SCENARIO 01 · SINGLE GATEWAY · MARKETPLACE AMI

Deploy a single Enforza gateway on AWS

A minimal, production-shaped landing zone for the Enforza NVA — one VPC, one firewall in a public subnet, and two private workload subnets whose internet-bound traffic is routed through the firewall for secure NAT and L7 egress filtering. Deploy it with Terraform or CloudFormation — including a click-by-click console upload if you'd rather not touch a command line.

◆ Marketplace-AMI variant

The firewall launches from the Enforza AWS Marketplace AMI and self-registers on first boot using the EC2 Instance Identity Document — no bootstrap key or token to supply. Terraform and CloudFormation build the identical architecture.

YOU'LL NEED

- An active Enforza AWS Marketplace subscription and the resulting AMI id for your region (see page 4).
- An AWS account, a region, and credentials configured.
- `Terraform ≥ 1.5` (AWS provider ~> 5.0) or the AWS CLI / Console for CloudFormation.

The architecture

One VPC (`10.0.0.0/16`) with an Internet Gateway. A public subnet hosts the Enforza firewall with an Elastic IP; two private subnets (one per AZ) send `0.0.0.0/0` to the firewall's network interface, so workload egress is filtered and SNAT'd on the way out.

What it deploys, and how it boots

THE RESOURCES

- One VPC (`10.0.0.0/16`) and one Internet Gateway.
- One public subnet (`10.0.0.0/24`, AZ 0) hosting the Enforza firewall with an Elastic IP.
- Two private workload subnets (`10.0.1.0/24` AZ 0, `10.0.2.0/24` AZ 1).
- One Enforza EC2 instance (default `c6i.large`) on the Enforza Marketplace AMI, source/destination check disabled.
- A public route table (`0.0.0.0/0 → IGW`) and two private route tables (`0.0.0.0/0 → the firewall's network interface`).

All resources are tagged `Project=enforza` and `Scenario=01-single-gateway`.

How it registers on first boot

- The instance launches from the Enforza Marketplace AMI — the engine is already baked in.
- On first boot it self-registers with the Enforza cloud using the EC2 Instance Identity Document.
- There is no bootstrap key or token to supply — nothing to paste, nothing to expire.
- Once running, the firewall appears in the console, ready to claim (see page 8).

EXPLANATION — The Marketplace AMI proves its identity to the Enforza cloud with the signed EC2 Instance Identity Document, so no secret changes hands. The networking is identical to the deployment-key variant; only the image and the way the engine registers differ.

PREREQUISITE

Subscribe and get the AMI id

1 Get the Enforza Marketplace AMI id

AWS Marketplace: subscribe to Enforza, then find the AMI id for your target region —

- Ensure your AWS Marketplace subscription is active
- Note the AMI id for the region you'll deploy into; it looks like `ami-xxxxxxxxxxxxxxxxx`
- You'll pass it to the deploy command (or paste it as a stack parameter)

EXPLANATION — The Marketplace AMI is region-specific — the id differs per region, so grab the one that matches where you're deploying. Unlike the deployment-key variant, there's no key to generate: the AMI self-registers on first boot using its signed EC2 Instance Identity Document.

◆ **No key, no token**

This variant has nothing one-time to manage. The only input unique to the firewall is the AMI id for your region.

OPTION A · TERRAFORM

Deploy with Terraform

2 Init, plan, apply

From the repo, work in the `terraform/` directory and pass the Marketplace AMI id as a variable.

bash — terraform

```
$ cd terraform
$ terraform init
$ terraform plan  -var enforza_ami_id=ami-xxxxxxxxxxxxxxxx
$ terraform apply -var enforza_ami_id=ami-xxxxxxxxxxxxxxxx
```

EXPLANATION — `init` downloads the AWS provider, `plan` shows exactly what will be created, and `apply` builds it. The whole VPC, subnets, route tables, firewall instance and Elastic IP come up together; the firewall self-registers on first boot — no key to pass.

3 Optional overrides

Append any of these `-var` flags to `plan` / `apply`:

bash — overrides

```
-var instance_type=c6i.xlarge
-var ssh_key_name=my-key
# CIDRs: vpc_cidr, public_subnet_cidr, private_subnet_a_cidr, private_subnet_b_cidr
```

OPTION B · CLOUDFORMATION (CLI)

Deploy with the AWS CLI

2 Deploy the stack

One self-contained template builds the identical architecture. Pass the AMI id as a parameter override.

bash — aws-cli

```
$ aws cloudformation deploy \
--template-file cloudformation/template.yaml \
--stack-name enforza-single-gateway \
--parameter-overrides EnforzaAmiId=ami-xxxxxxxxxxxxxxxx \
--capabilities CAPABILITY_NAMED_IAM
```

EXPLANATION — `CAPABILITY_NAMED_IAM` is required because the stack creates a named IAM role for the instance. The command uploads the template, creates the stack, and waits — when it returns, the gateway is building. Optional parameters: `InstanceType`, `SshKeyName`, and the CIDR parameters (`VpcCidr`, `PublicSubnetCidr`, `PrivateSubnetACidr`, `PrivateSubnetBCidr`).

◆ **Prefer clicking to typing?**

The next page walks the same template through the AWS Console — download the file, upload it to CloudFormation, and fill the parameters in a form. No command line needed.

OPTION B · CLOUDFORMATION (CONSOLE — SEMI-CLICKOPS)

Upload the template in the AWS Console

1. **Download the template.** Grab `cloudformation/template.yaml` from the scenario repo and save it locally.
2. **Open CloudFormation.** AWS Console → CloudFormation → Create stack → `With new resources (standard)`.
3. **Choose the template.** Prepare template → `Template is ready`. Specify template → `Upload a template file` → Choose file → select `template.yaml` → Next.
4. **Name it and set parameters.** Stack name `enforza-single-gateway`. Under Parameters, paste `EnforzaAmiId = ami-xxxxxxxxxxxxxxxxx`. Leave the rest at defaults (or set `InstanceType`, `SshKeyName`, CIDRs) → Next.
5. **Stack options.** Nothing required — tags are already baked in by the template → Next.
6. **Acknowledge IAM & submit.** On Review, tick "I acknowledge that AWS CloudFormation might create IAM resources with custom names" → Submit.
7. **Watch it build.** The Events tab streams progress to CREATE_COMPLETE. The Outputs tab then shows `FirewallInstanceId` — the instance you'll claim in the console.

EXPLANATION — The acknowledgement is the console's equivalent of the CLI's `--capabilities CAPABILITY_NAMED_IAM` — CloudFormation won't create a named IAM role without your explicit consent. Everything else is the same stack the CLI would deploy, just driven through a form.

AFTER THE DEPLOY

Claim the firewall, and how traffic flows

✓ Claim the firewall

The Marketplace AMI self-registers on first boot using the EC2 Instance Identity Document — no bootstrap key or token. Once the instance is running, claim it in the Enforza console using your AWS account id and the instance id (the `firewall_instance_id` / `FirewallInstanceId` output).

→ How traffic flows end-to-end

- A workload in a private subnet sends traffic to the internet.
- Its private route table sends `0.0.0.0/0` to the firewall's network interface — not to an IGW or NAT gateway.
- The Enforza engine inspects and filters the egress at Layer 7, then SNATs the packet to the firewall's Elastic IP.
- The public route table sends `0.0.0.0/0` to the Internet Gateway, and the packet leaves.
- Return traffic follows the reverse path back through the firewall to the workload.

Source/dest check & teardown

◆ Why source/destination check is disabled

By default EC2 enforces a source/destination check: an instance only accepts packets whose source or destination is its own address, and silently drops everything else. A NAT/forwarding appliance exists precisely to handle packets addressed to other hosts (the private workloads). Disabling the check (`source_dest_check = false` in Terraform, `SourceDestCheck: false` in CloudFormation) is what permits the instance to forward those packets. Without it, the gateway would drop all transit traffic and nothing would reach the internet.

TEARDOWN — STOP THE METER

Remove everything when you're done. Then delete the firewall from the Enforza console.

bash — teardown

```
# Terraform
$ cd terraform
$ terraform destroy -var enforza_ami_id=ami-xxxxxxxxxxxxxxxxx

# CloudFormation
$ aws cloudformation delete-stack --stack-name enforza-single-gateway
```

✓ Done

Same architecture whichever path you took — one gateway doing secure NAT and L7 egress filtering for two private workload subnets. To scale out, add private subnets and point their route tables at the same firewall ENI, or deploy another gateway.