Problem

Running infrastructure on cloud platforms can be expensive, particularly when relying exclusively on On-Demand instances for workloads that don't require constant availability. Balancing performance and cost-efficiency is challenging, as managing a mix of instance types and pricing models can become complex and time-consuming.

Solution

We use Terraform to automate the provisioning and management of AWS Spot Instances, which helps optimize costs while maintaining the required levels of availability and performance. By creating an Auto Scaling group that utilizes a mix of On-Demand and Spot Instances, we strike a balance between cost savings and resource availability.

Implementation

1. AWS Provider Configuration

We define the AWS provider to manage resources in a specific region.

```
provider "aws" {
  region = "us-west-2"
}
```

---

2. Launch Template

We create a launch template to define the configuration for instances, including the AMI and instance type.

```
resource "aws_launch_template" "example" {
  name_prefix   = "example-template"
  image_id      = "ami-0c55b159cbfafe1f0" # Amazon Linux 2 AMI
  instance_type = "t3.micro"
  # Additional configurations (e.g., security groups, IAM roles) can be added
here
}
```

---

3. Auto Scaling Group

We create an Auto Scaling group with a **mixed instances policy** that balances On-Demand and Spot Instances. The policy specifies:

- Base On-Demand capacity for guaranteed availability.
- Spot Instances for cost savings, using the **capacity-optimized** strategy to prioritize instance availability.

```
resource "aws_autoscaling_group" "example" {
  name                = "example-asg"
  vpc_zone_identifier = ["subnet-12345678", "subnet-87654321"]
  min_size            = 2
  max_size            = 10
  desired_capacity    = 4

  mixed_instances_policy {
    launch_template {
      launch_template_specification {
        launch_template_id = aws_launch_template.example.id
        version            = "$Latest"
      }

      override {
        instance_type     = "t3.micro"
        weighted_capacity = "1"
      }
      override {
        instance_type     = "t3.small"
        weighted_capacity = "2"
      }
    }

    instances_distribution {
      on_demand_base_capacity                  = 1
      on_demand_percentage_above_base_capacity = 25
      spot_allocation_strategy                 = "capacity-optimized"
    }
  }

  tag {
    key                 = "Name"
    value               = "ASG-Instance"
    propagate_at_launch = true
  }
}
```

Discussion

Deploying a cost-optimized infrastructure using Terraform and AWS Spot Instances provides several benefits:

1. **Cost Savings**
   By leveraging Spot Instances, we significantly reduce infrastructure costs compared to using only On-Demand Instances.
2. **Automated Management**
   Terraform automates the creation and management of Auto Scaling groups, reducing manual effort and minimizing the risk of errors.
3. **Flexibility**
   The mixed instances policy enables workloads to run on the most cost-effective instance types available at any given time.
4. **Availability**
   Maintaining a base capacity of On-Demand Instances and using the **capacity-optimized** strategy for Spot Instances ensures a balance between cost savings and application availability.
5. **Scalability**
   The Auto Scaling group dynamically adjusts instance counts based on demand, ensuring performance while optimizing costs.

Best Practices

When optimizing costs with Terraform and Spot Instances, we follow these best practices:

- **Instance Diversity:**
  Use a variety of instance types and sizes to increase the chances of obtaining Spot Instances and improve application resilience.

- **Spot Instance Handling:**
  Design applications to handle Spot Instance interruptions gracefully. For example, use the two-minute termination notice to complete tasks or save data before an instance is terminated.
- **Monitoring and Alerting:**
  Implement CloudWatch alarms to monitor the Auto Scaling group and alert on capacity issues or other potential problems.
- **Regular Review:**
  Periodically evaluate the instance type mix and the balance between On-Demand and Spot Instances to ensure alignment with performance needs and cost goals.
- **Spot Instance Data Feed:**
  Enable the Spot Instance data feed to gain insights into usage patterns and cost savings.
- **Reserved Instances:**
  For predictable, long-term workloads, consider using a mix of Reserved, On-Demand, and Spot Instances for optimal cost efficiency.
- **Auto Scaling Group Metrics:**
  Enable group metrics to track performance and cost efficiency in detail.

Summary

Using Terraform to manage an Auto Scaling group with mixed On-Demand and Spot Instances allows us to significantly reduce costs while maintaining availability and scalability. By leveraging Terraform’s automation capabilities and AWS's Spot Instances, we optimize infrastructure for both performance and cost-efficiency. This approach ensures that our applications remain resilient and cost-effective.