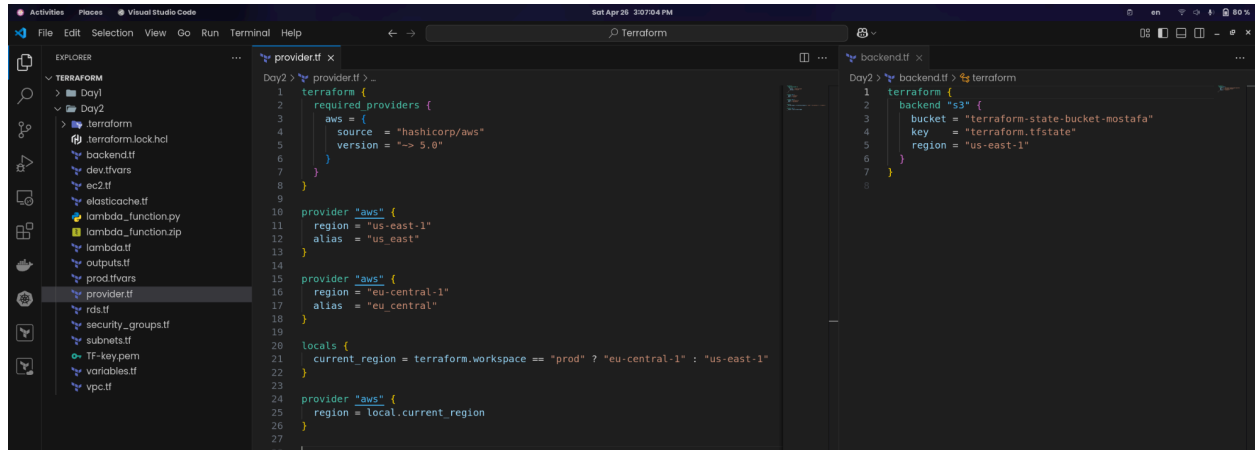


Terraform Lab 2

Provider & backend

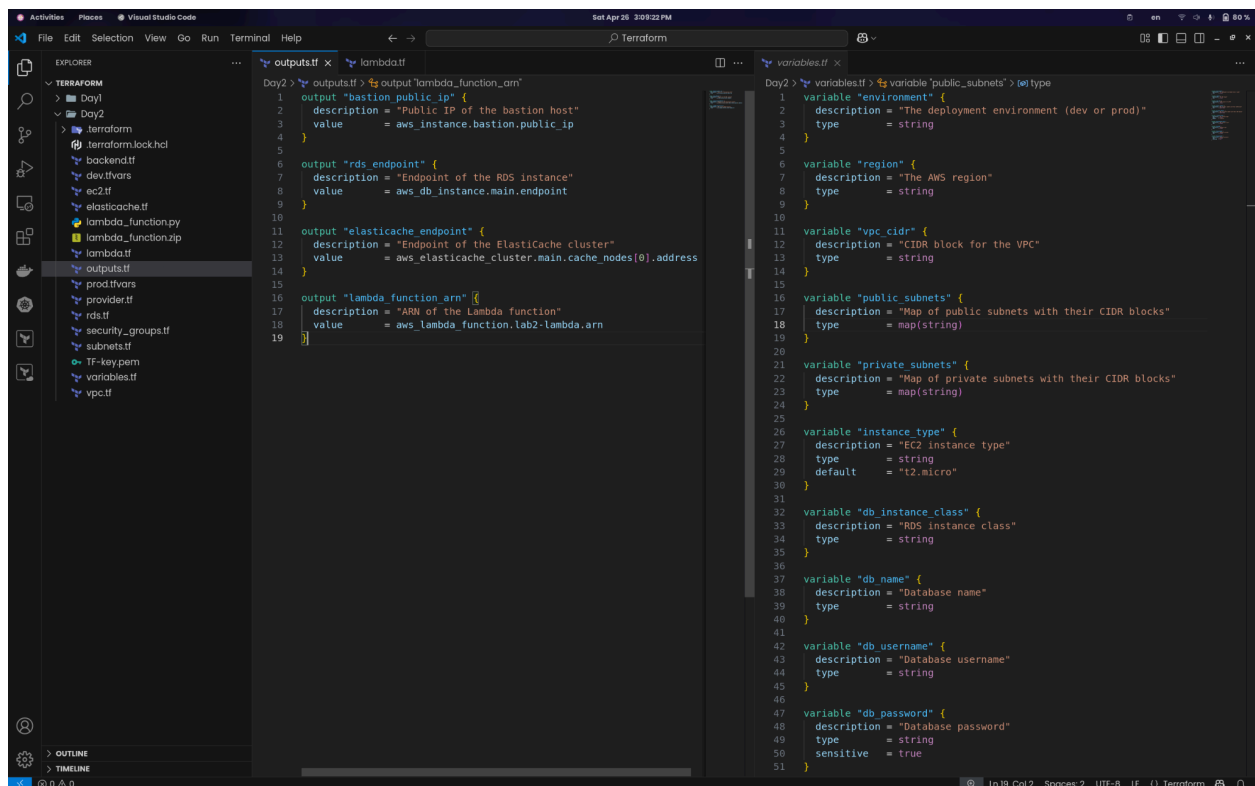


The screenshot shows the Visual Studio Code interface with two Terraform configuration files open. The left pane shows the Explorer view with a project structure including Day1, Day2, .terraform, and various .tf files. The main editor displays the content of 'provider.tf' and 'backend.tf'.

```
Day2 > provider.tf
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "~> 5.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   region = "us-east-1"
12   alias = "us-east"
13 }
14
15 provider "aws" {
16   region = "eu-central-1"
17   alias = "eu-central"
18 }
19
20 locals {
21   current_region = terraform.workspace == "prod" ? "eu-central-1" : "us-east-1"
22 }
23
24 provider "aws" {
25   region = local.current_region
26 }
27
```

```
Day2 > backend.tf
1 terraform {
2   backend "s3" {
3     bucket = "terraform-state-bucket-mostafa"
4     key = "terraform.tfstate"
5     region = "us-east-1"
6   }
7 }
```

Variables & outputs



The screenshot shows the Visual Studio Code interface with two Terraform configuration files open. The left pane shows the Explorer view with a project structure including Day1, Day2, .terraform, and various .tf files. The main editor displays the content of 'outputs.tf' and 'variables.tf'.

```
Day2 > outputs.tf
1 output "bastion_public_ip" {
2   description = "Public IP of the bastion host"
3   value       = aws_instance.bastion.public_ip
4 }
5
6 output "rds_endpoint" {
7   description = "Endpoint of the RDS instance"
8   value       = aws_db_instance.main.endpoint
9 }
10
11 output "elasticache_endpoint" {
12   description = "Endpoint of the Elasticache cluster"
13   value       = aws_elasticache_cluster.main.cache_nodes[0].address
14 }
15
16 output "lambda_function_arn" {
17   description = "ARN of the Lambda function"
18   value       = aws_lambda_function.lab2-lambda.arn
19 }
```

```
Day2 > variables.tf
1 variable "environment" {
2   description = "The deployment environment (dev or prod)"
3   type        = string
4 }
5
6 variable "region" {
7   description = "The AWS region"
8   type        = string
9 }
10
11 variable "vpc_cidr" {
12   description = "CIDR block for the VPC"
13   type        = string
14 }
15
16 variable "public_subnets" {
17   description = "Map of public subnets with their CIDR blocks"
18   type        = map(string)
19 }
20
21 variable "private_subnets" {
22   description = "Map of private subnets with their CIDR blocks"
23   type        = map(string)
24 }
25
26 variable "instance_type" {
27   description = "EC2 instance type"
28   type        = string
29   default     = "t2.micro"
30 }
31
32 variable "db_instance_class" {
33   description = "RDS instance class"
34   type        = string
35 }
36
37 variable "db_name" {
38   description = "Database name"
39   type        = string
40 }
41
42 variable "db_username" {
43   description = "Database username"
44   type        = string
45 }
46
47 variable "db_password" {
48   description = "Database password"
49   type        = string
50   sensitive   = true
51 }
```

Dev & Prod tfvars

```
prod.tfvars x
Day2 > prod.tfvars > environment
1 environment = "prod"
2 region      = "eu-central-1"
3 vpc_cidr    = "10.1.0.0/16"
4 public_subnets = {
5   "prod-public-1" = "10.1.1.0/24",
6   "prod-public-2" = "10.1.2.0/24"
7 }
8 private_subnets = {
9   "prod-private-1" = "10.1.3.0/24",
10  "prod-private-2" = "10.1.4.0/24"
11 }
12 instance_type = "t2.micro"
13 db_instance_class = "db.t2.micro"
14 db_name         = "proddb"
15 db_username     = "produser"
16 db_password     = "ProdPassword123!"

dev.tfvars x
Day2 > dev.tfvars > environment
1 environment = "dev"
2 region      = "us-east-1"
3 vpc_cidr    = "10.0.0.0/16"
4 public_subnets = {
5   "dev-public-1" = "10.0.1.0/24",
6   "dev-public-2" = "10.0.2.0/24"
7 }
8 private_subnets = {
9   "dev-private-1" = "10.0.3.0/24",
10  "dev-private-2" = "10.0.4.0/24"
11 }
12 instance_type = "t2.micro"
13 db_instance_class = "db.t2.micro"
14 db_name         = "devddb"
15 db_username     = "devuser"
16 db_password     = "DevPassword123!"
```

VPC & subnets

```
vpc.tf x
Day2 > vpc.tf > resource "aws_internet_gateway" "main"
1 resource "aws_vpc" "main" {
2   cidr_block = var.vpc_cidr
3   enable_dns_support = true
4   enable_dns_hostnames = true
5
6   tags = {
7     Name = "${var.environment}-vpc"
8     Environment = var.environment
9   }
10 }
11
12 resource "aws_internet_gateway" "main" {
13   vpc_id = aws_vpc.main.id
14
15   tags = {
16     Name = "${var.environment}-igw"
17     Environment = var.environment
18   }
19 }

subnets.tf x
Day2 > subnets.tf > resource "aws_subnet" "public"
1 resource "aws_subnet" "public" {
2   for_each = var.public_subnets
3   vpc_id = aws_vpc.main.id
4   cidr_block = each.value
5   availability_zone = element(data.aws_availability_zones.available.names, index(
6     map_public_ip_on_launch = true
7   ))
8
9   tags = {
10     Name = each.key
11     Environment = var.environment
12   }
13 }
14
15 resource "aws_subnet" "private" {
16   for_each = var.private_subnets
17   vpc_id = aws_vpc.main.id
18   cidr_block = each.value
19   availability_zone = element(data.aws_availability_zones.available.names, index(keys(
20     map_public_ip_on_launch = true
21   )))
22
23   tags = {
24     Name = each.key
25     Environment = var.environment
26   }
27 }
28
29 data "aws_availability_zones" "available" {
30   state = "available"
31 }
```

RT & RT association

```

resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.main.id
  }

  tags = {
    Name      = "${var.environment}-public-rt"
    Environment = var.environment
  }
}

resource "aws_route_table_association" "public" {
  for_each = aws_subnet.public
  subnet_id = each.value.id
  route_table_id = aws_route_table.public.id
}

resource "aws_route_table" "private" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name      = "${var.environment}-private-rt"
    Environment = var.environment
  }
}

resource "aws_route_table_association" "private" {
  for_each = aws_subnet.private
  subnet_id = each.value.id
  route_table_id = aws_route_table.private.id
}

```

EC2 & its SG

```

Day2 > security_groups.tf x
1 resource "aws_security_group" "bastion" {
2   name      = "${var.environment}-bastion-sg"
3   description = "Allow SSH access to bastion host"
4   vpc_id    = aws_vpc.main.id
5
6   ingress {
7     from_port = 22
8     to_port   = 22
9     protocol  = "tcp"
10    cidr_blocks = ["0.0.0.0/0"]
11  }
12
13  egress {
14    from_port = 0
15    to_port   = 0
16    protocol  = "-1"
17    cidr_blocks = ["0.0.0.0/0"]
18  }
19
20  tags = {
21    Name      = "${var.environment}-bastion-sg"
22    Environment = var.environment
23  }
24 }
25
26 resource "aws_security_group" "rds" {

Day2 > ec2.tf x
1 data "aws_ami" "amazon_linux" {
2   most_recent = true
3   owners      = ["amazon"]
4
5   filter {
6     name   = "name"
7     values = ["amzn2-ami-hvm-*x86_64-gp2"]
8   }
9 }
10
11 resource "aws_instance" "bastion" {
12   ami              = data.aws_ami.amazon_linux.id
13   instance_type    = var.instance_type
14   subnet_id        = values(aws_subnet.public)[0].id
15   vpc_security_group_ids = [aws_security_group.bastion.id]
16   key_name         = "TF-key"
17
18   tags = {
19     Name      = "${var.environment}-bastion"
20     Environment = var.environment
21   }
22
23   provisioner "local-exec" {
24     command = "echo 'Bastion Public IP: ${self.public_ip}'"
25   }
26 }

```

RED & its SG

The screenshot shows the Visual Studio Code interface with two Terraform files open. The left pane shows the Explorer view with a project structure including files like `Day1`, `Day2`, `terraform.lock.hcl`, `backend.tf`, `dev.tfvars`, `ec2.tf`, `elasticache.tf`, `lambda_function.py`, `lambda_function.zip`, `outputs.tf`, `prod.tfvars`, `provider.tf`, `rds.tf`, `security_groups.tf`, `subnets.tf`, `TF-key.pem`, `variables.tf`, and `vpc.tf`. The main editor shows the `rds.tf` file with Terraform code for an `aws_db_instance` resource. The code includes settings for `engine` (MySQL), `instance_class` (`db.t3.micro`), `storage_type` (`gp2`), and `tags`. It also defines `username`, `password`, and `db_name`. The `security_groups.tf` file is also visible, showing an `aws_security_group` resource for the RDS instance, with `ingress` and `egress` rules defined.

Elasticcach & its SG

The screenshot shows the Visual Studio Code interface with two Terraform files open. The left pane shows the Explorer view with a project structure including files like `Day1`, `Day2`, `terraform.lock.hcl`, `backend.tf`, `dev.tfvars`, `ec2.tf`, `elasticache.tf`, `lambda_function.py`, `lambda_function.zip`, `outputs.tf`, `prod.tfvars`, `provider.tf`, `rds.tf`, `security_groups.tf`, `subnets.tf`, `TF-key.pem`, `variables.tf`, and `vpc.tf`. The main editor shows the `security_groups.tf` file with Terraform code for an `aws_security_group` resource for Elasticache. The code includes settings for `name`, `description`, `vpc_id`, and `tags`. It also defines `ingress` and `egress` rules. The `elasticache.tf` file is also visible, showing an `aws_elasticache_subnet_group` resource and an `aws_elasticache_cluster` resource.

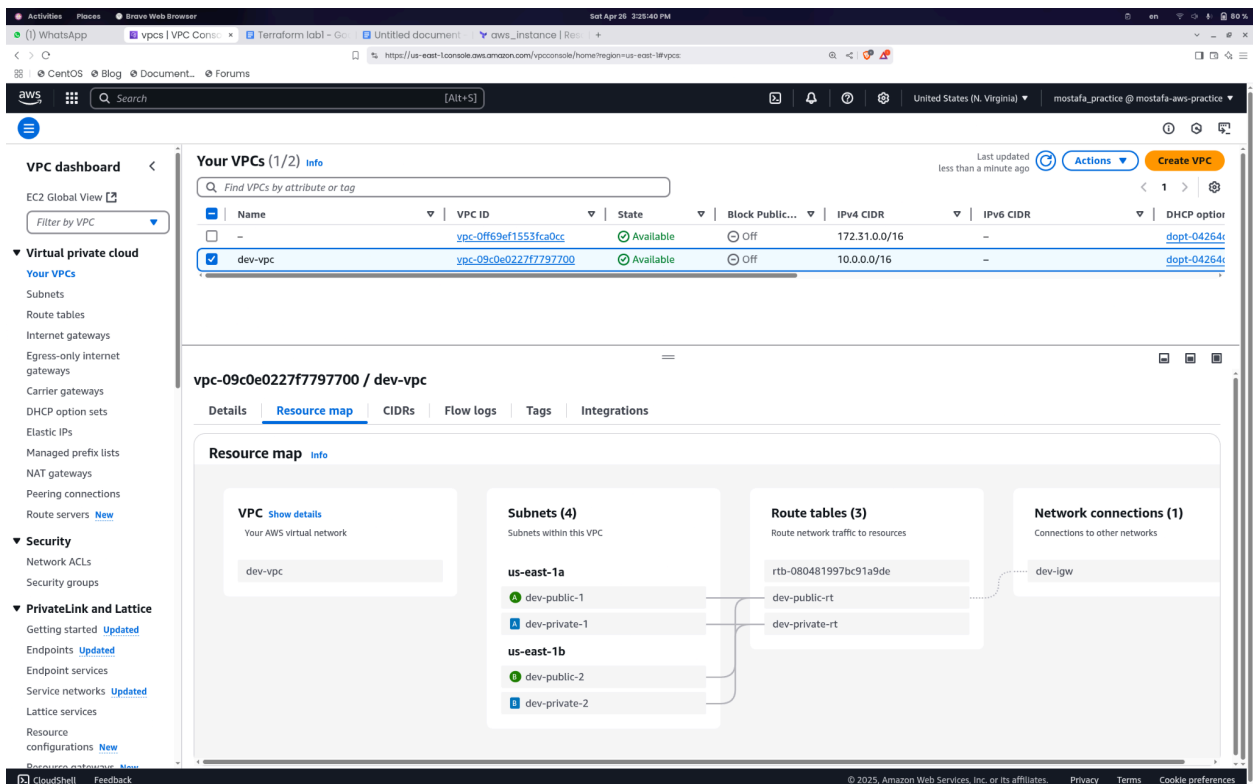
Lambda

```
Day2 > lambda.tf x
1 resource "aws_iam_role" "lambda_role" {
2   name = "simple-lambda-role"
3
4   assume_role_policy = <<EOF
5   {
6     "Version": "2012-10-17",
7     "Statement": [
8       {
9         "Action": "sts:AssumeRole",
10        "Principal": {
11          "Service": "lambda.amazonaws.com"
12        },
13        "Effect": "Allow",
14        "Sid": ""
15      }
16    ]
17  }
18  EOF
19 }
20
21 resource "aws_iam_role_policy_attachment" "lambda_policy" {
22   role       = aws_iam_role.lambda_role.name
23   policy_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
24 }
25
26 data "archive_file" "lambda_zip" {
27   type      = "zip"
28   source_file = "${path.module}/lambda_function.py"
29   output_path = "${path.module}/lambda_function.zip"
30 }
31
32 resource "aws_lambda_function" "lab2-lambda" {
33   filename      = data.archive_file.lambda_zip.output_path
34   function_name = "simple-lambda"
35   role          = aws_iam_role.lambda_role.arn
36   handler       = "lambda_function.lambda_handler"
37   runtime       = "python3.12"
38   source_code_hash = data.archive_file.lambda_zip.output_base64sha256
39 }
40
```

```
Day2 > lambda_function.py x
1 def lambda_handler(event, context):
2   return "Hello World"
3
```

AWS Console After Apply

Network



SGs

EC2 Global View

Events

▼ Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

▼ Images

AMIs

AMI Catalog

▼ Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

▼ Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

▼ Load Balancing

Load Balancers

Target Groups

Trust Stores

Security Groups (6) info

Find security groups by attribute or tag

☐

Name

☐

-

☐

-

☐

dev-rds-sg

☐

dev-elasticache-sg

☐

-

☐

dev-bastion-sg

☐

sg-01f68ed57b93eaba0

☐

sg-0b33a8e479358801d

☐

sg-0fb75e9b2fab9433f

☐

sg-09bd0f50656dcc7a9

☐

sg-03cf27fb68b98abd9

☐

sg-0194e9fd7ce21d295

☐

default

☐

launch-wizard-1

☐

dev-rds-sg

☐

dev-elasticache-sg

☐

default

☐

dev-bastion-sg

☐

sg-01f68ed57b93eaba0

☐

sg-0b33a8e479358801d

☐

sg-0fb75e9b2fab9433f

☐

sg-09bd0f50656dcc7a9

☐

sg-03cf27fb68b98abd9

☐

sg-0194e9fd7ce21d295

☐

default

☐

launch-wizard-1

☐

dev-rds-sg

☐

dev-elasticache-sg

☐

default

☐

dev-bastion-sg

☐

vpc-0ff69ef1553fca0cc

☐

vpc-0ff69ef1553fca0cc

☐

vpc-09c0e0227f7797700

☐

vpc-09c0e0227f7797700

☐

vpc-09c0e0227f7797700

☐

vpc-09c0e0227f7797700

☐

default VPC security group

☐

launch-wizard-1 created 2025-01-23T2...

☐

Allow MySQL access from private subnets

☐

Allow Redis access from private subnets

☐

default VPC security group

☐

Allow SSH access to bastion host

Actions

Export security groups to CSV

Create security group

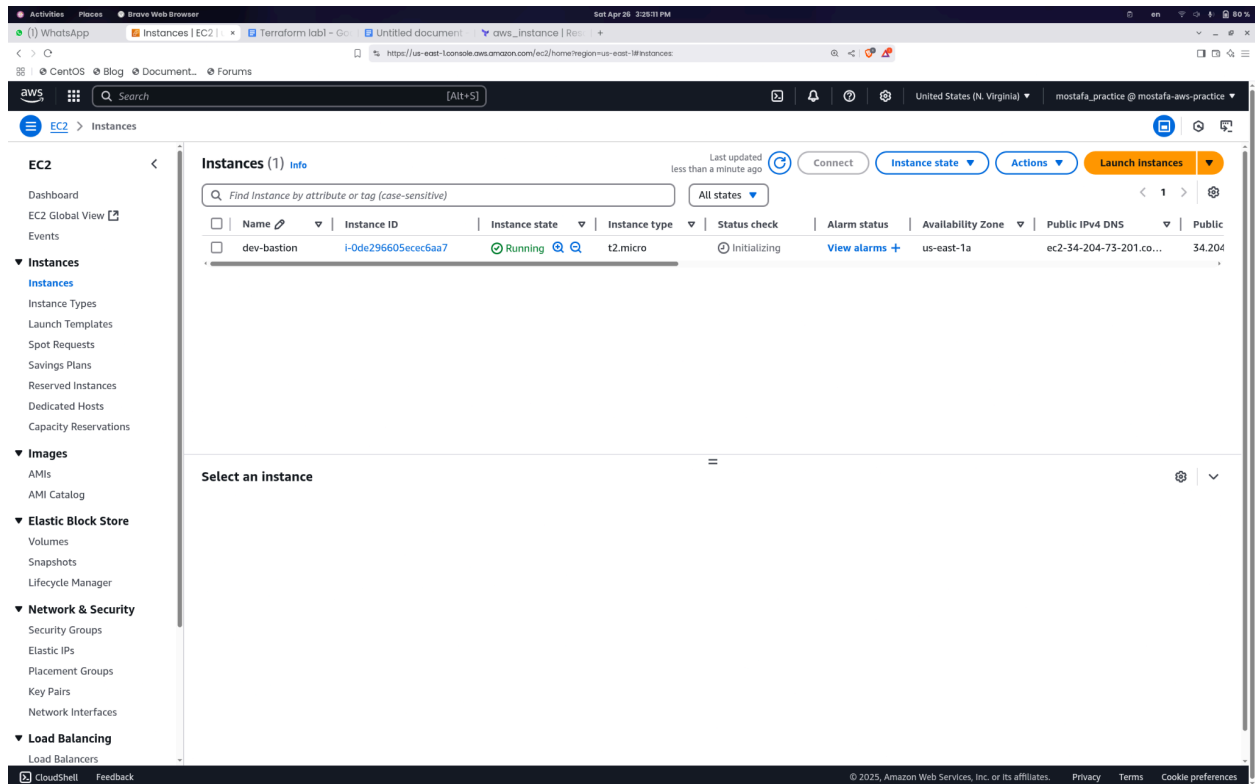
Select a security group

CloudShell

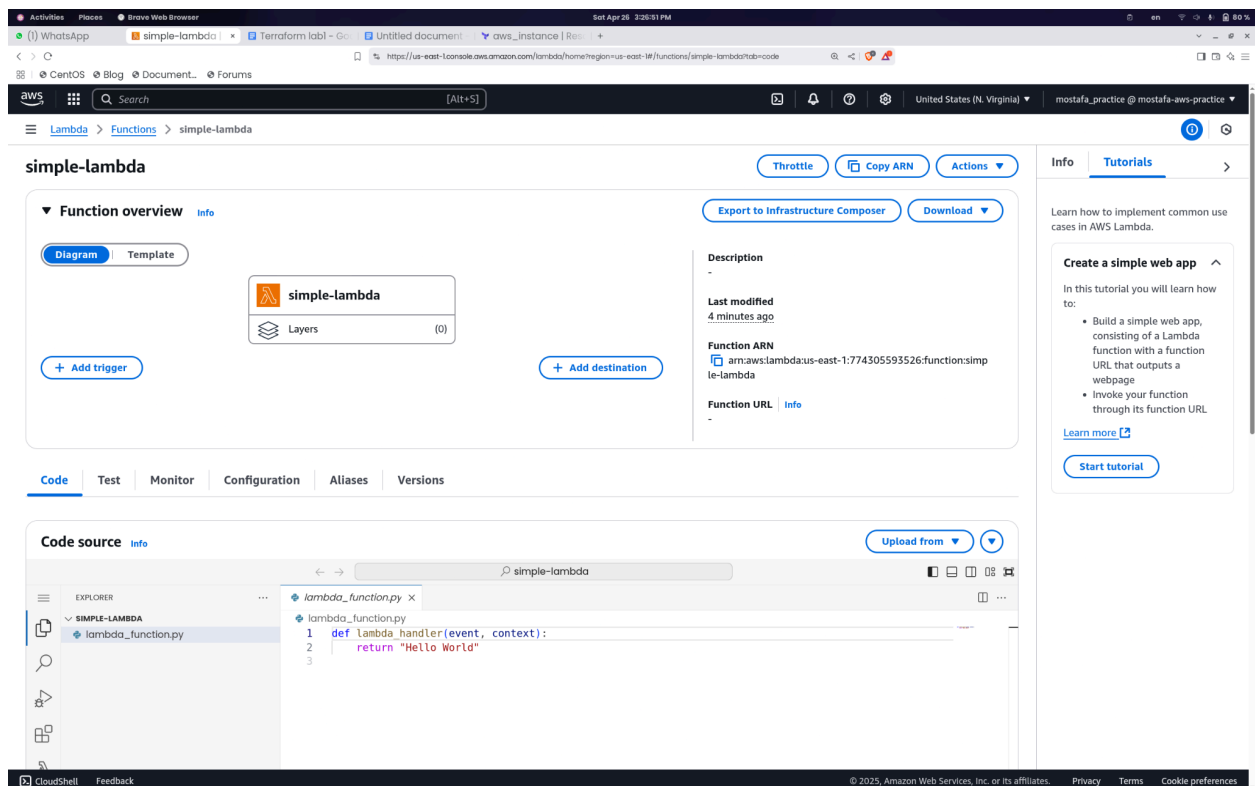
Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2



Lambda



Elastic cash

The screenshot displays the AWS ElasticCache console interface. At the top, a navigation bar includes the AWS logo, a search bar, and the current region (United States (N. Virginia)). The main header shows 'ElasticCache' and 'Redis OSS caches'. A left-hand navigation menu lists various resources and configurations. The main content area features a notification banner about Valkey, a search bar for Redis caches, and a table listing existing caches. The table has columns for Cache name, Status, Description, Engine version, Configuration, and Created date. One cache, 'dev-redis', is listed with a status of 'Available'.

Amazon ElasticCache

Dashboard

Resources

- Valkey caches [New](#)
- Memcached caches
- Redis OSS caches**
- Global datastores
- Reserved nodes
- Backups

Configurations

- Subnet groups
- Parameter groups
- User management
- User group management

Events

Service updates

ElasticCache cluster client

[Documentation](#)

[Amazon MemoryDB](#)

Redis OSS caches (1) [Info](#)

[Upgrade to Valkey](#) [View details](#) [Actions](#) [Create Redis OSS cache](#)

	Cache name	Status	Description	Engine version	Configuration	Created date
<input type="radio"/>	dev-redis	Available	-	5.0.6	cache.t2.micro	April 26, 2025, 15:27:4...

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

RDS

ActivitiesPlacesBrowse Web Browser

(2) WhatsAppDatabases | AuroraTerraform labl - CUntitled documentaws_instance | R...+https://us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#databases

CentOSBlogDocument...Forums

awsSearch[Alt+S]

United States (N. Virginia)mostafa_practice@mostafa-aws-practice

Aurora and RDS

Dashboard

Databases

Query Editor

Performance insights

Snapshots

Exports in Amazon S3

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Zero-ETL Integrations [New](#)

Events

Event subscriptions

Recommendations 0

Certificate update

Consider creating a blue/green deployment to minimize downtime during upgrades

You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. [RDS User Guide](#) [Aurora User Guide](#)

Notifications 0 0 0 0 4 0

Databases (1)

Filter by databases

Group resources

Modify

Actions

Create database

DB Identifier

Status

Role

Engine

Region ...

Size

Recommendations

CPU

dev-mysql

Available

Instance

MySQL Co...

us-east-1b

db.t3.micro

18.72%

CloudShellFeedback

© 2025, Amazon Web Services, Inc. or its affiliates. PrivacyTermsCookie preferences