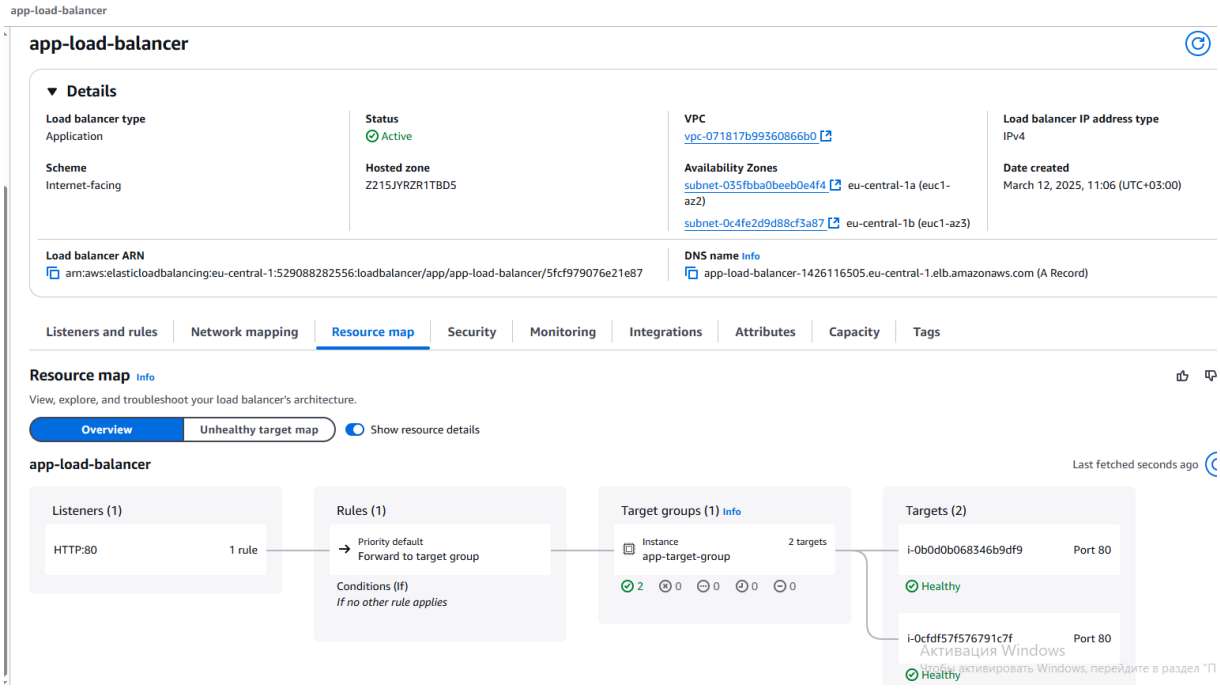
Homework_Lesson33_Report

"1. Создайте Load Balancer в Amazon Web Services/GCP с использованием AWS CLI, terraform, python
2. Настройте маршрутизацию трафика для вашего приложения с использованием Elastic Load Balancer и Auto Scaling.
3. Используя Amazon Route 53/DuckDNS, зарегистрируйте доменное имя для вашего
приложения (NGINX, Apache на выбор) и настройте DNS-записи для обеспечения его доступности
в Интернете.
4. Получите сертификат и настройте работу вашего Apache/NGINX по зашифрованному соединению.
5.* Изучите документацию AWS по настройке маршрутизации трафика в Route 53 и настройте различные типы маршрутизации, такие как
весовая маршрутизация, маршрутизация на основе геоположения и
другие.
6.* Используйте AWS CLI для автоматизации настройки DNS-записей и маршрутизации трафика в Route 53.
7. Реализуйте функцию автоматического масштабирования для вашего приложения, используя Amazon EC2 Auto Scaling и Elastic Load Balancer/GCP Load Balancing. Масштабирование необходимо проверить и сделать скриншоты, которые бы показывали их применение

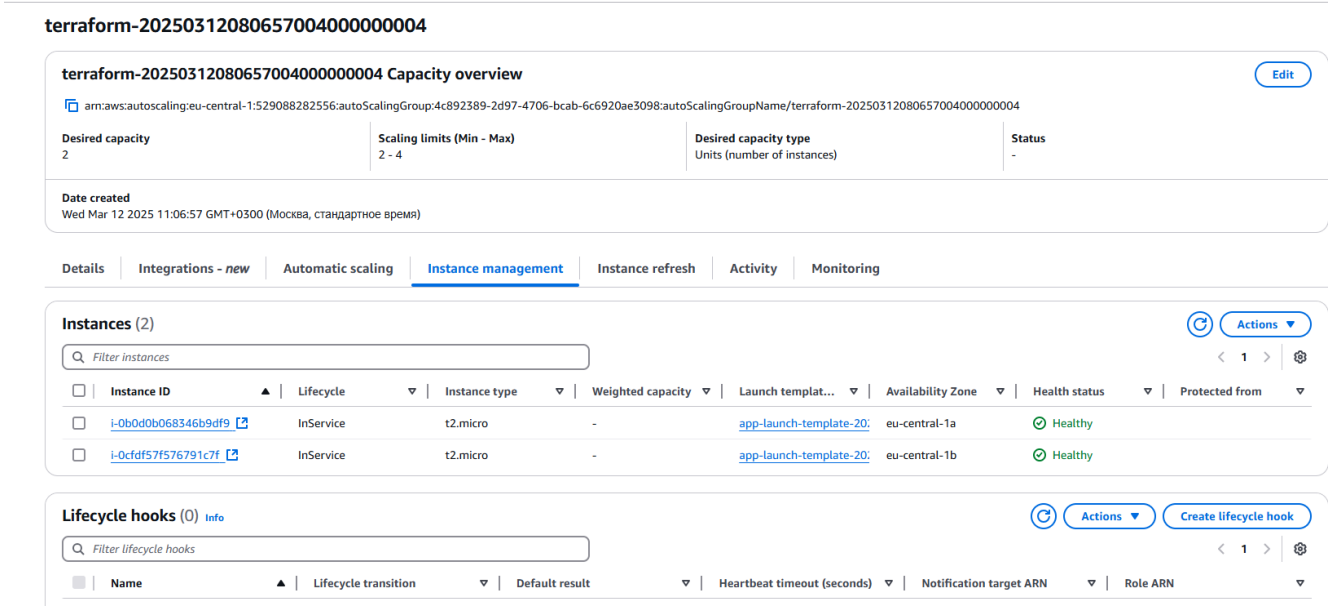
Запускаем конфиг терраформа.

```
PS C:\Users\Admin> cd D:\Terraform\Cloud3
PS D:\Terraform\Cloud3> terraform apply

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.app will be created
  + resource "aws_autoscaling_group" "app" {
      + arn                       = (known after apply)
      + availability_zones        = (known after apply)
      + default_cooldown          = (known after apply)
      + desired_capacity          = 2
      + force_delete              = true
      + force_delete_warm_pool    = false
      + health_check_grace_period = 300
      + health_check_type         = "EC2"
      + id                        = (known after apply)
      + ignore_failed_scaling_activities = false
      + load_balancers            = (known after apply)
      + max_size                  = 4
      + metrics_granularity       = "1Minute"
      + min_size                  = 2
      + name                      = (known after apply)
      + name_prefix               = (known after apply)
      + predicted_capacity        = (known after apply)
      + protect_from_scale_in     = false
      + service_linked_role_arn   = (known after apply)
      + target_group_arns         = (known after apply)
      + vpc_zone_identifier       = (known after apply)
      + wait_for_capacity_timeout = "0"
      + warm_pool_size            = (known after apply)

      + availability_zone_distribution (known after apply)

      + launch_template {
          + id      = (known after apply)
          + name    = (known after apply)
          + version = "$Latest"
        }

      + mixed_instances_policy (known after apply)

      + tag {
          + key                 = "Name"
          + propagate_at_launch = true
          + value               = "nginx"
        }

      + traffic_source (known after apply)
    }

  # aws_internet_gateway.my_internet_gateway will be created
  + resource "aws_internet_gateway" "my_internet_gateway" {
```

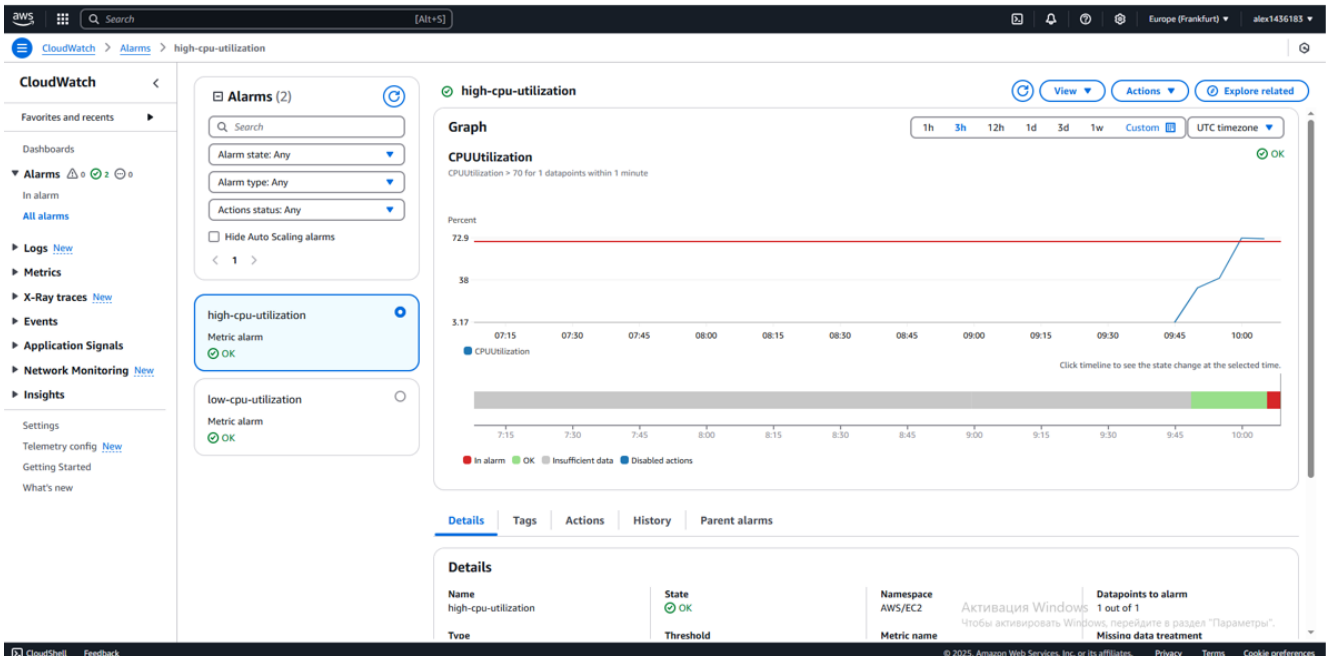
Наш созданный Load balancer.



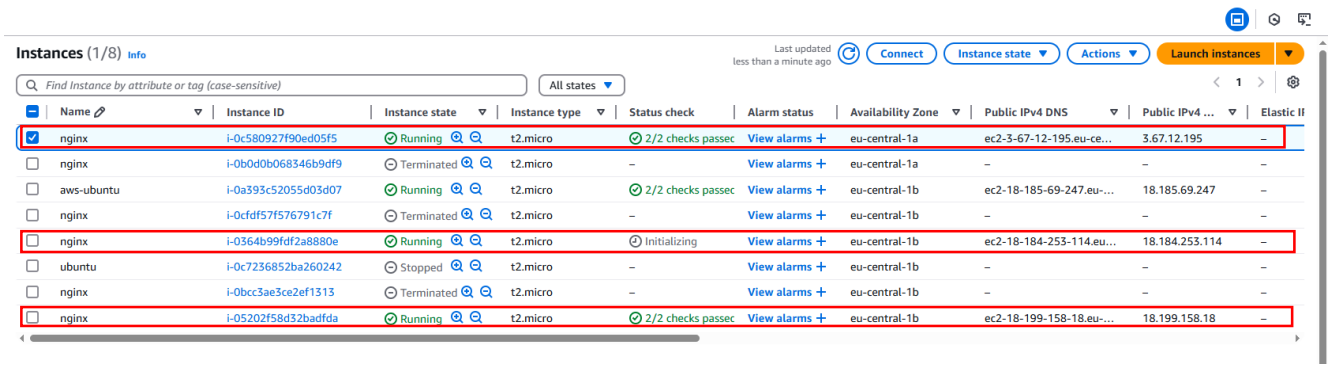
Созданная Auto scaling groups. Минимальное количество истесав 2, максимальное количество 4.



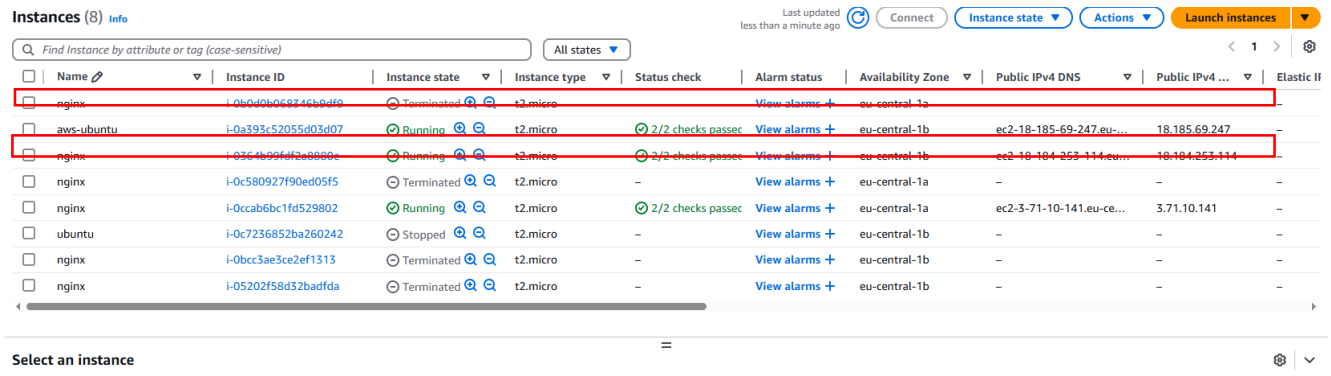
Загрузим наши cpu на одном из наших инстансов.



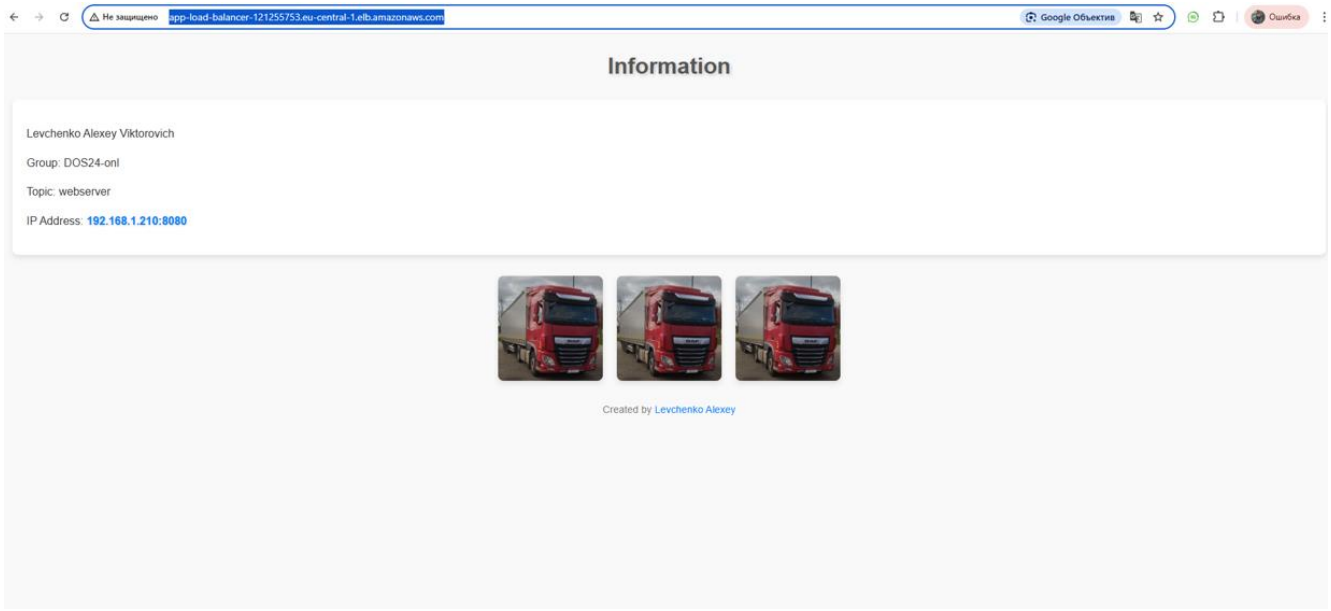
В итоге у нас теперь 3 инстенса.



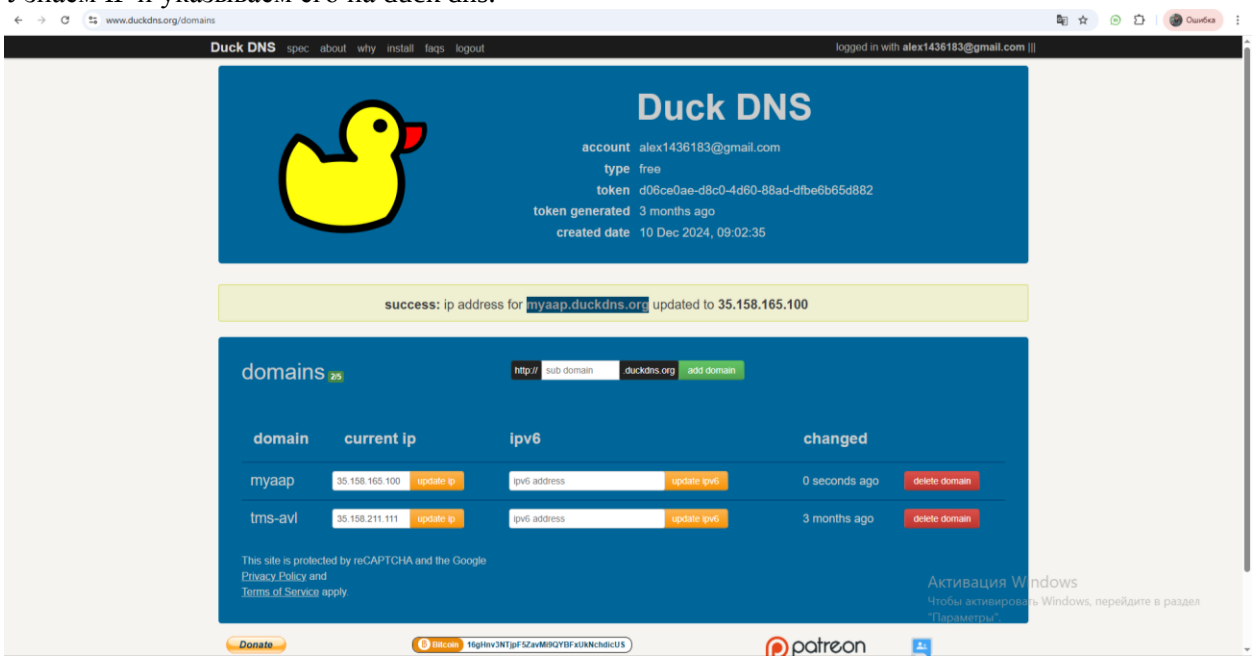
После того как загрузка снизилась ниже 30 % и оставалась такой в течении минуты у нас сново осталось 2 инстенса.



Переходим по ссылке на load balanser.



Узнаем IP и указываем его на duck dns.



Переходим по ссылке и проверяем.