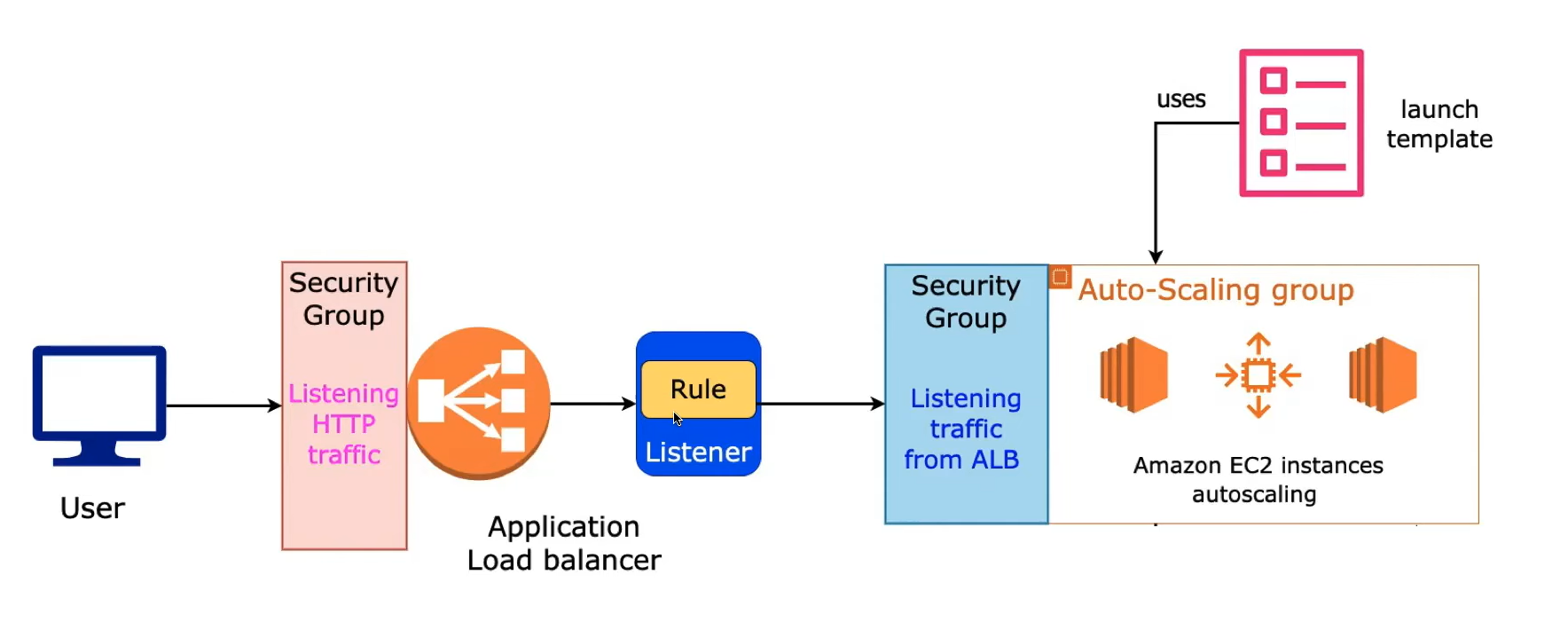
**Deploy a Pre-Built Dockerized Application**

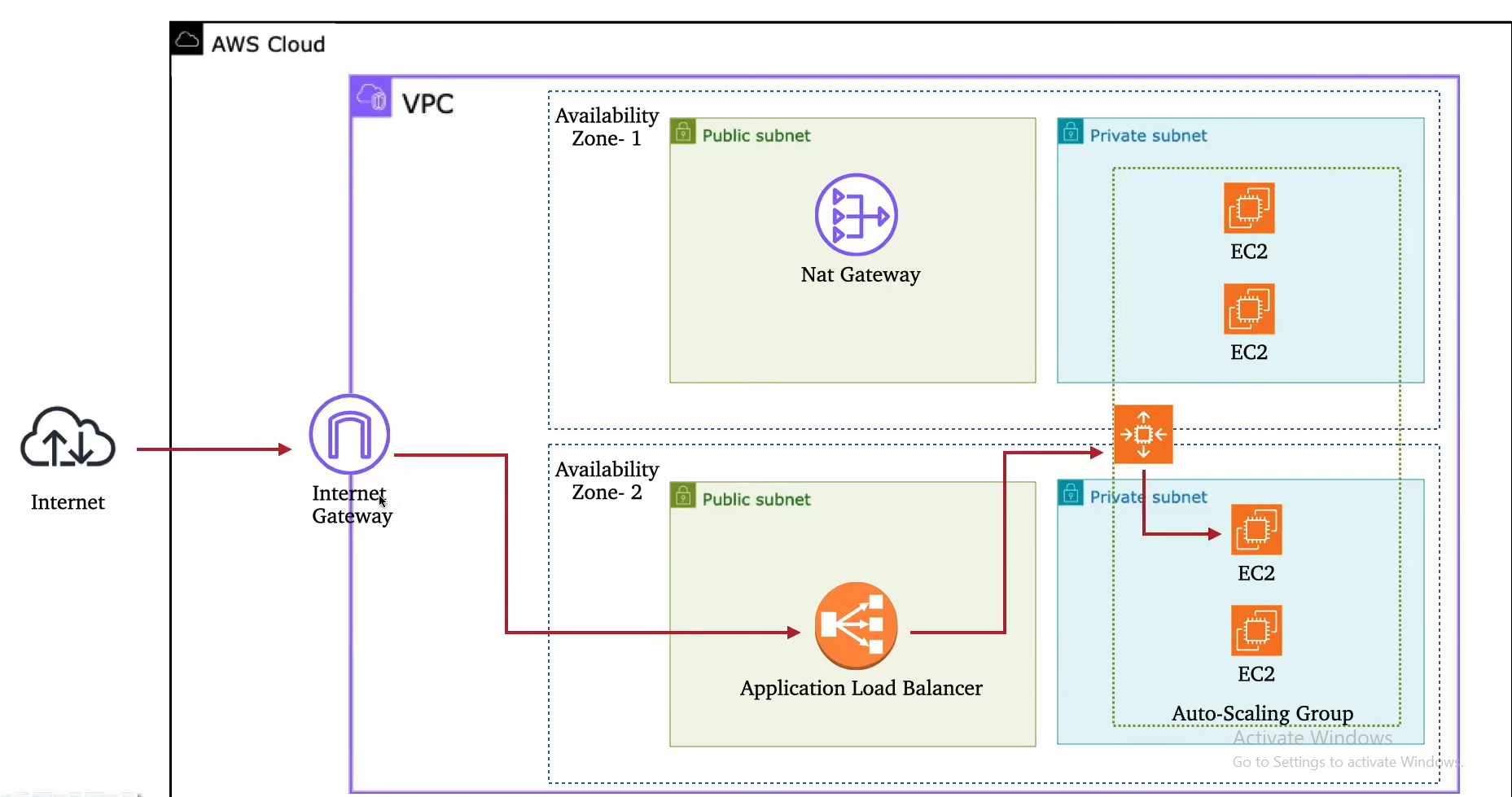
**High Level Architecture based on requirement:**

First we will use launch template to configure and launch EC2 instances within the auto scaling group this template ensures consistency and simplifies the management, then next we will create application load balancer with a dedicated security group that allowed internet traffic. We will then set up a listener for load balancer defining the rule that determines how it routes request to our ec2 instances. Finally we will create a separate security group for auto scaling group, which we will be for underline EC2 instances which will only allow traffic that is originating from the application load balancer keeping things secure.



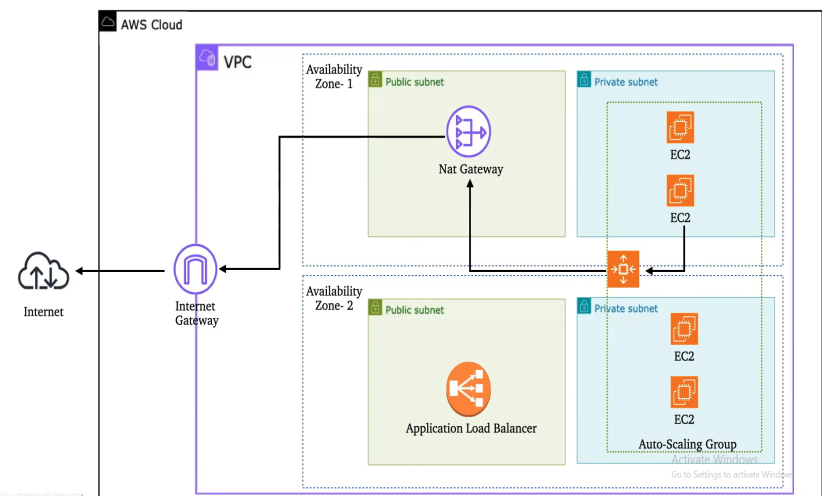
**Break down of the High level Architecture 1:**

First create VPC with public and private subnet spread across to a availability zones for redundancy with in the public subnet. Then we will deploy an internet gateway and Nat gateway in one of the public subnet internet gateway will allow internet traffic to the resources in the public subnet this Nat gateway allow private subnet resources to access the internet securely. After that we will create and deploy auto scaling group in a private subnet for security auto scaling group will launch ec2 instances, so in the ec2 instances, we will deploy required application and deploy a simple HTML test application which will return a response to confirm everything is working. Once that is done we will create application load balancer in the public subnet making it accessible from the internet and it will efficiently distribute incoming request to the ec2 instances with in the auto scaling group.



**Break down of the High level Architecture 2:**

If an EC2 instance need to install dependencies or any updates then it will use Nat gateway for secure internet access like this from the outside world user will access your application through internet gateway. So this gateway then forward request to the application load balancer then it will smartly distribute the workload among the EC2 instances in your auto scaling group.



Below is the step-by-step documentation to create above requirement infrastructure using Terraform.

This documentation will guide you through the setup and deployment of the infrastructure based on the defined files.

Here, I used main.tf and provider.tf and vpc.tf files to configure the setup.

**Step 1: Prerequisites**

Install Terraform: Ensure Terraform is installed on your system. You can download it from the official Terraform website.

AWS Account: You need an AWS account with the necessary permissions to create resources like VPC, EC2 instances, ALB, etc.

AWS CLI: Install and configure the AWS CLI with your credentials using aws configure.

**Step 2: Set Up Terraform Configuration Files**

Following are the Terraform configuration files:

provider.tf: Defines the AWS provider and credentials.

vpc.tf: Defines the VPC, subnets, internet gateway, NAT gateway, and route tables.

main.tf: Defines the security groups, ALB, EC2 launch template, auto-scaling group, and related resources.

**Step 3: Initialize Terraform**

Open a terminal and navigate to the directory containing your .tf files and run the following command to initialize Terraform:

* terraform init

So when we run the above command this command downloads the necessary provider plugins (e.g., AWS, libraries..etc) and sets up the working directory.

**Step 4: Review the Configuration**

Before applying the configuration, review the resources defined in each file:

**1. provider.tf**

Review the AWS provider and region and other details.

**2. vpc.tf**

VPC: Creates a VPC with the CIDR block 10.0.0.0/16.

Subnets: Creates public and private subnets in two availability zones (ap-south-1a and ap-south-1b).

Internet Gateway: Attaches an internet gateway to the VPC for public subnet internet access.

NAT Gateway: Creates a NAT gateway in the public subnet to allow private subnet instances to access the internet.

Route Tables: Configures route tables for public and private subnets.

**3. main.tf**

Security Groups:

alb\_sg: Allows HTTP traffic (port 80) from the internet to the ALB.

ec2\_sg: Allows all traffic from the ALB to EC2 instances.

Application Load Balancer (ALB):

Creates an ALB in the public subnets.

Configures a target group and listener to forward HTTP traffic to EC2 instances.

EC2 Launch Template:

Defines an EC2 launch template with an AMI and instance type (t2.micro).

Installs Docker and runs a simple web application using a Docker container.

Auto Scaling Group (ASG):

Creates an ASG with 2 desired instances, scaling between 2 and 3 instances.

Associates the ASG with the ALB target group.

**Step 5: Standard Terraform style**

* terraform fmt

This command will help you Terraform style conventions. This ensures consistency in code formatting across teams and makes it easier to read and maintain.

**Step 6: Validate the Configuration**

Run the following command to validate the Terraform configuration:

* terraform validate

This command checks for syntax errors and ensures the configuration is valid.

**Step 7: Plan the Deployment**

Run the following command to generate an execution plan:

* terraform plan

This command shows the resources Terraform will create, modify, or delete. Review the plan to ensure it matches your expectations.

**Step 8: Apply the Configuration**

To create the infrastructure, run:

* terraform apply

Terraform will prompt you to confirm the action. Type yes to proceed.

**Step 9:** **Validate** **Deployed Infrastructure Steps**

Step 9.1: Create a VPC

* A VPC named yt-vpc with the CIDR block 10.0.0.0/16 will be created.
* DNS support and DNS hostnames are enabled.

Step 9.2: Create Subnets

* Public Subnets: Two public subnets will be created in the specified availability zones (ap-south-1a and ap-south-1b).
* Private Subnets: Two private subnets will also be created in the same availability zones.

Step 9.3: Create Internet Gateway

* An Internet Gateway (YT-Internet Gateway) will be attached to the VPC to allow internet access for the public subnets.

Step 9.4: Create Route Tables

* Public Route Table: A route table for public subnets will be created, with a default route (0.0.0.0/0) pointing to the Internet Gateway.
* Private Route Table: A route table for private subnets will be created, with a default route (0.0.0.0/0) pointing to the NAT Gateway.

Step 9.5: Create NAT Gateway

* An Elastic IP will be allocated for the NAT Gateway.
* A NAT Gateway (YT-Nat Gateway) will be created in one of the public subnets to allow private subnets to access the internet.

Step 9.6: Create Security Groups

* ALB Security Group: A security group (yt-alb-sg) will be created to allow inbound HTTP traffic (port 80) from the internet and allow all outbound traffic.
* EC2 Security Group: A security group (yt-ec2-sg) will be created to allow all traffic from the ALB security group and allow all outbound traffic.

Step 9.7: Create Application Load Balancer (ALB)

* An Application Load Balancer (yt-app-lb) will be created in the public subnets.
* A target group (yt-web-server-tg) will be created for the ALB to route traffic to EC2 instances on port 80.
* An ALB listener will be configured to forward HTTP traffic (port 80) to the target group.

Step 9.8: Create Launch Template for EC2 Instances

* A launch template (yt-web-server) will be created with the following configuration:
* AMI ID: ami-013e83f579886baeb
* Instance Type: t2.micro
* Security Group: yt-ec2-sg
* User Data: A script to install Docker, pull a Docker image (yeasy/simple-web), and run it on port 80.

Step 9.9: Create Auto Scaling Group (ASG)

* An Auto Scaling Group (yt-web-server-asg) will be created with the following configuration:
* Desired Capacity: 2
* Minimum Size: 2
* Maximum Size: 3
* Target Group: yt-web-server-tg
* Subnets: Private subnets
* Health Check Type: EC2

**Step 10: Verify the Output of the Deployment**

ALB DNS Name: After the deployment, Terraform will output the DNS name of the ALB. We can access the application using this DNS name in your browser.

Example Output: alb\_dns\_name = "yt-app-lb-1234567890.ap-south-1.elb.amazonaws.com"

**AWS Management Console:** Finally Log in to the AWS Management Console to verify the resources:

Check the VPC, subnets, and route tables in the VPC section.

Verify the ALB, target group, and EC2 instances in the EC2 section.

Confirm the auto-scaling group and launch template in the Auto Scaling section.