

✅ Steps to Deploy ELB using Terraform

🚀 This project provisions an AWS Classic Load Balancer and spins up EC2 instances behind it. You can see traffic being distributed across the instances through ELB.

🔍 Step 0: Review the Project:

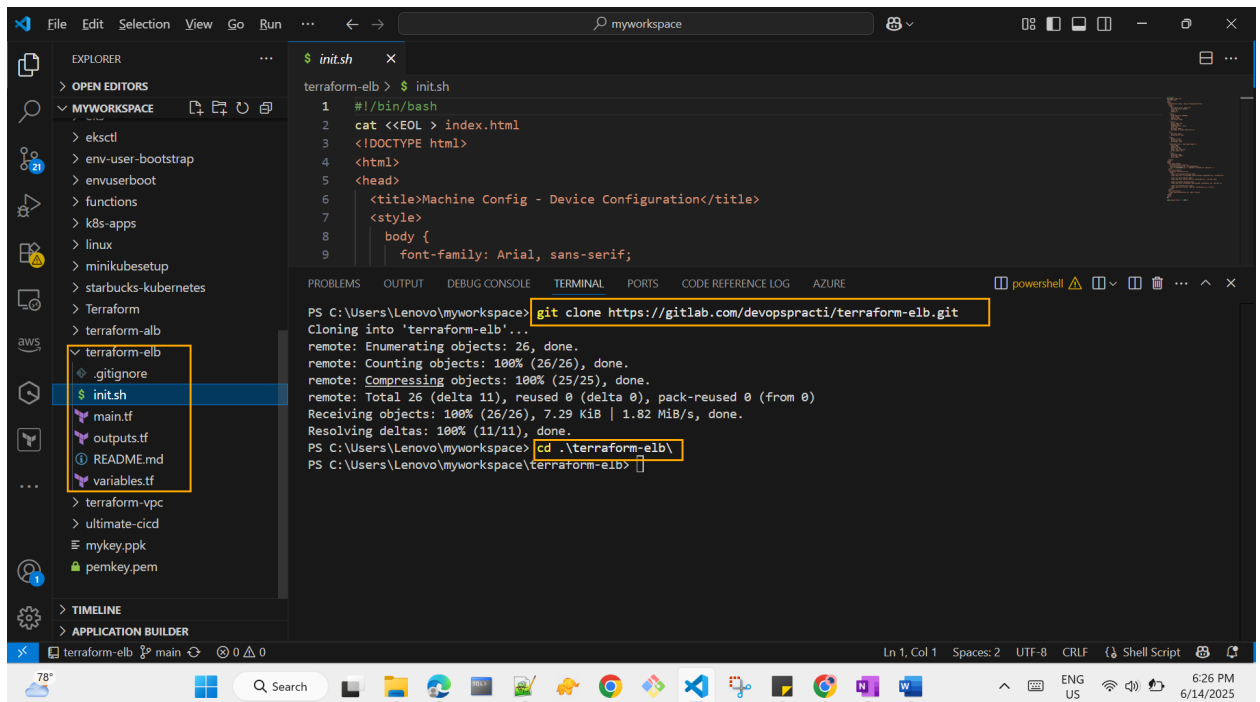
Go to: <https://gitlab.com/shankar-k/terraform-elb>

Understand the structure:

- **main.tf** provisions the ELB and EC2 instances
- **variables.tf** declares reusable inputs
- **outputs.tf** gives the ELB DNS for browser access

📁 Step 1: Clone the Repository:

git <https://gitlab.com/shankar-k/terraform-elb>
cd terraform-elb



The screenshot shows a Visual Studio Code window with the Explorer sidebar on the left and a terminal at the bottom. The Explorer sidebar shows a file tree for a workspace named 'myworkspace'. Under the 'terraform-elb' folder, the following files are listed: `.gitignore`, `init.sh`, `main.tf`, `outputs.tf`, `README.md`, and `variables.tf`. The `init.sh` file is selected and highlighted. The terminal window shows the execution of the `git clone` command to clone the repository from <https://gitlab.com/devopspracti/terraform-elb.git>. The output of the command is visible, showing the cloning process and the directory structure. The terminal prompt is `PS C:\Users\Lenovo\myworkspace>`.

⚙️ Step 2: Initialize Terraform:

terraform init

This will initialize the backend and download required providers.

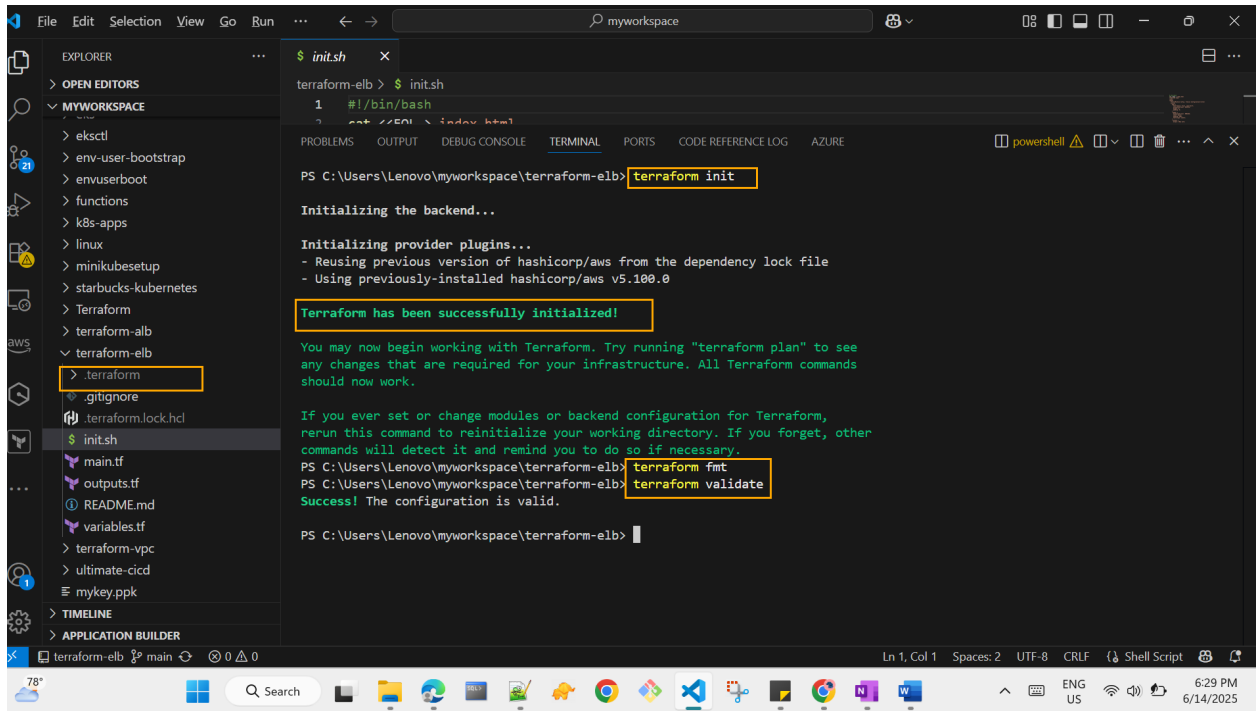
🔧 Step 3: Format and Validate:

terraform fmt

terraform validate

fmt ensures your code is properly formatted.

validate checks for configuration errors.



```
terraform-elb > $ initsh
1 #!/bin/bash
2 cat /dev/null > index.html

terraform-elb > terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.100.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

PS C:\Users\Lenovo\myworkspace\terraform-elb> terraform fmt
PS C:\Users\Lenovo\myworkspace\terraform-elb> terraform validate
Success! The configuration is valid.

PS C:\Users\Lenovo\myworkspace\terraform-elb>
```

Step 4: Apply Terraform:

terraform apply -auto-approve

This command will:

- Create EC2 instances
- Launch a Classic Load Balancer (ELB)
- Attach the instances to the ELB

File Edit Selection View Go Run ... myworkspace

EXPLORER

OPEN EDITORS

MYWORKSPACE

- > eksctl
- > env-user-bootstrap
- > envuserboot
- > functions
- > k8s-apps
- > linux
- > minikubsetup
- > starbucks-kubernetes
- > Terraform
 - > terraform-alb
 - > terraform-elb
 - > .terraform
 - > .gitignore
 - > terraform.lock.hcl
 - > terraform.tfstate.lockinfo
 - > init.sh
 - > main.tf
 - > outputs.tf
 - > README.md
 - > terraform.tfstate
 - > variables.tf
 - > terraform-vpc
- > TIMELINE
- > APPLICATION BUILDER

init.sh

```
terraform-elb > $ init.sh
1 #!/bin/bash
2 terraform apply -auto-approve
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG AZURE

PS C:\Users\Lenovo\myworkspace\terraform-elb> terraform apply -auto-approve

data.aws_availability_zones.all: Reading...

data.aws_availability_zones.all: Read complete after 0s [id=ap-south-1]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

- + create

Terraform will perform the following actions:

```
# aws_autoscaling_group.example will be created
+ resource "aws_autoscaling_group" "example" {
  + arn                               = (known after apply)
  + availability_zones                 = [
    + "ap-south-1a",
    + "ap-south-1b",
  ]
  + default_cooldown                  = (known after apply)
  + desired_capacity                  = (known after apply)
  + force_delete                      = false
  + force_delete_warm_pool            = false
  + health_check_grace_period         = 300
  + health_check_type                 = "ELB"
  + id                               = (known after apply)
  + ignore_failed_scaling_activities = false
  + load_balancers                    = [
    + "itg-elb-example",
  ]
}
```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF (Shell Script) 6:31 PM 6/14/2025

File Edit Selection View Go Run ... myworkspace

EXPLORER

OPEN EDITORS

MYWORKSPACE

- > eksctl
- > env-user-bootstrap
- > envuserboot
- > functions
- > k8s-apps
- > linux
- > minikubsetup
- > starbucks-kubernetes
- > Terraform
 - > terraform-alb
 - > terraform-elb
 - > .terraform
 - > .gitignore
 - > terraform.lock.hcl
 - > init.sh
 - > main.tf
 - > outputs.tf
 - > README.md
 - > terraform.tfstate
 - > variables.tf
 - > terraform-vpc
 - > ultimate-cicd
- > TIMELINE
- > APPLICATION BUILDER

init.sh

```
terraform-elb > $ init.sh
1 #!/bin/bash
2 terraform apply -auto-approve
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG AZURE

PS C:\Users\Lenovo\myworkspace\terraform-elb> terraform apply -auto-approve

data.aws_availability_zones.all: Reading...

data.aws_availability_zones.all: Read complete after 0s [id=ap-south-1]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

- + create

Terraform will perform the following actions:

```
# aws_autoscaling_group.example will be created
+ resource "aws_autoscaling_group" "example" {
  + arn                               = (known after apply)
  + availability_zones                 = [
    + "ap-south-1a",
    + "ap-south-1b",
  ]
  + default_cooldown                  = (known after apply)
  + desired_capacity                  = (known after apply)
  + force_delete                      = false
  + force_delete_warm_pool            = false
  + health_check_grace_period         = 300
  + health_check_type                 = "ELB"
  + id                               = (known after apply)
  + ignore_failed_scaling_activities = false
  + load_balancers                    = [
    + "itg-elb-example",
  ]
}
```

Plan: 5 to add, 0 to change, 0 to destroy.

Changes to Outputs:

```
+ clb_dns_name = (known after apply)
aws_security_group.instance: Creating...
aws_security_group.elb: Creating...
aws_security_group.instance: Creation complete after 2s [id=sg-042d9218cece3c9c]
aws_launch_template.example: Creating...
aws_security_group.elb: Creation complete after 2s [id=sg-05e0cb3f1d4992fd5]
aws_elb.example: Creating...
aws_elb.example: Creation complete after 3s [id=itg-elb-example]
aws_launch_template.example: Creation complete after 6s [id=lt-007419762f1a96da2]
aws_autoscaling_group.example: Creating...
aws_autoscaling_group.example: Still creating... [10s elapsed]
aws_autoscaling_group.example: Creation complete after 16s [id=itg-example-asg]
```

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

Outputs:

```
clb_dns_name = "itg-elb-example-681254676.ap-south-1.elb.amazonaws.com"
```

PS C:\Users\Lenovo\myworkspace\terraform-elb>

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF (Shell Script) 6:31 PM 6/14/2025

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:instanceState=running

EC2 > Instances

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

▼ Auto Scaling

- Auto Scaling Groups

Settings

Instances (2) Info

Last updated less than a minute ago

Connect Instance state Actions Launch instances

Find Instance by attribute or tag (case-sensitive)

Instance state = running Clear filters

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
<input type="checkbox"/>	itg-asg-project	i-0caec9f8ee6d4e808	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1b	ec2-13-
<input type="checkbox"/>	itg-asg-project	i-0cd93ca6860178c35	Running	t2.medium	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-

Select an instance

CloudShell Feedback

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

78°

Search

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#AutoScalingGroups:

EC2 > Auto Scaling groups

EC2

- Dashboard
- 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

Auto Scaling groups (1) Info

Last updated less than a minute ago

Launch configurations Launch templates Actions Create Auto Scaling group

Search your Auto Scaling groups

<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	ap
<input type="checkbox"/>	itg-example-asg	itg-example-launchtemplate	Version Lat 2	-	2	2	10	ap

0 Auto Scaling groups selected

The first screenshot shows the 'Load balancers (1)' page in the AWS Management Console. It displays a table with one load balancer: 'itg-elb-example' with DNS name 'itg-elb-example-681254676.ap-south-1.elb.amazonaws.com'. Below the table, it states '0 load balancers selected'.

The second screenshot shows the 'Target instances' page for the 'itg-elb-example' load balancer. It displays a table with two target instances, both with a health status of 'In-service'.

Instance ID	Name	Health status	Health status description	Security groups
i-0caec9f8ee6d4e808	itg-asg-project	In-service	Not applicable	itg-example-instance-elb
i-0cd93ca6860178c35	itg-asg-project	In-service	Not applicable	itg-example-instance-elb

Step 5: Access the Application via ELB:

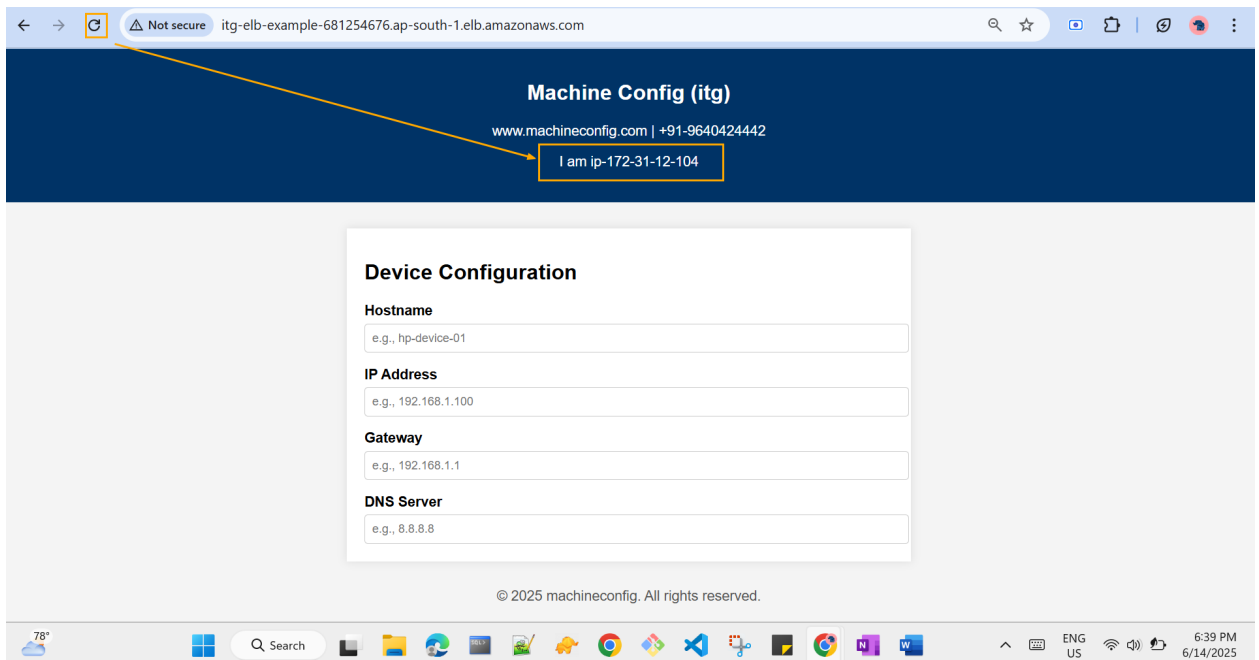
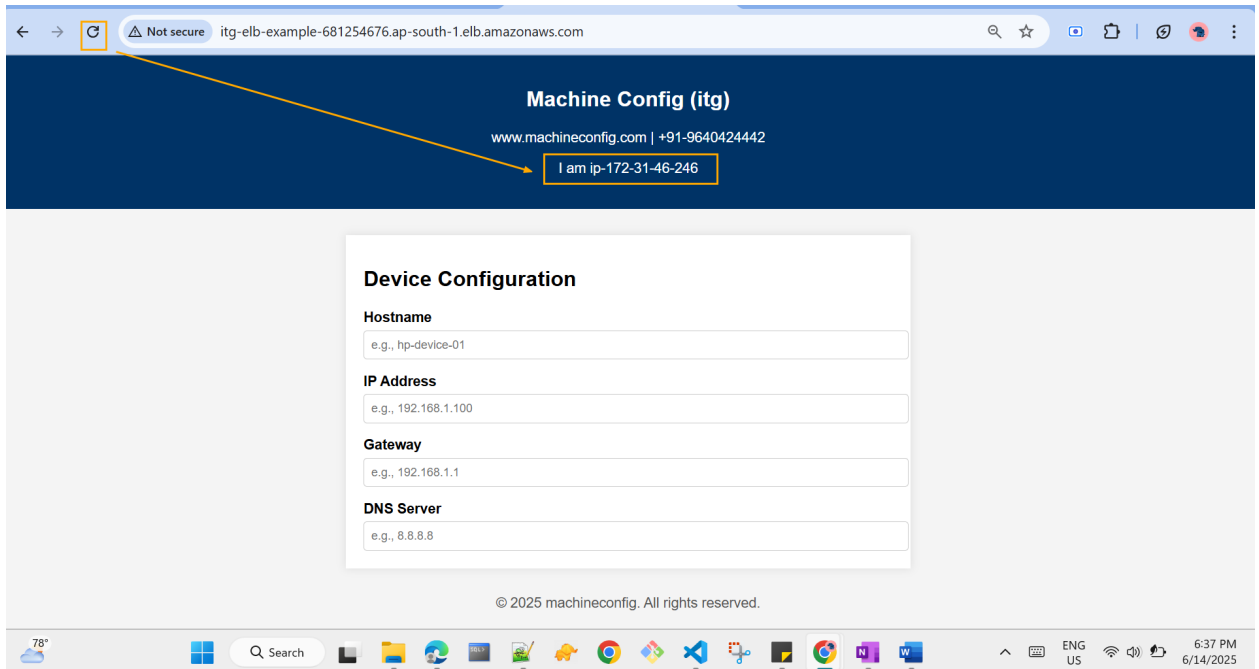
1. After apply completes, note the ELB DNS from the output:

Outputs:

elb_dns = "oneconfig-elb-example-90386787.ap-south-1.elb.amazonaws.com"

2. Open the DNS in your browser:
<http://itg-elb-example-681254676.ap-south-1.elb.amazonaws.com>
3. Refresh the page multiple times to see the IP address change in the

browser. This shows the ELB load balancing traffic across multiple EC2 instances.



Final Outcome

You've now:

- Deployed an ELB and EC2s using Terraform
- Verified load balancing in action
- Followed IaC best practices with validation and formatting