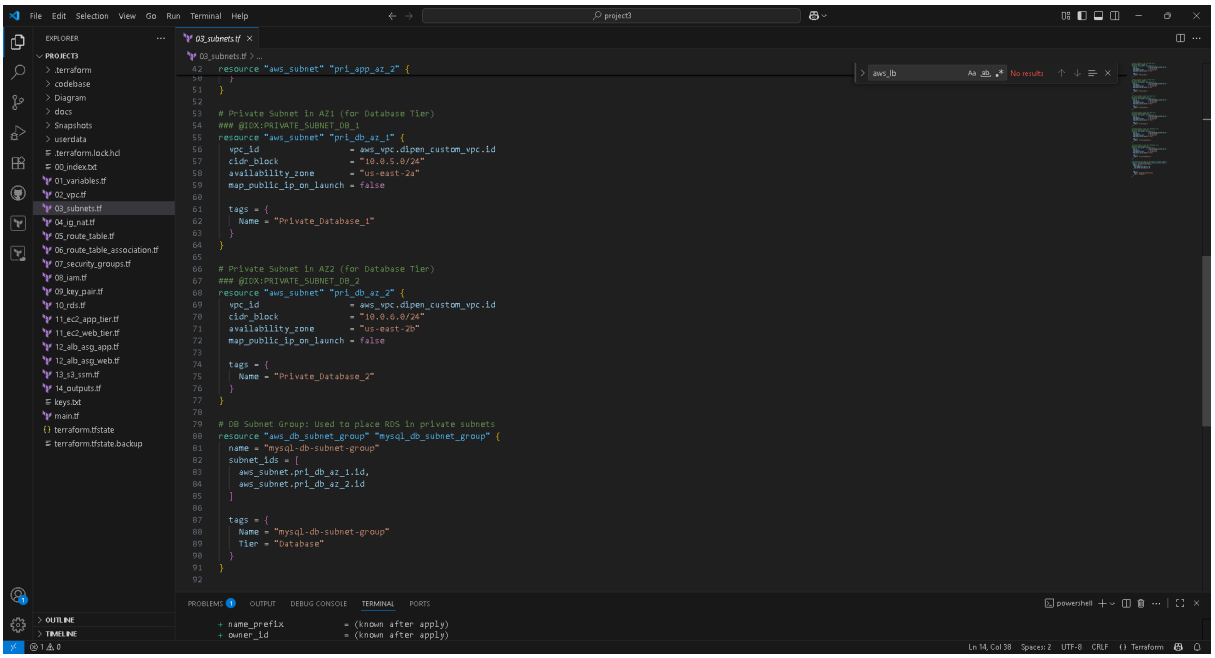


Phase 3: Subnet Creation

Created six subnets distributed across two Availability Zones:

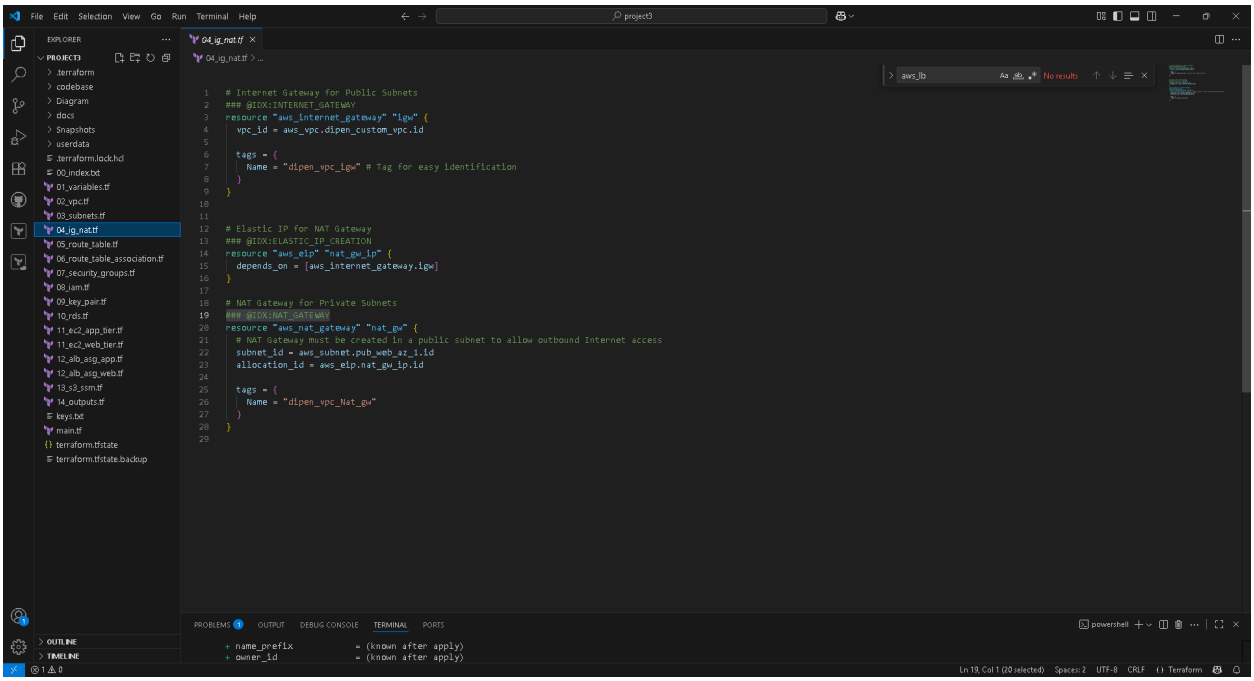
- **Public Subnets (Web Tier)**: Host public-facing ALB and Web EC2 instances.
- **Private Subnets (App Tier)**: Host internal ALB and App EC2 instances.
- **Private Subnets (DB Tier)**: Host RDS MySQL instance.
  Each tier is isolated to enhance security and fault tolerance.





Phase 4: Internet Gateway and NAT Gateway

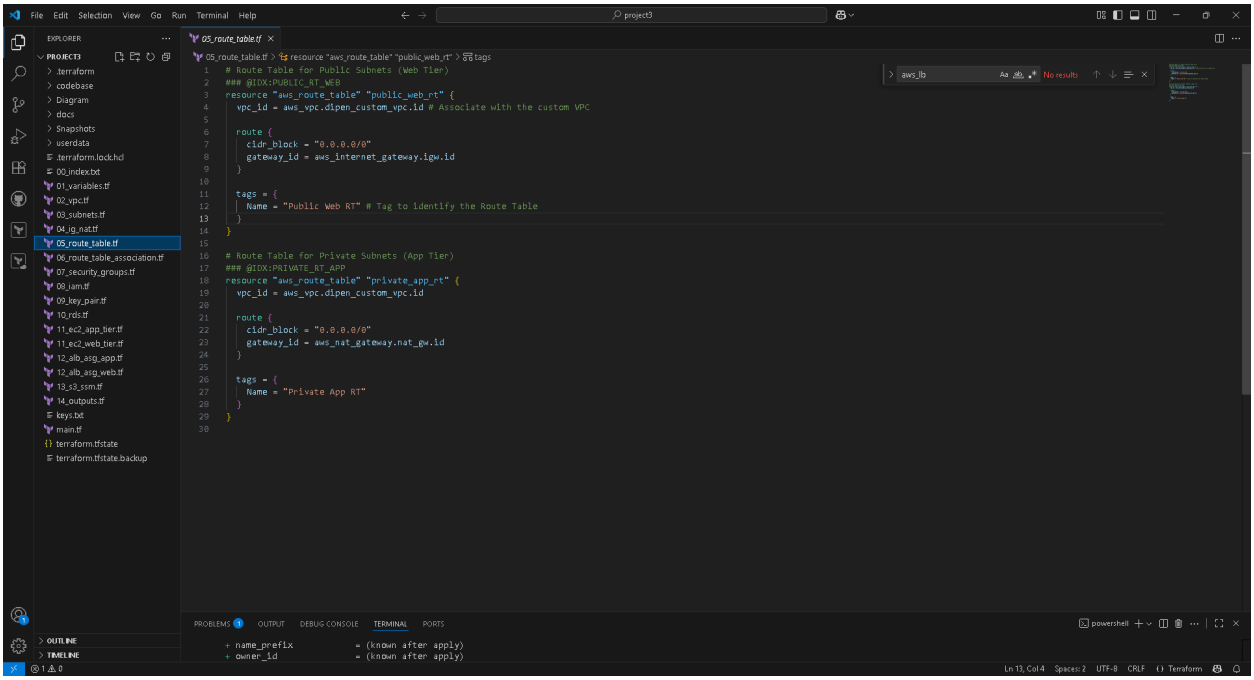
- **Internet Gateway (IGW)**: Allows resources in public subnets to access the internet.
- **NAT Gateway**: Enables private subnet instances (App/DB tiers) to access the internet for updates without being directly exposed.



Phase 5: Route Table Creation

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

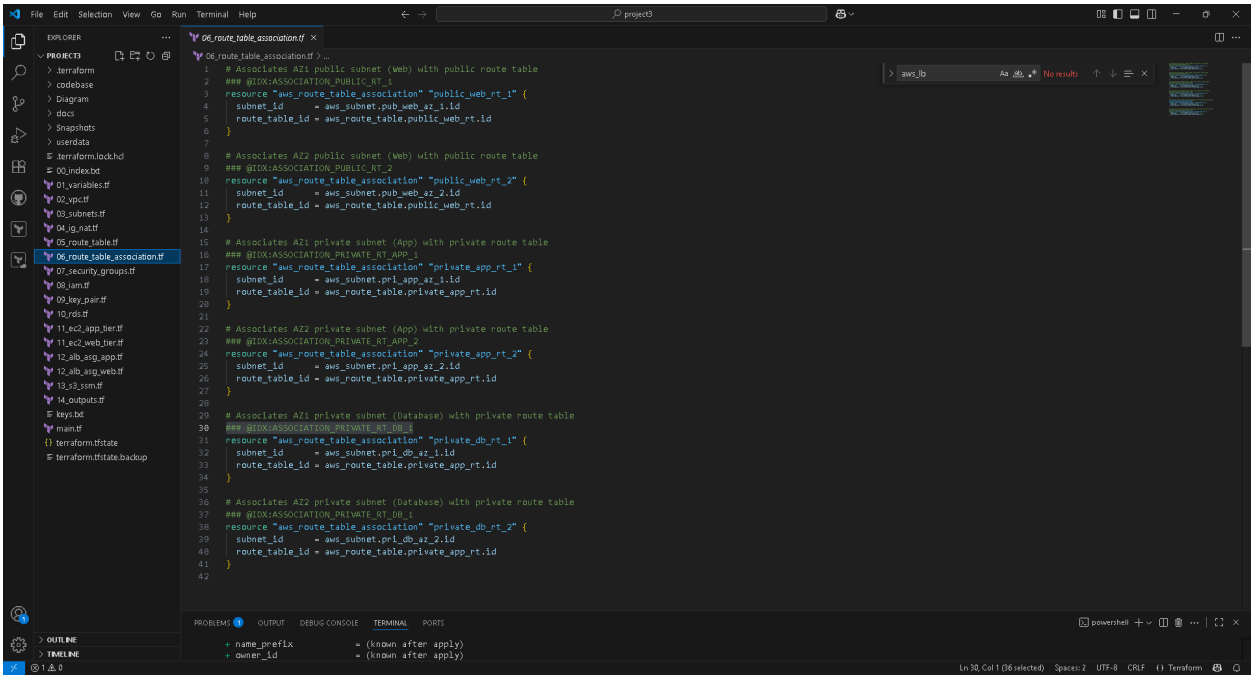
Private Route Tables: Routes outbound traffic from private subnets through NAT Gateway. This ensures correct routing for each tier.



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.

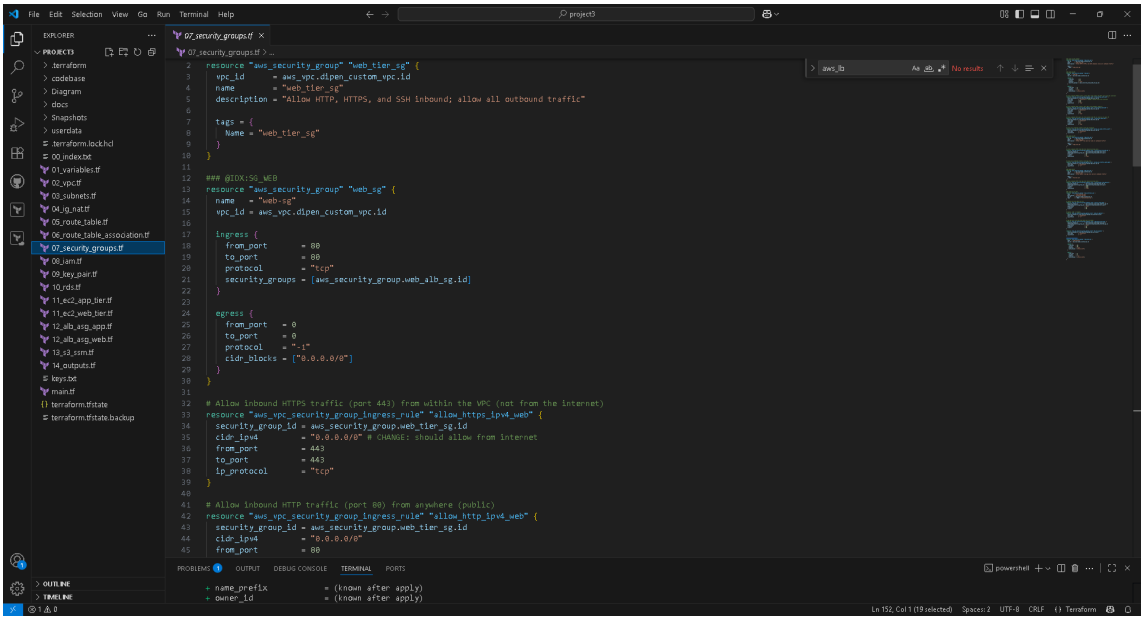


Phase 7: Security Groups

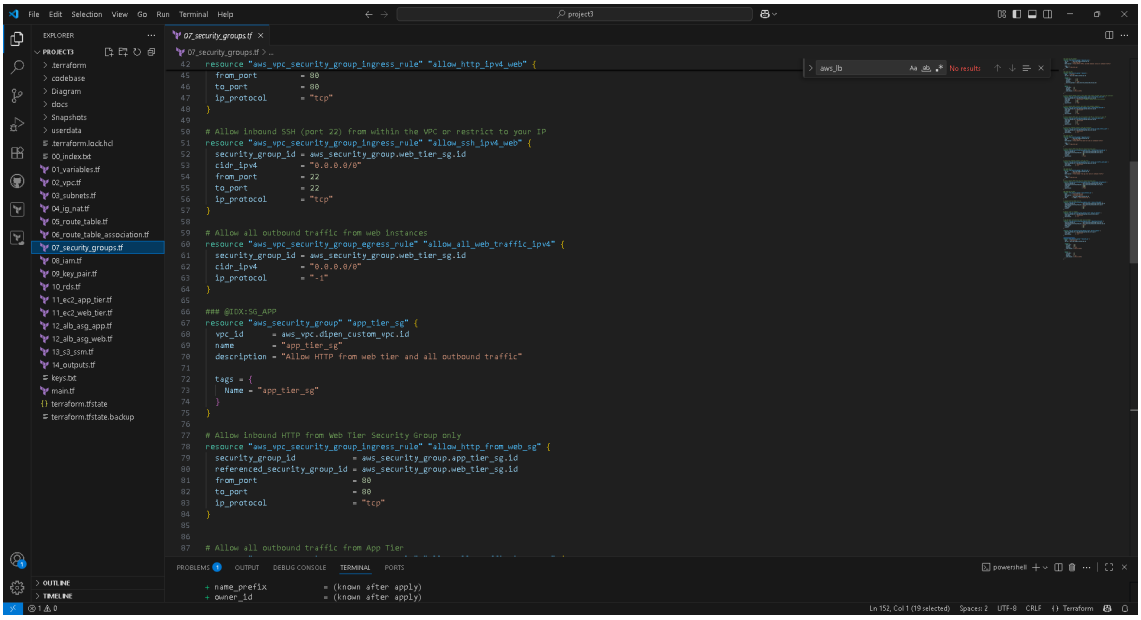
Created tier-specific Security Groups:

- **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"
  }
}

### @IDX: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" # CHANGE: 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
```



```
resource "aws_vpc_security_group_ingress_rule" "allow_http_ipv4_web" {
  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"
}

### @IDX: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
```

```tf
}


# Allow all outbound traffic from App Tier
resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4_app" {
  security_group_id = aws_security_group.app_tier_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1"
}

### @IDX:SG_DB
resource "aws_security_group" "db_tier_sg" {
  vpc_id      = aws_vpc.dipen_custom_vpc.id
  name        = "db_tier_sg"
  description = "Allow mysql from app tier and all outbound traffic"

  tags = {
    Name = "db_tier_sg"
  }
}
# Allow inbound HTTP from App Tier Security Group only
resource "aws_vpc_security_group_ingress_rule" "allow_mysql_from_app_sg" {
  security_group_id            = aws_security_group.db_tier_sg.id
  referenced_security_group_id = aws_security_group.app_tier_sg.id
  from_port                    = 3306
  to_port                      = 3306
  ip_protocol                  = "tcp"
}

### @IDX:SG_ALB_APP
resource "aws_security_group" "alb_internal_sg" {
  name        = "alb-internal-sg"
  description = "Internal ALB SG"
  vpc_id      = aws_vpc.dipen_custom_vpc.id
  tags        = { Name = "alb-internal-sg" }
}

# Ingress 80 from Web Tier SG
resource "aws_vpc_security_group_ingress_rule" "alb_in_from_web" {
  security_group_id            = aws_security_group.alb_internal_sg.id
  referenced_security_group_id = aws_security_group.web_tier_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 80
  to_port                      = 80
```

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

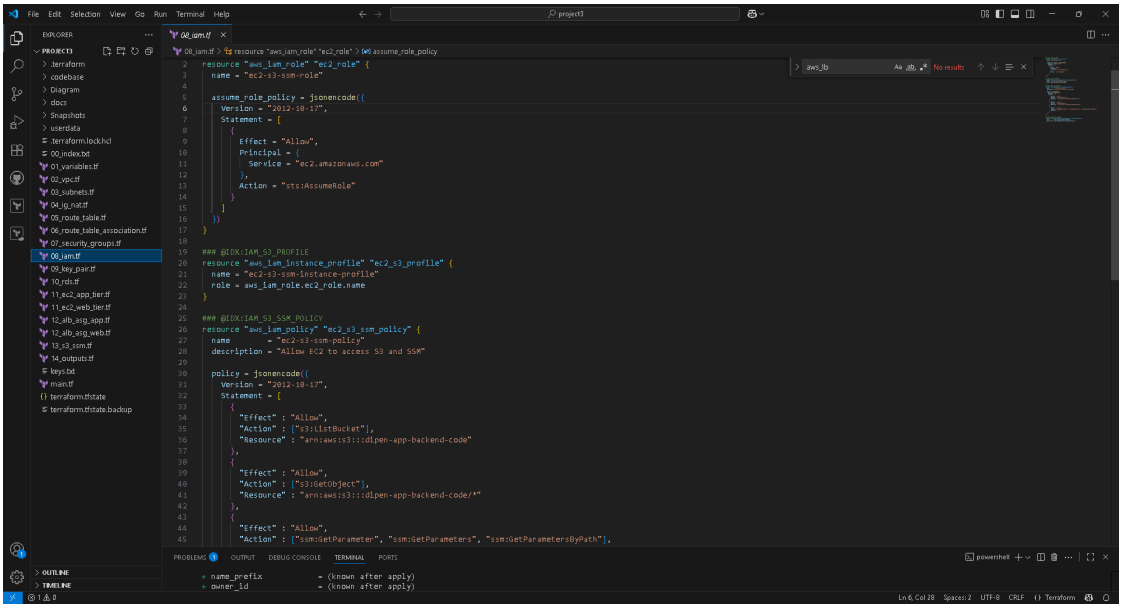
```tf
}

# Egress: ALB -> targets
resource "aws_vpc_security_group_egress_rule" "alb_all_egress" {
  security_group_id = aws_security_group.alb_internal_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1"
}

# App SG must allow 80 from ALB (not from Web)
resource "aws_vpc_security_group_ingress_rule" "app_http_from_alb" {
  security_group_id            = aws_security_group.app_tier_sg.id
  referenced_security_group_id = aws_security_group.alb_internal_sg.id
  ip_protocol                  = "tcp"
  from_port                    = 80
  to_port                      = 80
}

resource "aws_vpc_security_group_egress_rule" "db_all_egress" {
  security_group_id = aws_security_group.db_tier_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1"
}

### @IDX:SG_ALB_WEB
resource "aws_security_group" "web_alb_sg" {
  name   = "web-alb-sg"
  vpc_id = aws_vpc.dipen_custom_vpc.id

  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"]
  }
}
```

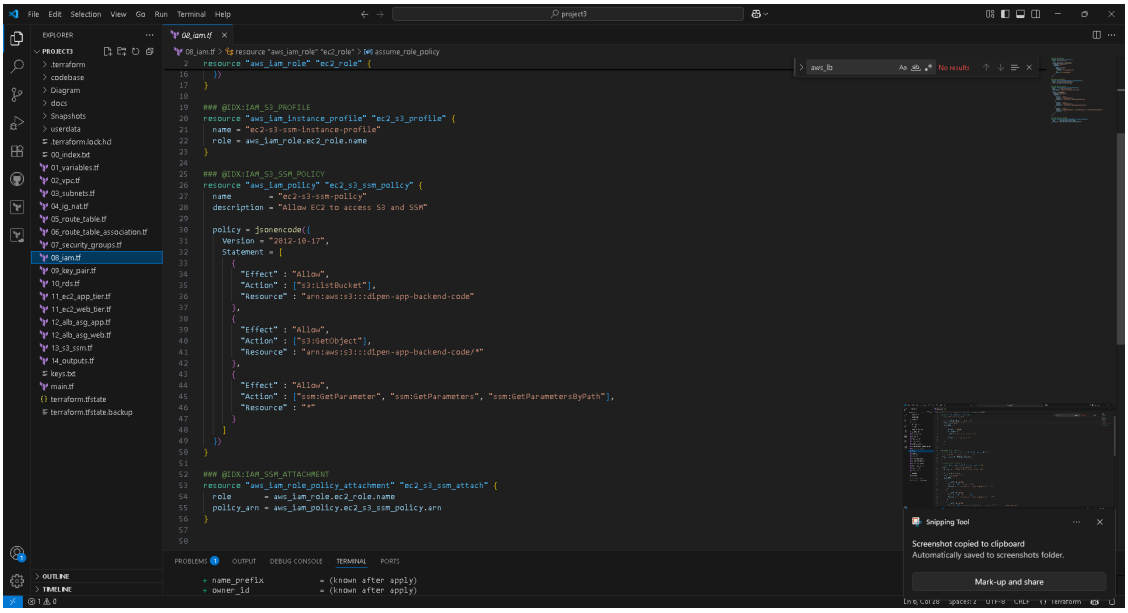
```
  + name_prefix             = (known after apply)
  + owner_id                = (known after apply)
```

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.

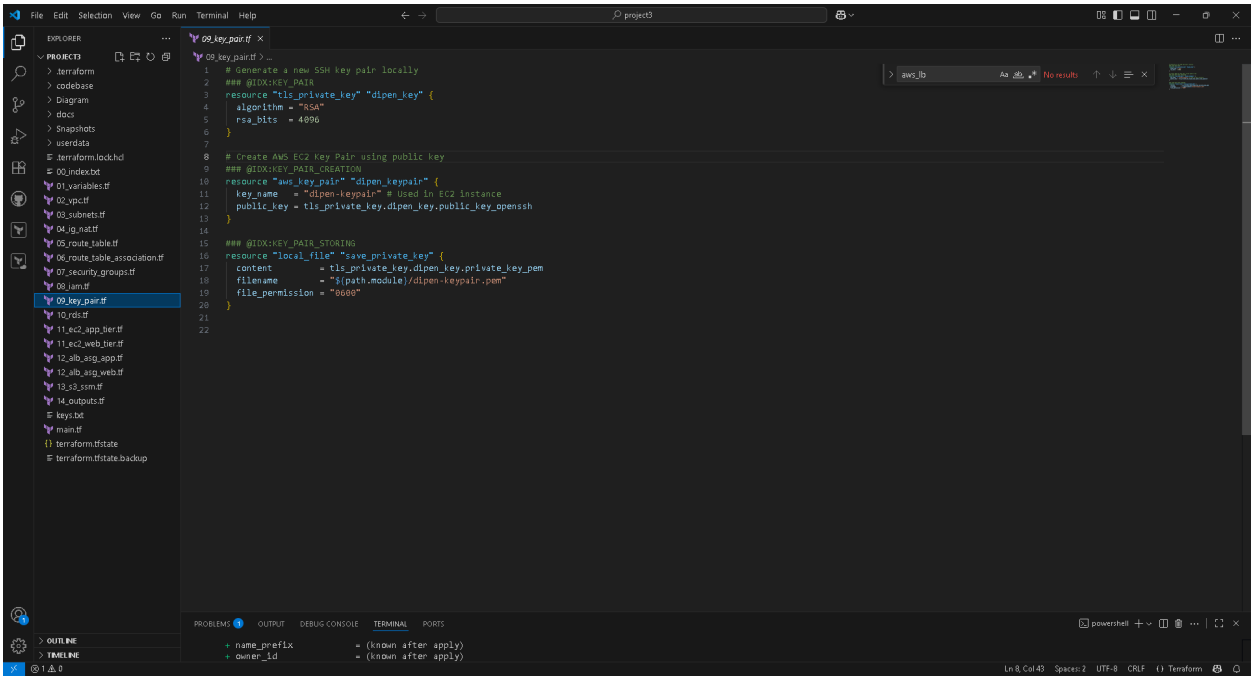




Phase 9: Key Pair Creation and Storing

Generated AWS Key Pair for SSH access:

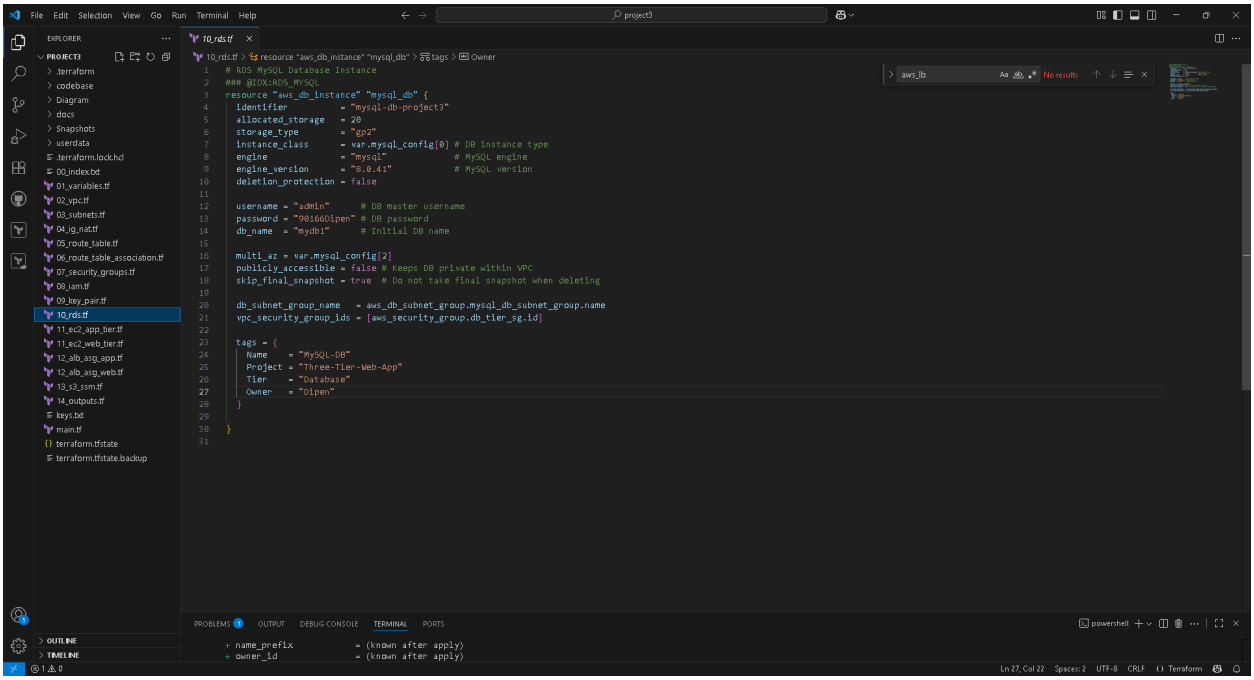
- Stored .pem file securely.
- Associated with EC2 instances for secure administrative access.



Phase 10: RDS Creation

Deployed **Amazon RDS MySQL** instance:

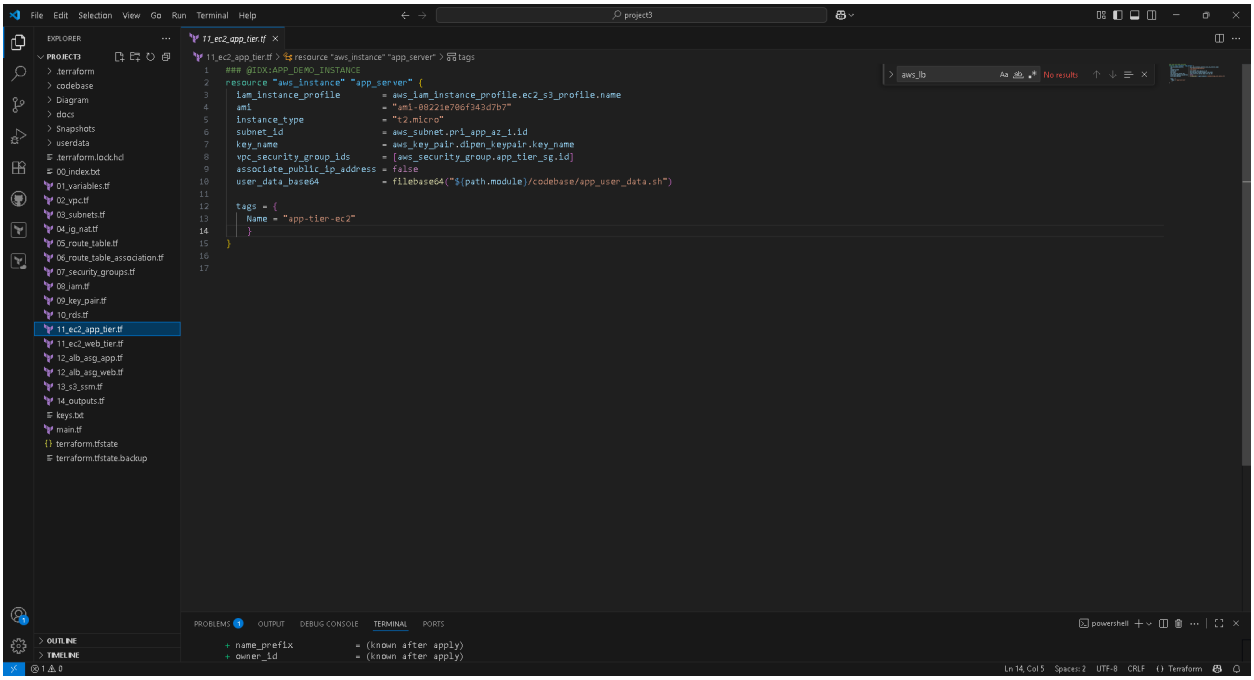
- Multi-AZ deployment in private DB subnets.
- Security Group restricts access to App Tier only.

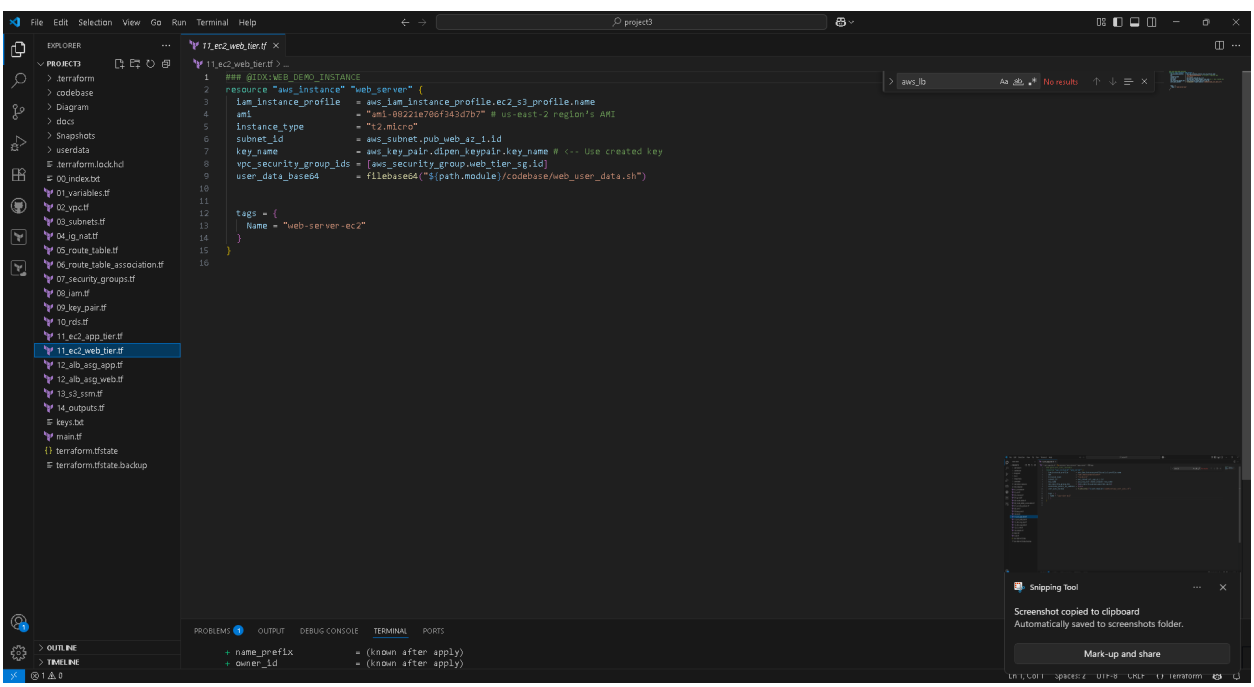


Phase 11: Test Instance Creation

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.

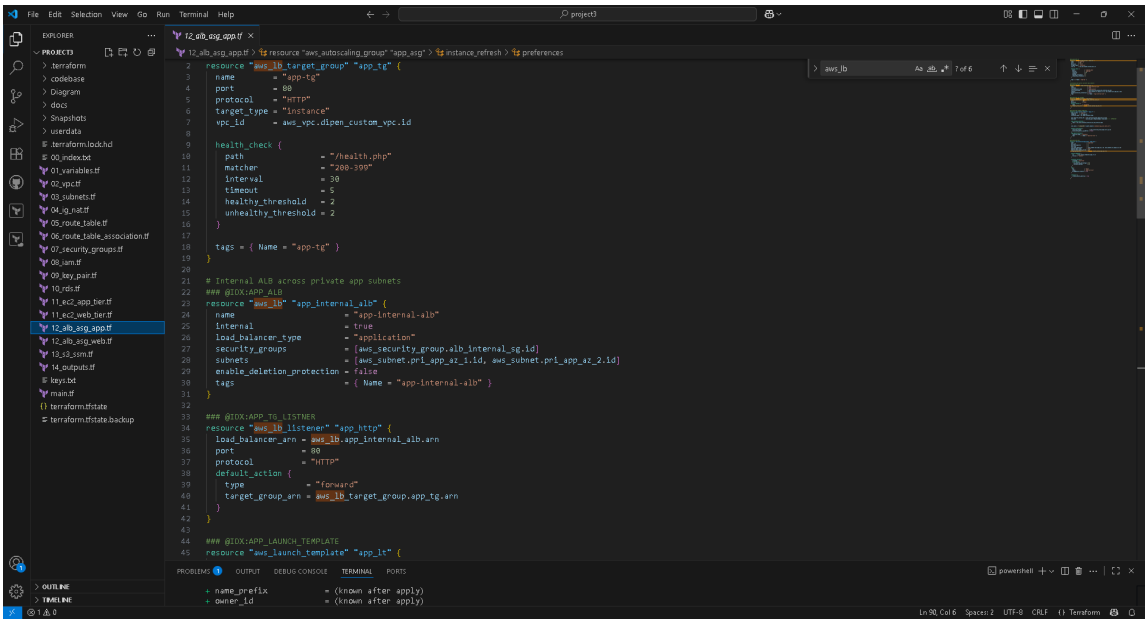


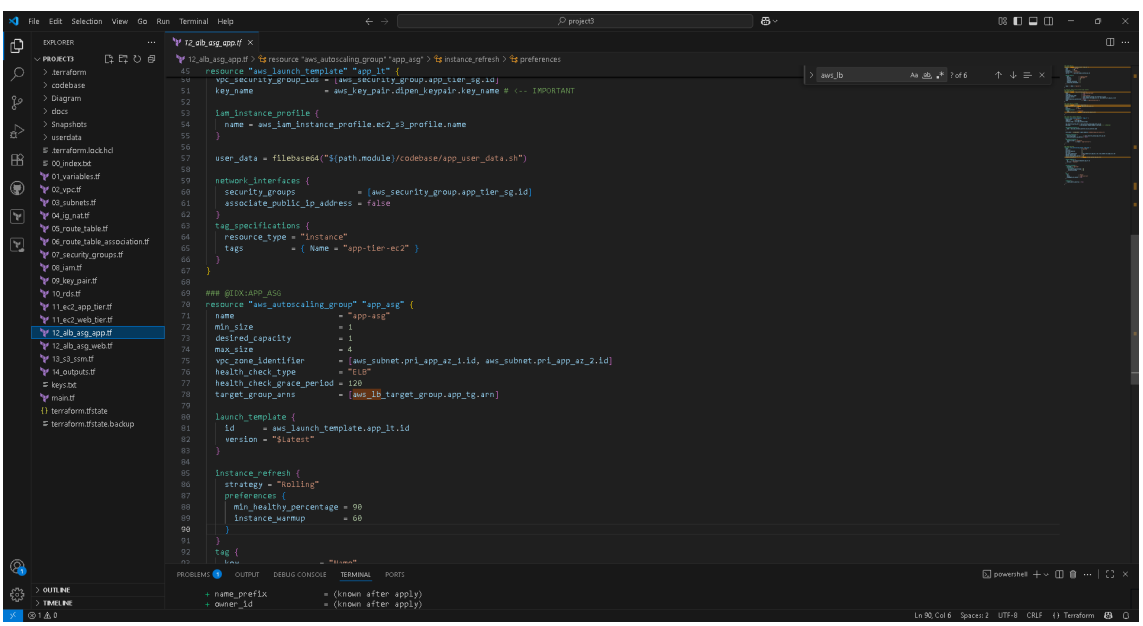


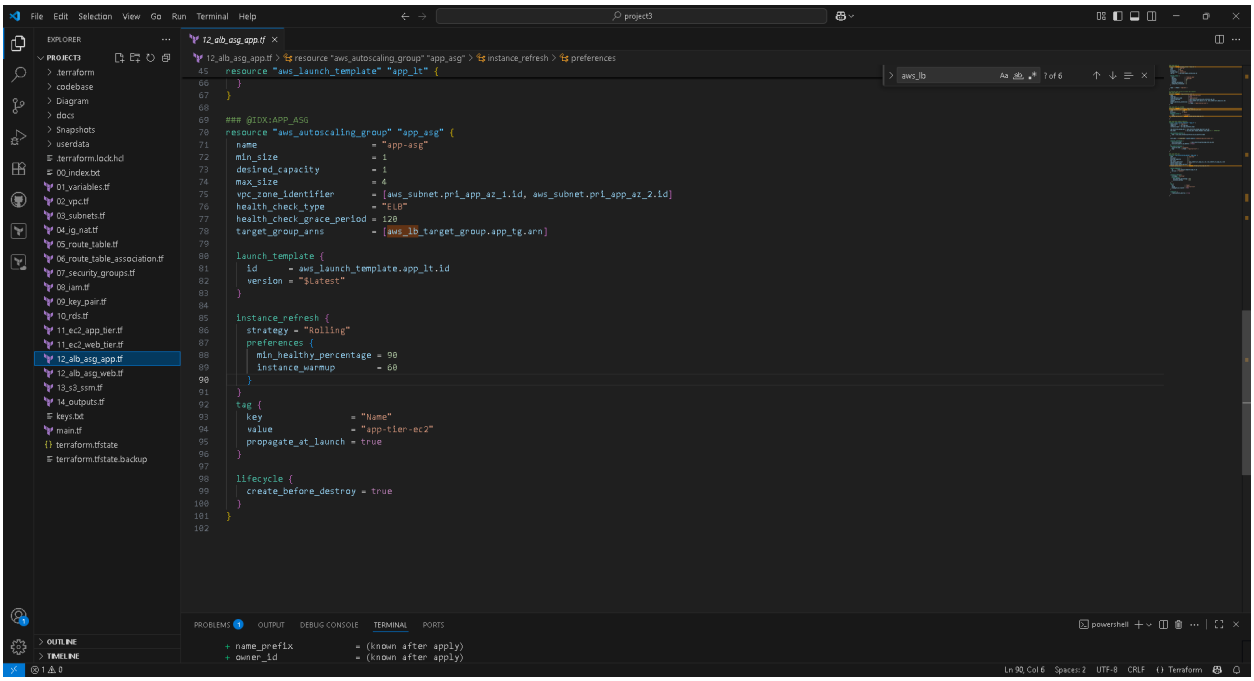
Phase 11: Application Load Balancer, Target Groups, Auto Scaling Group

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.

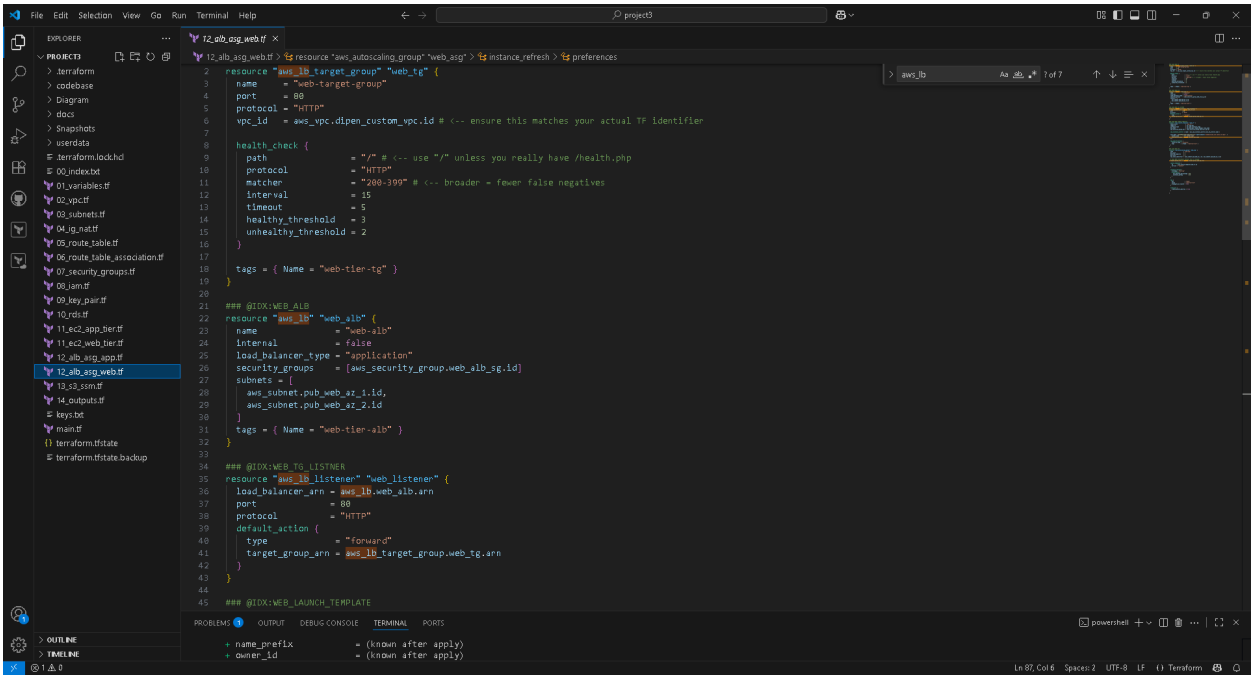




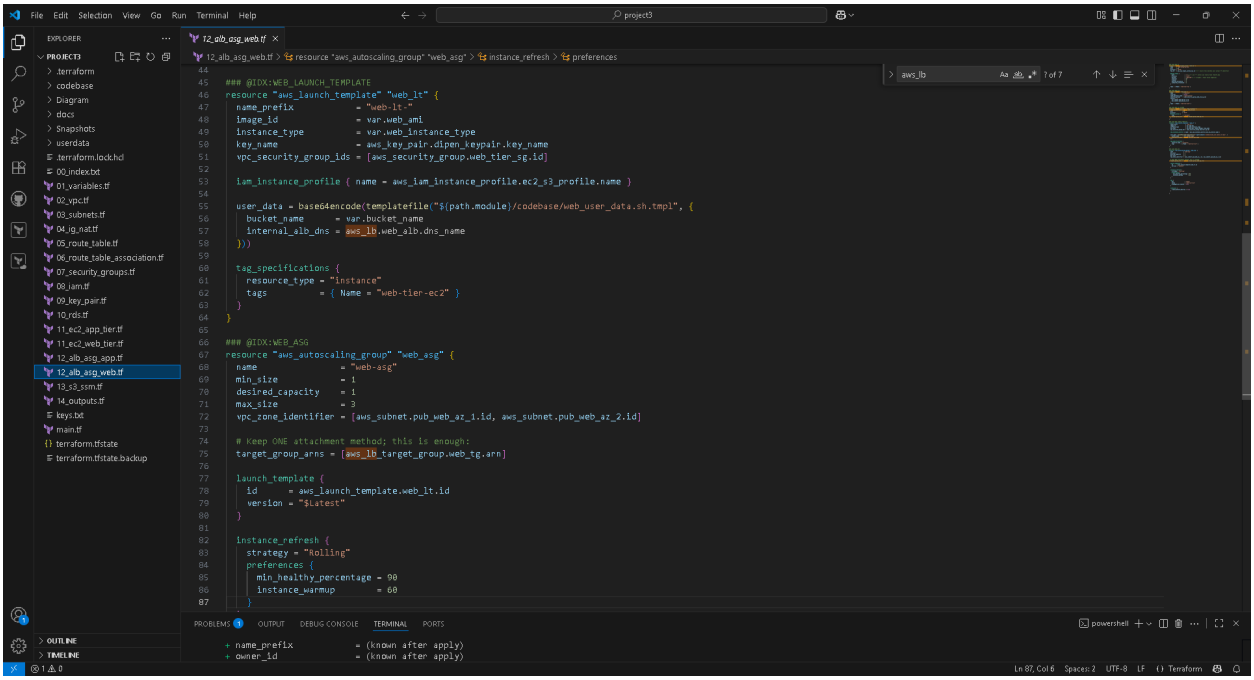

File Edit Selection View Go Run Terminal Help
project3
EXPLORER
PROJECT3
.terraform
codebase
Diagram
docs
Snapshots
userdata
.terraform.lock.hcl
00_index.txt
01_variables.tf
02_vpc.tf
03_subnets.tf
04_ig_nat.tf
05_route_table.tf
06_route_table_association.tf
07_security_groups.tf
08_iam.tf
09_key_pair.tf
10_rds.tf
11_ec2_app_tier.tf
11_ec2_web_tier.tf
12_alb_asg_app.tf
12_alb_asg_web.tf
13_s3_ssm.tf
14_outputs.tf
keys.txt
main.tf
terraform.tfstate
terraform.tfstate.backup
OUTLINE
TIMELINE
12_alb_asg_app.tf
12_alb_asg_app.tf > resource "aws_autoscaling_group" "app_asg" > instance_refresh > preferences
aws_lb
7 of 6
45 resource "aws_launch_template" "app_lt" {
66 }
67 }
68
69 ### @IDX:APP_ASG
70 resource "aws_autoscaling_group" "app_asg" {
71 name = "app-asg"
72 min_size = 1
73 desired_capacity = 1
74 max_size = 4
75 vpc_zone_identifier = [aws_subnet.pri_app_az_1.id, aws_subnet.pri_app_az_2.id]
76 health_check_type = "ELB"
77 health_check_grace_period = 120
78 target_group_arns = [aws_lb_target_group.app_tg.arn]
79
80 launch_template {
81 id = aws_launch_template.app_lt.id
82 version = "$Latest"
83 }
84
85 instance_refresh {
86 strategy = "Rolling"
87 preferences {
88 min_healthy_percentage = 90
89 instance_warmup = 60
90 }
91 }
92 tag {
93 key = "Name"
94 value = "app-tier-ec2"
95 propagate_at_launch = true
96 }
97
98 lifecycle {
99 create_before_destroy = true
100 }
101 }
102
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
+ name_prefix = (known after apply)
+ owner_id = (known after apply)
Ln 90, Col 6 Spaces: 2 UTF-8 CRLF {} Terraform

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.



```tf
resource "aws_lb_target_group" "web_tg" {
  name     = "web-target-group"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.dlpen_custom_vpc.id # <-- ensure this matches your actual TF identifier

  health_check {
    path                = "/" # <-- use "/" unless you really have /health.php
    protocol            = "HTTP"
    matcher             = "200-399" # <-- broader = fewer false negatives
    interval            = 15
    timeout             = 5
    healthy_threshold   = 3
    unhealthy_threshold = 2
  }

  tags = { Name = "web-tier-tg" }
}

### @IDX:WEB_ALB
resource "aws_lb" "web_alb" {
  name               = "web-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.web_alb_sg.id]
  subnets = [
    aws_subnet.pub_web_az_1.id,
    aws_subnet.pub_web_az_2.id
  ]
  tags = { Name = "web-tier-alb" }
}

### @IDX:WEB_TG_LISTNER
resource "aws_lb_listener" "web_listener" {
  load_balancer_arn = aws_lb.web_alb.arn
  port              = 80
  protocol          = "HTTP"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.web_tg.arn
  }
}

### @IDX:WEB_LAUNCH_TEMPLATE
```



```tf
### @IDX:WEB_LAUNCH_TEMPLATE
resource "aws_launch_template" "web_lt" {
  name_prefix            = "web-lt-"
  image_id               = var.web_ami
  instance_type          = var.web_instance_type
  key_name               = aws_key_pair.dlpen_keypair.key_name
  vpc_security_group_ids = [aws_security_group.web_tier_sg.id]

  iam_instance_profile { name = aws_iam_instance_profile.ec2_s3_profile.name }

  user_data = base64encode(templatefile("${path.module}/codebase/web_user_data.sh.tmpl", {
    bucket_name      = var.bucket_name
    internal_alb_dns = aws_lb.web_alb.dns_name
  }))

  tag_specifications {
    resource_type = "instance"
    tags          = { Name = "web-tier-ec2" }
  }
}

### @IDX:WEB_ASG
resource "aws_autoscaling_group" "web_asg" {
  name                  = "web-asg"
  min_size              = 1
  desired_capacity      = 1
  max_size              = 3
  vpc_zone_identifier = [aws_subnet.pub_web_az_1.id, aws_subnet.pub_web_az_2.id]

  # Keep ONE attachment method; this is enough:
  target_group_arns = [aws_lb_target_group.web_tg.arn]

  launch_template {
    id      = aws_launch_template.web_lt.id
    version = "$Latest"
  }

  instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 90
      instance_warmup        = 60
    }
```

File Edit Selection View Go Run Terminal Help

project3

EXPLORER

PROJECT3

- .terraform
- codebase
- Diagram
- docs
- Snapshots
- userdata
- .terraform.lock.hcl
- 00_index.txt
- 01_variables.tf
- 02_vpc.tf
- 03_subnets.tf
- 04_ig_nat.tf
- 05_route_table.tf
- 06_route_table_association.tf
- 07_security_groups.tf
- 08_iam.tf
- 09_key_pair.tf
- 10_rds.tf
- 11_ec2_app_tier.tf
- 11_ec2_web_tier.tf
- 12_alb_asg_app.tf
- 12_alb_asg_web.tf
- 13_s3_ssm.tf
- 14_outputs.tf
- keys.txt
- main.tf
- terraform.tfstate
- terraform.tfstate.backup

OUTLINE

TIMELINE

12_alb_asg_web.tf > resource "aws_autoscaling_group" "web_asg" > instance_refresh > preferences

aws_lb ? of 7

```tf
resource "aws_launch_template" "web_lt" {
  tag_specifications {
    tags          = { Name = "web-tier-ec2" }
  }
}

### @IDX:WEB_ASG
resource "aws_autoscaling_group" "web_asg" {
  name                = "web-asg"
  min_size            = 1
  desired_capacity    = 1
  max_size            = 3
  vpc_zone_identifier = [aws_subnet.pub_web_az_1.id, aws_subnet.pub_web_az_2.id]

  # Keep ONE attachment method; this is enough:
  target_group_arns = [aws_lb_target_group.web_tg.arn]

  launch_template {
    id      = aws_launch_template.web_lt.id
    version = "$Latest"
  }

  instance_refresh {
    strategy = "Rolling"
    preferences {
      min_healthy_percentage = 90
      instance_warmup        = 60
    }
  }

  tag {
    key                 = "Name"
    value               = "web-tier-ec2"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
  }

}
```

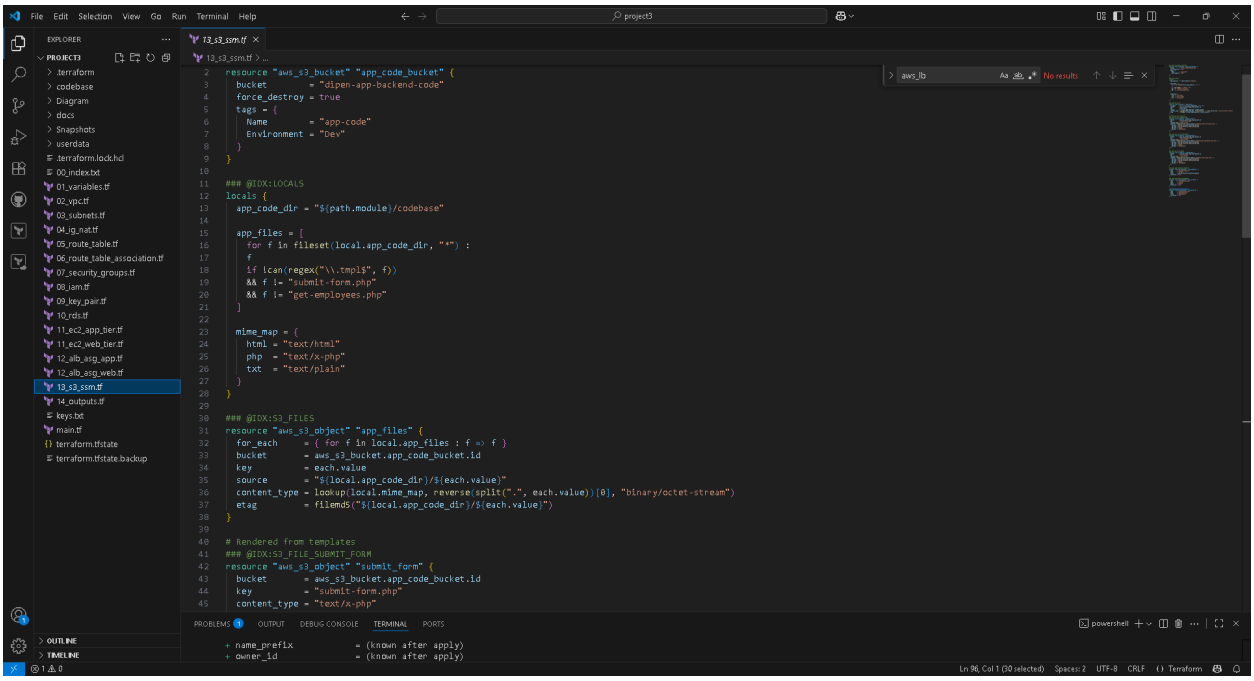
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
+ name_prefix            = (known after apply)
+ Owner_id               = (known after apply)
```

Ln 87, Col 6 Spaces: 2 UTF-8 LF Terraform

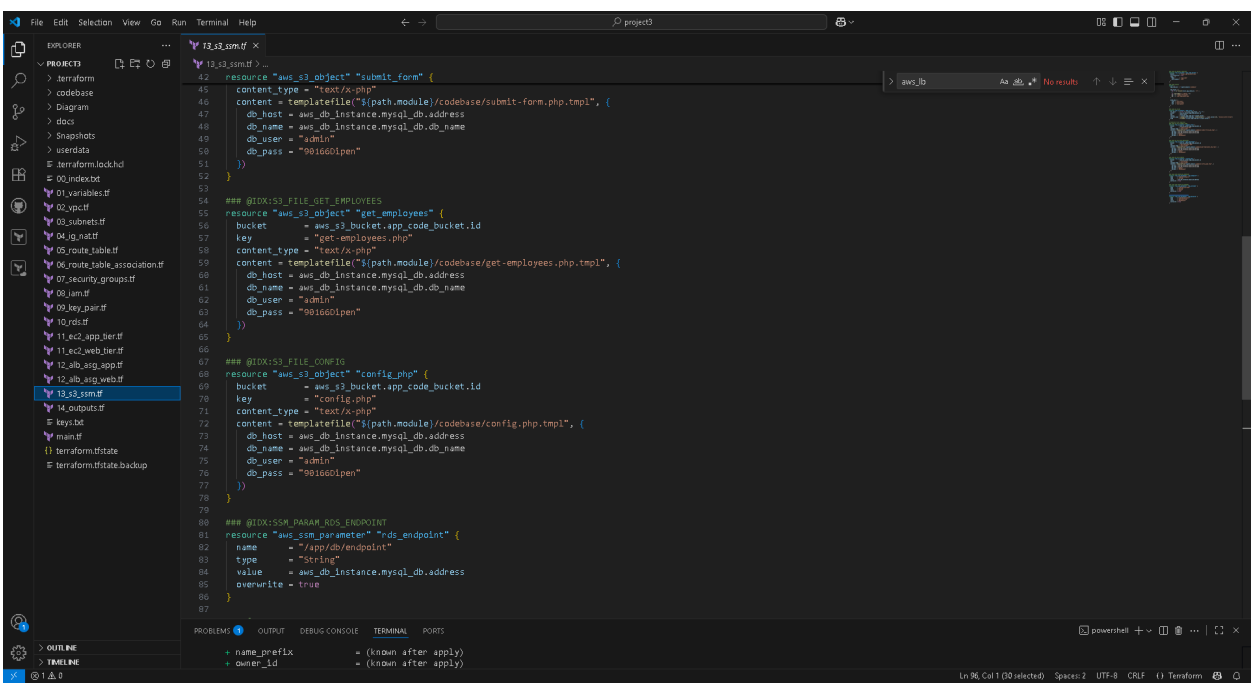
Phase 12: S3_SSM Implementation

- **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.



```
resource "aws_s3_bucket" "app_code_bucket" {
  bucket        = "dipen-app-backend-code"
  force_destroy = true
  tags = {
    Name        = "app-code"
    Environment = "Dev"
  }
}

### @IDX:LOCALS
locals {
  app_code_dir = "${path.module}/codebase"

  app_files = [
    for f in fileset(local.app_code_dir, "*") :
    f
    if !can(regex("\\.tmpl$", f))
    && f != "submit-form.php"
    && f != "get-employees.php"
  ]

  mime_map = {
    html = "text/html"
    php  = "text/x-php"
    txt  = "text/plain"
  }
}

### @IDX:S3_FILES
resource "aws_s3_object" "app_files" {
  for_each     = { for f in local.app_files : f => f }
  bucket       = aws_s3_bucket.app_code_bucket.id
  key          = each.value
  source       = "${local.app_code_dir}/${each.value}"
  content_type = lookup(local.mime_map, reverse(split(".", each.value))[0], "binary/octet-stream")
  etag         = filemd5("${local.app_code_dir}/${each.value}")
}

# Rendered from templates
### @IDX:S3_FILE_SUBMIT_FORM
resource "aws_s3_object" "submit_form" {
  bucket       = aws_s3_bucket.app_code_bucket.id
  key          = "submit-form.php"
  content_type = "text/x-php"
```

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



```
resource "aws_s3_object" "submit_form" {
  content_type = "text/x-php"
  content = templatefile("${path.module}/codebase/submit-form.php.tmpl", {
    db_host = aws_db_instance.mysql_db.address
    db_name = aws_db_instance.mysql_db.db_name
    db_user = "admin"
    db_pass = "90166Dipen"
  })
}

### @IDX:S3_FILE_GET_EMPLOYEES
resource "aws_s3_object" "get_employees" {
  bucket       = aws_s3_bucket.app_code_bucket.id
  key          = "get-employees.php"
  content_type = "text/x-php"
  content = templatefile("${path.module}/codebase/get-employees.php.tmpl", {
    db_host = aws_db_instance.mysql_db.address
    db_name = aws_db_instance.mysql_db.db_name
    db_user = "admin"
    db_pass = "90166Dipen"
  })
}

### @IDX:S3_FILE_CONFIG
resource "aws_s3_object" "config_php" {
  bucket       = aws_s3_bucket.app_code_bucket.id
  key          = "config.php"
  content_type = "text/x-php"
  content = templatefile("${path.module}/codebase/config.php.tmpl", {
    db_host = aws_db_instance.mysql_db.address
    db_name = aws_db_instance.mysql_db.db_name
    db_user = "admin"
    db_pass = "90166Dipen"
  })
}

### @IDX:SSM_PARAM_RDS_ENDPOINT
resource "aws_ssm_parameter" "rds_endpoint" {
  name      = "/app/db/endpoint"
  type      = "String"
  value     = aws_db_instance.mysql_db.address
  overwrite = true
}
```

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

File Edit Selection View Go Run Terminal Help

project3

EXPLORER

PROJECT3

- .terraform
- codebase
- Diagram
- docs
- Snapshots
- userdata
- .terraform.lock.hcl
- 00_index.txt
- 01_variables.tf
- 02_vpc.tf
- 03_subnets.tf
- 04_igw_nat.tf
- 05_route_table.tf
- 06_route_table_association.tf
- 07_security_groups.tf
- 08_iam.tf
- 09_key_pair.tf
- 10_rds.tf
- 11_ec2_app_tier.tf
- 11_ec2_web_tier.tf
- 12_alb_asg_app.tf
- 12_alb_asg_web.tf
- 13_s3_ssm.tf
- 14_outputs.tf
- keys.txt
- main.tf
- terraform.tfstate
- terraform.tfstate.backup

OUTLINE

TIMELINE

13_s3_ssm.tf > ...

aws_lb — No results

```tf
resource "aws_s3_object" "get_employees" {
  content = templatefile("${path.module}/codebase/get-employees.php.tmpl", {
    db_user = "admin"
    db_pass = "90166Dipen"
  })
}

### @IDX:S3_FILE_CONFIG
resource "aws_s3_object" "config_php" {
  bucket       = aws_s3_bucket.app_code_bucket.id
  key          = "config.php"
  content_type = "text/x-php"
  content = templatefile("${path.module}/codebase/config.php.tmpl", {
    db_host = aws_db_instance.mysql_db.address
    db_name = aws_db_instance.mysql_db.db_name
    db_user = "admin"
    db_pass = "90166Dipen"
  })
}

### @IDX:SSM_PARAM_RDS_ENDPOINT
resource "aws_ssm_parameter" "rds_endpoint" {
  name      = "/app/db/endpoint"
  type      = "String"
  value     = aws_db_instance.mysql_db.address
  overwrite = true
}

### @IDX:SSM_PARAM_DB_USERNAME
resource "aws_ssm_parameter" "db_username" {
  name      = "/app/db/username"
  type      = "String"
  value     = "admin"
  overwrite = true
}

### @IDX:SSM_PARAM_DB_PASSWORD
resource "aws_ssm_parameter" "db_password" {
  name      = "/app/db/password"
  type      = "SecureString"
  value     = "90166Dipen"
  overwrite = true
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell

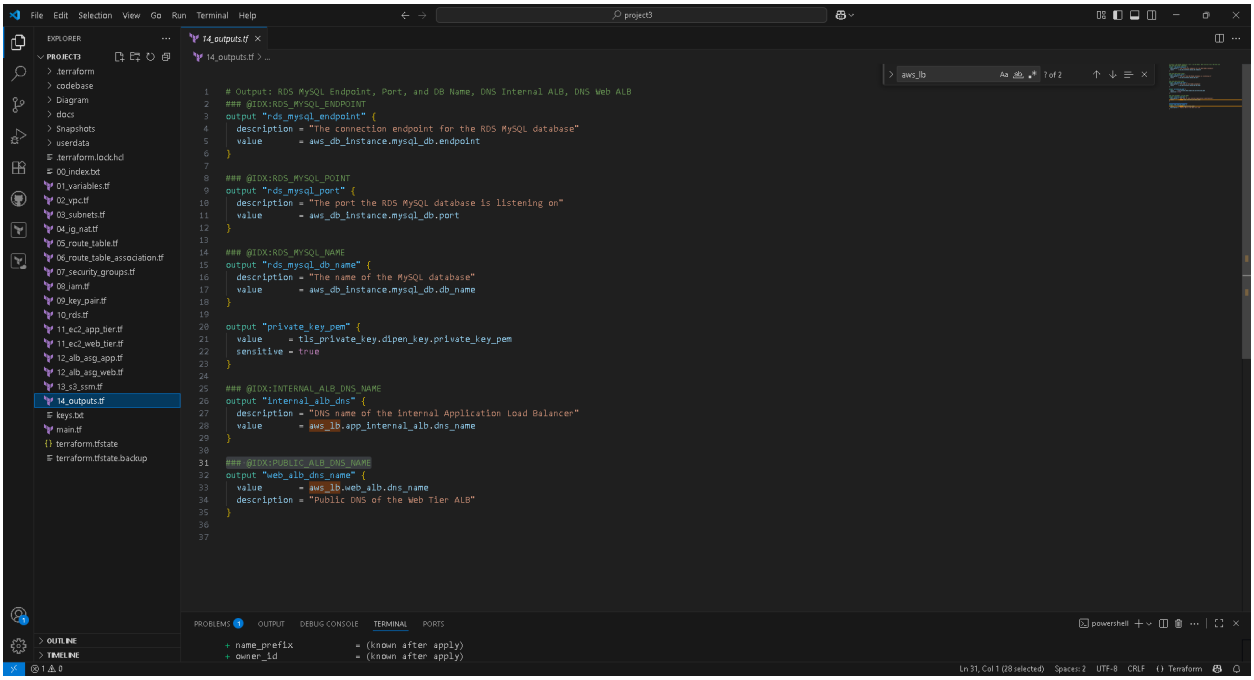
```
+ name_prefix             = (known after apply)
+ Owner_Id                = (known after apply)
```

Ln 96, Col 1 (30 selected) Spaces: 2 UTF-8 CRLF {} Terraform

Phase 14: Outputs

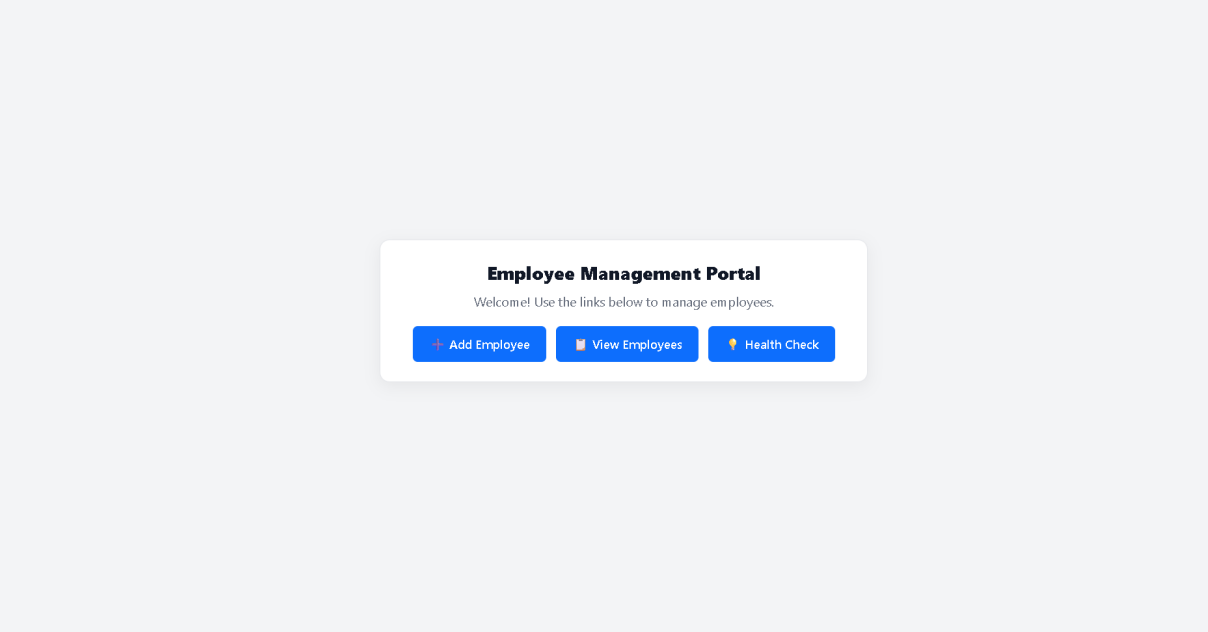
Defined Terraform outputs:

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

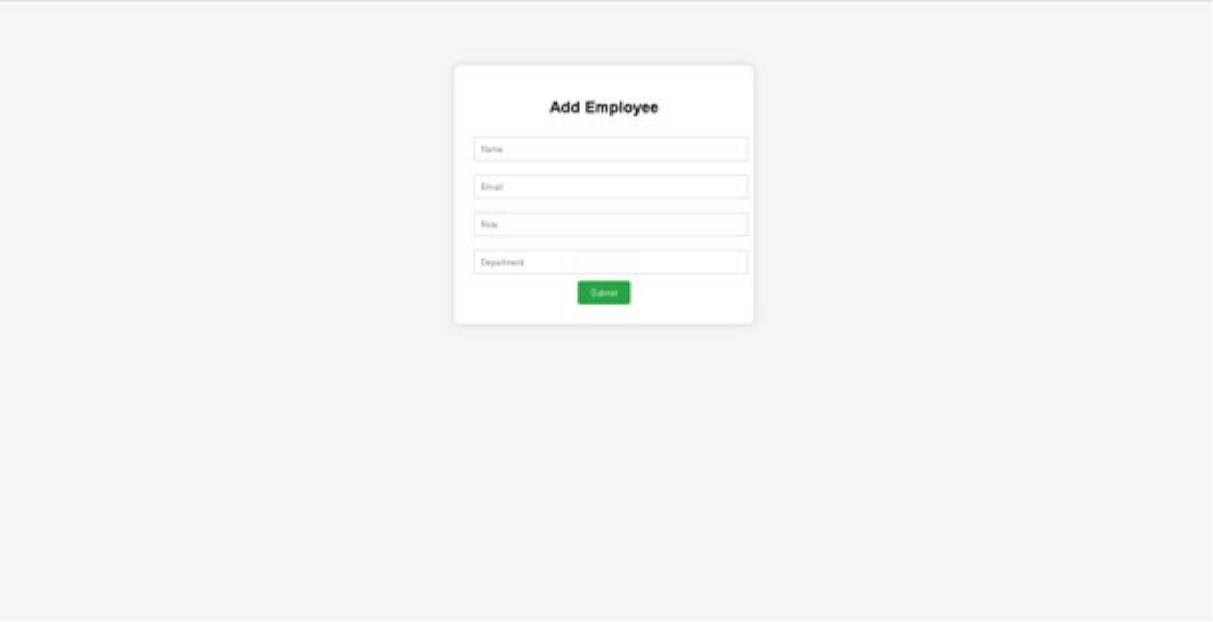


User Interface

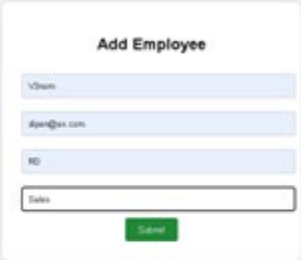
HomePage



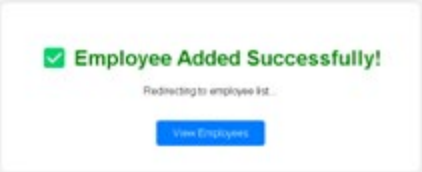
Add Employee



Data Input (form)



Form Submission



View Employees (Data Fetch from RDS)

Employee Records

Id	Name	Email	Role	Department	Created At
8	V3nom	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	V3nom	v3nom@v3nom.com	RD	IT	2025-07-23 05:28:53
5	V3nom	v3nom@v3nom.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.j@example.com	Project Manager	Operations	2025-07-23 04:34:32