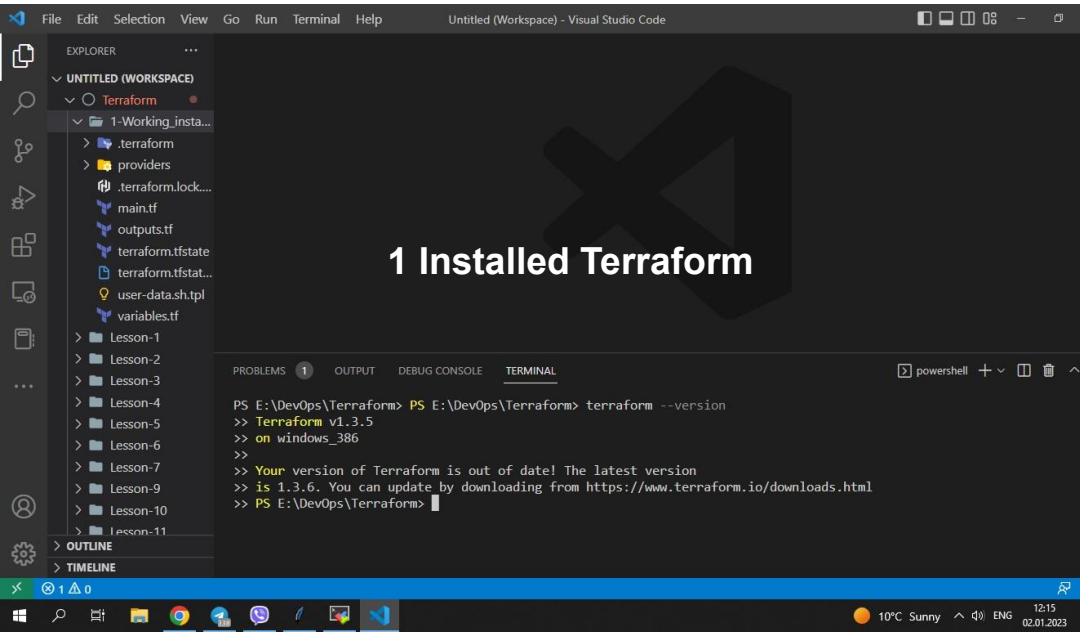

File Edit Selection View Go Run Terminal Help
Untitled (Workspace) - Visual Studio Code
EXPLORER
UNTITLED (WORKSPACE)
Terraform
1-Working_insta...
.terraform
providers
.terraform.lock....
main.tf
outputs.tf
terraform.tfstate
terraform.tfstat...
user-data.sh.tpl
variables.tf
Lesson-1
Lesson-2
Lesson-3
Lesson-4
Lesson-5
Lesson-6
Lesson-7
Lesson-9
Lesson-10
Lesson-11
OUTLINE
TIMELINE
1 Installed Terraform
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
PS E:\DevOps\Terraform> PS E:\DevOps\Terraform> terraform --version
>> Terraform v1.3.5
>> on windows_386
>>
>> Your version of Terraform is out of date! The latest version
>> is 1.3.6. You can update by downloading from https://www.terraform.io/downloads.html
>> PS E:\DevOps\Terraform>
1 0
10°C Sunny
ENG
12:15
02.01.2023

```
provider "aws" {

}


resource "aws_instance" "my_Ubuntu" {
  ami           = "ami-0caef02b518350c8b"
  instance_type = "t2.micro"

  tags = {
    Name    = "My Ubuntu Server"
    Owner   = "Anton Nevero"
    Project = "Terraform Lessons"
  }
}

resource "aws_instance" "my_Ubuntu2" {
  ami           = "ami-0caef02b518350c8b"
  instance_type = "t2.micro"

  tags = {
    Name    = "My Amazon Server"
    Owner   = "Anton Nevero"
    Project = "Terraform Lessons"
  }
}

```

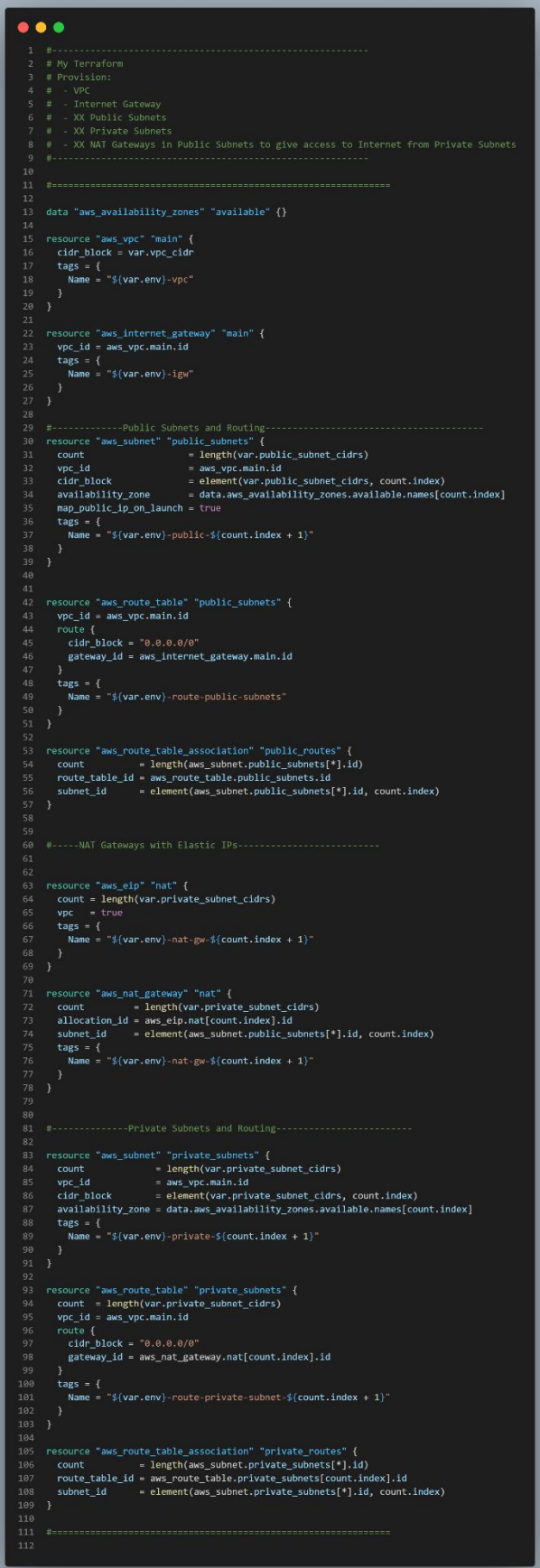
2. Running 2 instances on AWS

```
#-------------------------------------------------------------
# My Terraform
#
#
# Made by Anton Nevero
#-------------------------------------------------------------

provider "aws" {
  region = var.region
}

resource "aws_default_vpc" "default" {}

data "aws_ami" "latest_ubuntu" {
  owners      = ["099720109477"]
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-*"]
  }
}

/* data "aws_ami" "latest_amazon" {
  owners      = ["137112412989"]
  most_recent = true
  filter {
    name   = "name"
    values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
  }
}*/

resource "aws_instance" "my_server" {
  ami                    = data.aws_ami.latest_ubuntu.id
  instance_type          = var.instance_type
  count = 4
  vpc_security_group_ids = [aws_security_group.my_server.id]
  key_name               = "Ubuntu"
  //monitoring = var.enable_detailed_monitoring
  //tags = var.common_tags
  tags = merge(var.common_tags, { Name = "Working Instance" })

}

resource "aws_security_group" "my_server" {
  name   = "My Security Group"
  vpc_id = aws_default_vpc.default.id

  dynamic "ingress" {
    for_each = var.allow_ports
    content {
      from_port   = ingress.value
      to_port     = ingress.value
      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 = merge(var.common_tags, { Name = "My SG" })
}

```

3. Start 4 instances on AWS

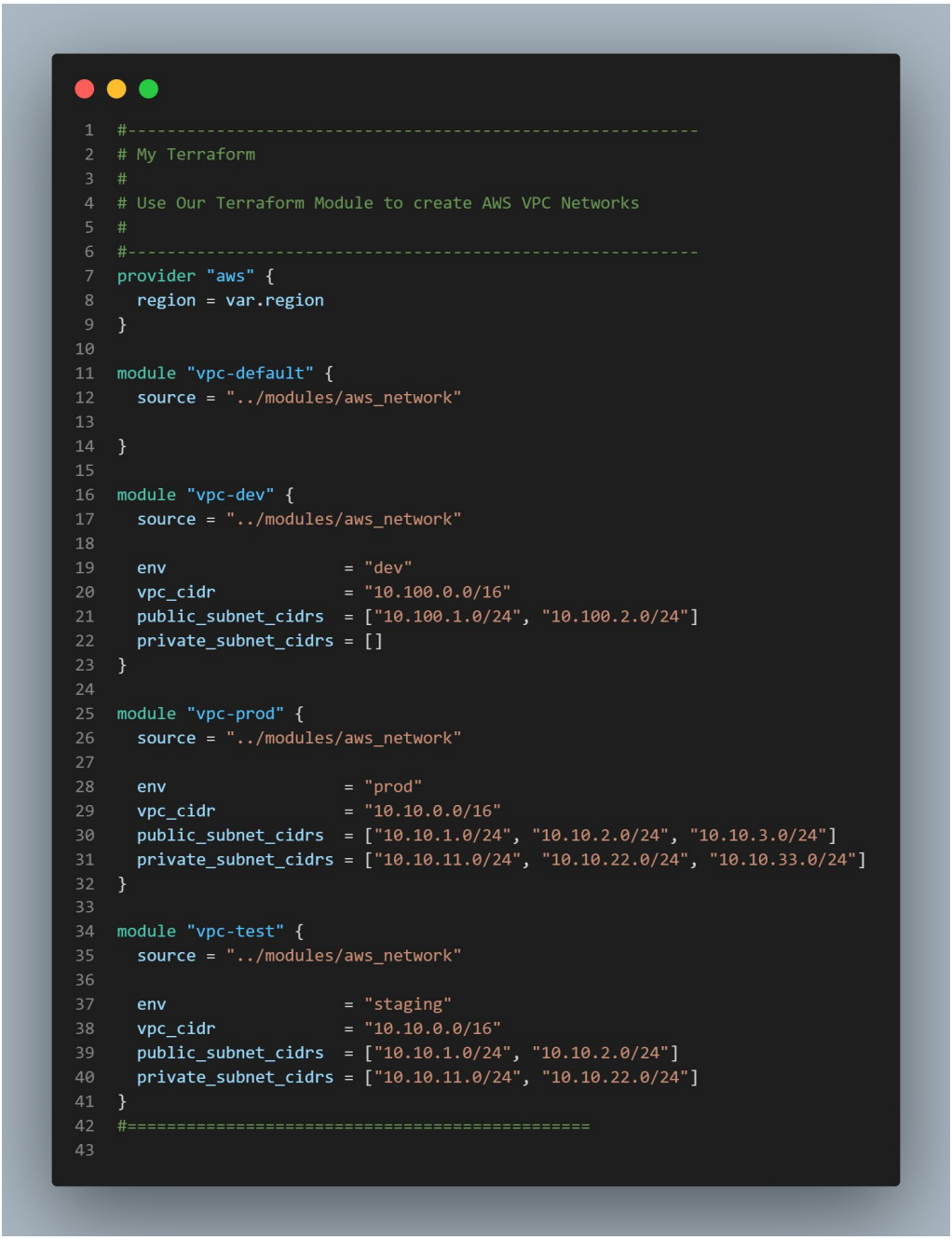
```hcl
#-------------------------------------------------------------
# My Terraform
#
# Genarate Password
# Store Password in SSM Parameter Store
# Get Password from SSM Parameter Store
# Example of Use Password in RDS
#
# Made by Anton Nevero
#-------------------------------------------------------------
provider "aws" {
  region = "ca-central-1"
}

// Generate Password
resource "random_string" "rds_password" {
  length           = 12
  special          = true
  override_special = "!#$&"

  keepers = {
    kepeer1 = var.name
    //keperr2 = var.something
  }
}

// Store Password in SSM Parameter Store
resource "aws_ssm_parameter" "rds_password" {
  name        = "/prod/mysql"
  description = "Master Password for RDS MySQL"
  type        = "SecureString"
  value       = random_string.rds_password.result
}

// Get Password from SSM Parameter Store
data "aws_ssm_parameter" "my_rds_password" {
  name       = "/prod/mysql"
  depends_on = [aws_ssm_parameter.rds_password]
}


// Example of Use Password in RDS
resource "aws_db_instance" "default" {
  identifier           = "prod-rds"
  allocated_storage    = 20
  storage_type         = "gp2"
  engine               = "mysql"
  engine_version       = "5.7"
  instance_class       = "db.t2.micro"
  name                 = "prod"
  username             = "administrator"
  password             = data.aws_ssm_parameter.my_rds_password.value
  parameter_group_name = "default.mysql5.7"
  skip_final_snapshot  = true
  apply_immediately    = true
}
```

4. Generate password for RDS



```
#------------------------------------------------------------
# My Terraform
# Provision:
#  - VPC
#  - Internet Gateway
#  - XX Public Subnets
#  - XX Private Subnets
#  - XX NAT Gateways in Public Subnets to give access to Internet from Private Subnets
#------------------------------------------------------------

#============================================================

data "aws_availability_zones" "available" {}

resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags = {
    Name = "${var.env}-vpc"
  }
}

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "${var.env}-igw"
  }
}

#-------------Public Subnets and Routing--------------------------------------
resource "aws_subnet" "public_subnets" {
  count                   = length(var.public_subnet_cidrs)
  vpc_id                  = aws_vpc.main.id
  cidr_block              = element(var.public_subnet_cidrs, count.index)
  availability_zone       = data.aws_availability_zones.available.names[count.index]
  map_public_ip_on_launch = true
  tags = {
    Name = "${var.env}-public-${count.index + 1}"
  }
}


resource "aws_route_table" "public_subnets" {
  vpc_id = aws_vpc.main.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.main.id
  }
  tags = {
    Name = "${var.env}-route-public-subnets"
  }
}

resource "aws_route_table_association" "public_routes" {
  count          = length(aws_subnet.public_subnets[*].id)
  route_table_id = aws_route_table.public_subnets.id
  subnet_id      = element(aws_subnet.public_subnets[*].id, count.index)
}


#-----NAT Gateways with Elastic IPs-------------------------


resource "aws_eip" "nat" {
  count = length(var.private_subnet_cidrs)
  vpc   = true
  tags = {
    Name = "${var.env}-nat-gw-${count.index + 1}"
  }
}

resource "aws_nat_gateway" "nat" {
  count         = length(var.private_subnet_cidrs)
  allocation_id = aws_eip.nat[count.index].id
  subnet_id     = element(aws_subnet.public_subnets[*].id, count.index)
  tags = {
    Name = "${var.env}-nat-gw-${count.index + 1}"
  }
}


#-------------Private Subnets and Routing-------------------------

resource "aws_subnet" "private_subnets" {
  count             = length(var.private_subnet_cidrs)
  vpc_id            = aws_vpc.main.id
  cidr_block        = element(var.private_subnet_cidrs, count.index)
  availability_zone = data.aws_availability_zones.available.names[count.index]
  tags = {
    Name = "${var.env}-private-${count.index + 1}"
  }
}

resource "aws_route_table" "private_subnets" {
  count  = length(var.private_subnet_cidrs)
  vpc_id = aws_vpc.main.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.nat[count.index].id
  }
  tags = {
    Name = "${var.env}-route-private-subnet-${count.index + 1}"
  }
}

resource "aws_route_table_association" "private_routes" {
  count          = length(aws_subnet.private_subnets[*].id)
  route_table_id = aws_route_table.private_subnets[count.index].id
  subnet_id      = element(aws_subnet.private_subnets[*].id, count.index)
}

#============================================================

```

5. Module for AWS



```
#-------------------------------------------------------
# My Terraform
#
# Use Our Terraform Module to create AWS VPC Networks
#
#-------------------------------------------------------
provider "aws" {
  region = var.region
}

module "vpc-default" {
  source = "../modules/aws_network"

}

module "vpc-dev" {
  source = "../modules/aws_network"

  env                 = "dev"
  vpc_cidr            = "10.100.0.0/16"
  public_subnet_cidrs  = ["10.100.1.0/24", "10.100.2.0/24"]
  private_subnet_cidrs = []
}

module "vpc-prod" {
  source = "../modules/aws_network"

  env                 = "prod"
  vpc_cidr            = "10.10.0.0/16"
  public_subnet_cidrs  = ["10.10.1.0/24", "10.10.2.0/24", "10.10.3.0/24"]
  private_subnet_cidrs = ["10.10.11.0/24", "10.10.22.0/24", "10.10.33.0/24"]
}

module "vpc-test" {
  source = "../modules/aws_network"

  env                 = "staging"
  vpc_cidr            = "10.10.0.0/16"
  public_subnet_cidrs  = ["10.10.1.0/24", "10.10.2.0/24"]
  private_subnet_cidrs = ["10.10.11.0/24", "10.10.22.0/24"]
}
#=============================================

```

6. Using module for AWS

```
#------------------------------------------------------------
# My Terraform
#
# Remote State on S3
#
# Made by Anton Nevero
#------------------------------------------------------------
provider "aws" {
  region = "eu-central-1"
}

terraform {
  backend "s3" {
    bucket = "anton-nevero-project-kgb-terraform-state" // Bucket where to SAVE Terraform State
    key    = "dev/network/terraform.tfstate"            // Object name in the bucket to SAVE Terraform State
    region = "eu-central-1"                               // Region where bucket created
  }
}

#============================================================

data "aws_availability_zones" "available" {}

resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags = {
    Name = "${var.env}-vpc"
  }
}

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "${var.env}-igw"
  }
}


resource "aws_subnet" "public_subnets" {
  count                   = length(var.public_subnet_cidrs)
  vpc_id                  = aws_vpc.main.id
  cidr_block              = element(var.public_subnet_cidrs, count.index)
  availability_zone       = data.aws_availability_zones.available.names[count.index]
  map_public_ip_on_launch = true
  tags = {
    Name = "${var.env}-puvlic-${count.index + 1}"
  }
}


resource "aws_route_table" "public_subnets" {
  vpc_id = aws_vpc.main.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.main.id
  }
  tags = {
    Name = "${var.env}-route-public-subnets"
  }
}

resource "aws_route_table_association" "public_routes" {
  count          = length(aws_subnet.public_subnets[*].id)
  route_table_id = aws_route_table.public_subnets.id
  subnet_id      = element(aws_subnet.public_subnets[*].id, count.index)
}

#============================================================
```

7. Save tfstate on S3 Bucket

```
#-------------------------------------------------------------
# My Terraform
#
# Remote State on S3
#
# Made by Anton Nevero
#-------------------------------------------------------------
provider "aws" {
  region = "eu-central-1"
}

terraform {
  backend "s3" {
    bucket = "anton-nevero-project-kgb-terraform-state" // Bucket where to SAVE Terraform State
    key    = "dev/servers/terraform.tfstate"             // Object name in the bucket to SAVE Terraform State
    region = "eu-central-1"                              // Region where bucket created
  }
}
#================================================================


data "terraform_remote_state" "network" {
  backend = "s3"
  config = {
    bucket = "anton-nevero-project-kgb-terraform-state" // Bucket from where to GET Terraform State
    key    = "dev/network/terraform.tfstate"             // Object name in the bucket to GET Terraform state
    region = "eu-central-1"                              // Region where bucket created
  }
}

data "aws_ami" "latest_amazon_linux" {
  owners      = ["amazon"]
  most_recent = true
  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-gp2"]
  }
}
#================================================================


resource "aws_instance" "web_server" {
  ami                    = data.aws_ami.latest_amazon_linux.id
  instance_type          = "t3.micro"
  vpc_security_group_ids = [aws_security_group.webserver.id]
  subnet_id              = data.terraform_remote_state.network.outputs.public_subnet_ids[0]
  user_data              = <<EOF
#!/bin/bash
yum -y update
yum -y install httpd
myip=`curl http://169.254.169.254/latest/meta-data/local-ipv4`
echo "<h2>WebServer with IP: $myip</h2><br>Build by Terraform with Remote State"  >  /var/www/html/index.html
sudo service httpd start
chkconfig httpd on
EOF
  tags = {
    Name = "${var.env}-WebServer"
  }
}

resource "aws_security_group" "webserver" {
  name = "WebServer Security Group"
  vpc_id = data.terraform_remote_state.network.outputs.vpc_id

  ingress {
    from_port = 80
    to_port = 80
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = [data.terraform_remote_state.network.outputs.vpc_cidr]
  }
  egress {
    from_port = 0
    to_port = 0
    protocol = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "${var.env}-web-server-sg"
    Owner = "Anton Nevero"
  }
}

#================================================================
```

8. Take tfstate from S3 Bucket