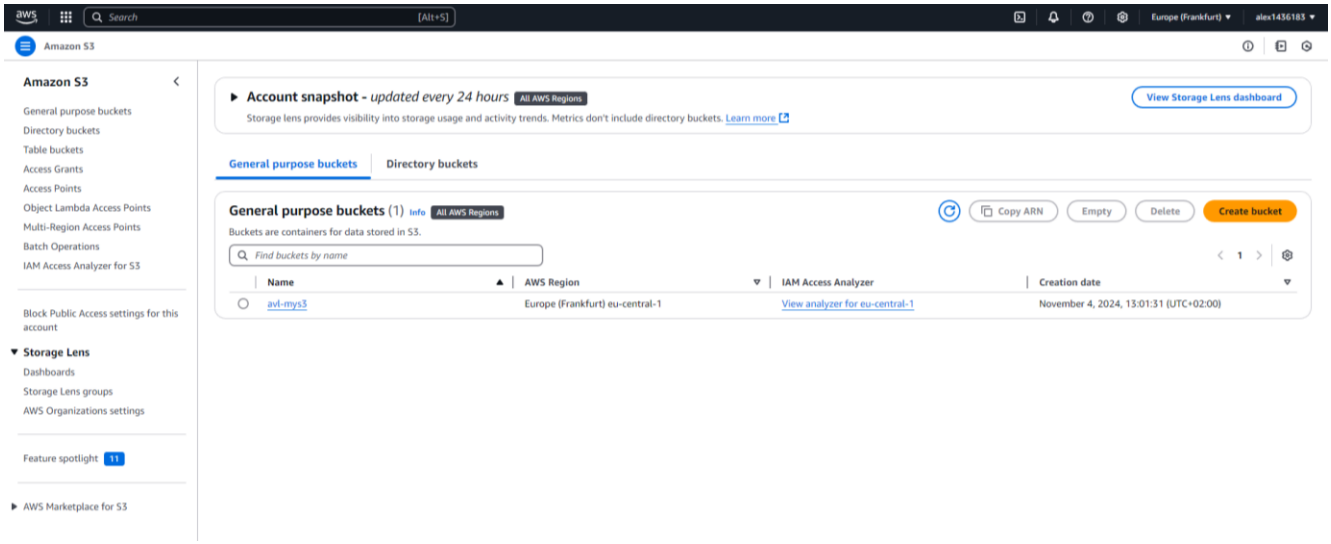
Homework_Lesson31_Report

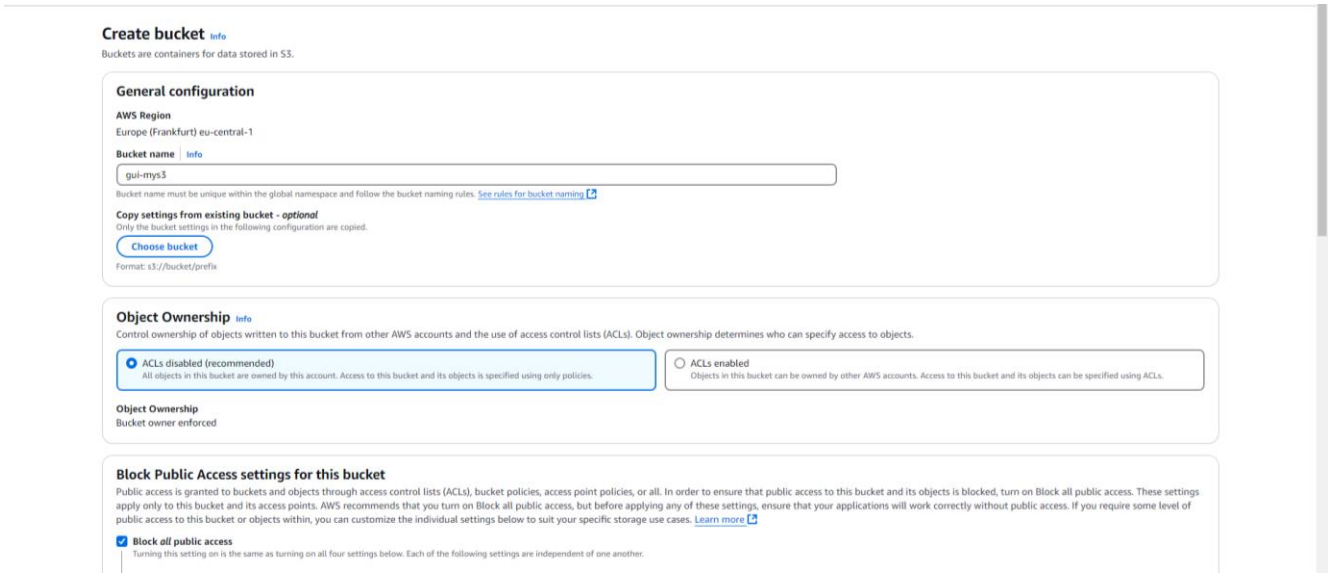
Задание:
"1. Создайте новый бакет Amazon S3/GCP Cloud Storage тремя способами: через GUI (консоль в браузере), с помощью CLI, конфиг в terraform.
2. Создайте Amazon EC2/GCP Compute Engine и настройте масштабирование, чтобы автоматически добавлять новые экземпляры в случае увеличения нагрузки. Сделайте это тремя способами: через GUI (консоль в браузере) с помощью CLI, конфиг в terraform.
3.* Создайте базу через GUI (консоль в браузере), с помощью CLI, конфиг в terraform данных Amazon RDS/GCP CloudSQL и подключитесь к ней из виртуальной машины, которую вы создали ранее. Сделайте это тремя способами: через GUI (консоль в браузере) с помощью CLI, конфиг в terraform."

Задание 1

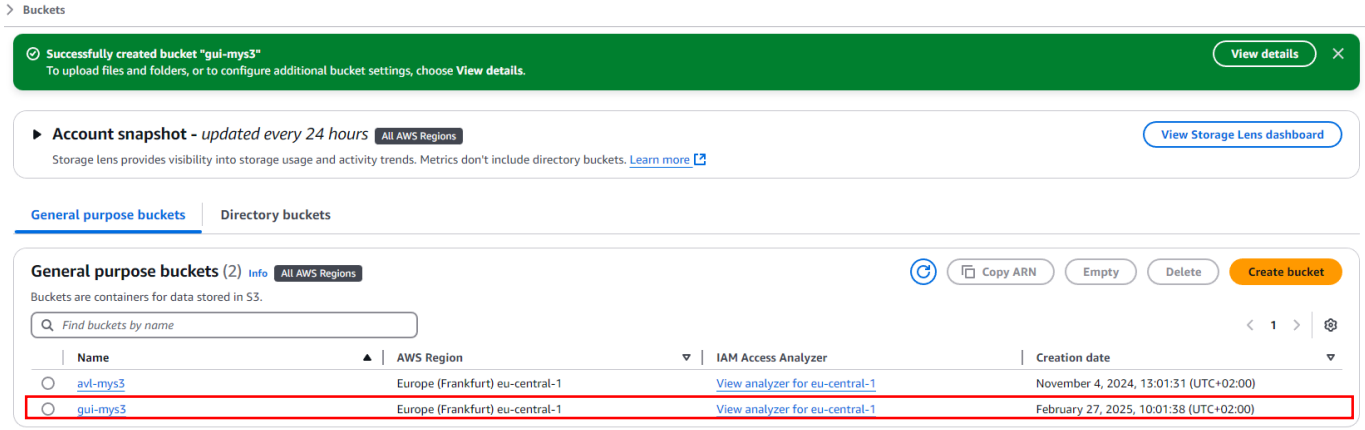
Создайте новый бакета через GUI.
Заходим в AWS S3 и нажимем Create bucket.



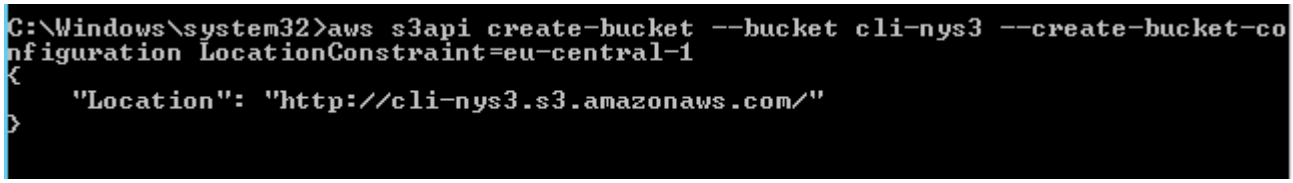
Заполяем имя бакета и необходимые нам параметры и нажимаем Create bucket.



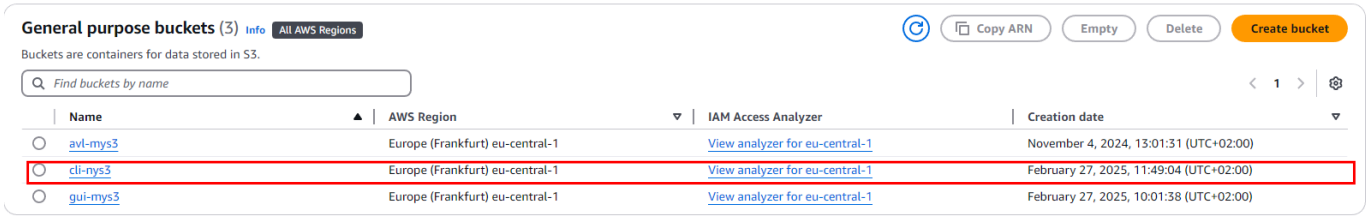
Новый бакет создан.



Создадим через CLI
CLI у нас уже установлен и сконфигурирован.



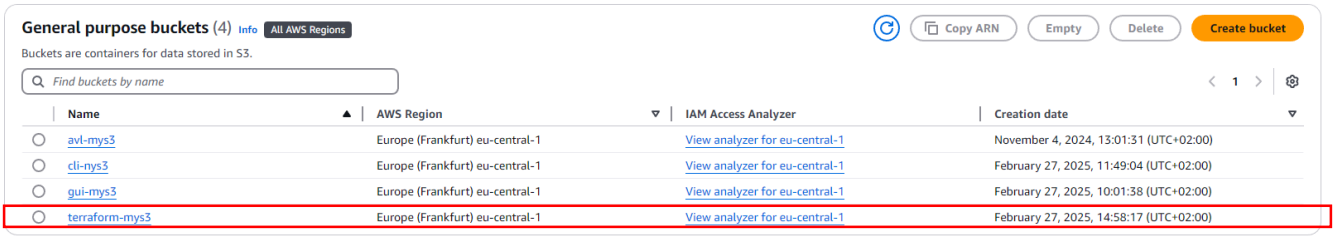
Новый бакет создан.



Создаем с помощью терформа.
Конфиг тераформа прикреплен отдельно.

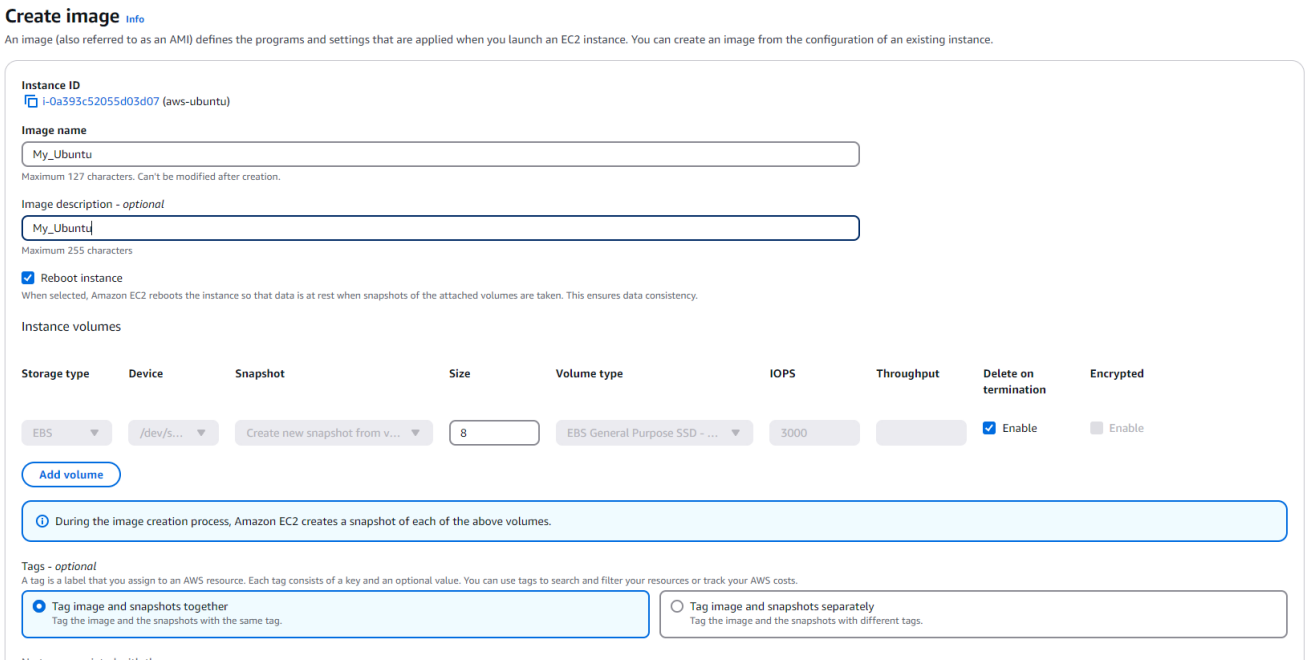
```
PS D:\Terraform> terraform apply

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_s3_bucket.my_bucket will be created
  + resource "aws_s3_bucket" "my_bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "terraform-mys3"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags_all                    = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

Новый бакет создан.

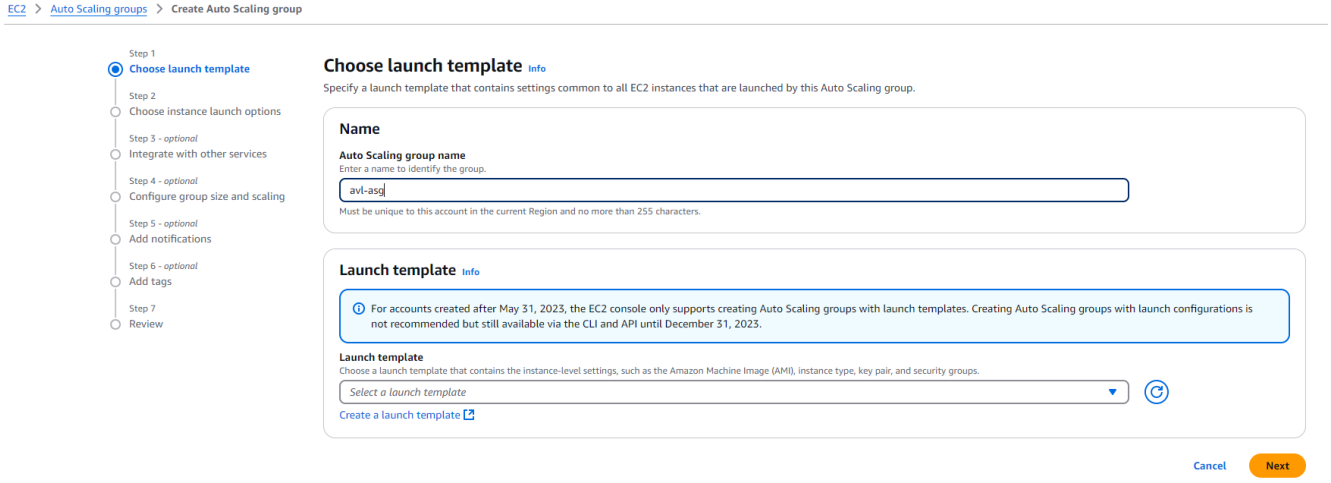


Задание 2

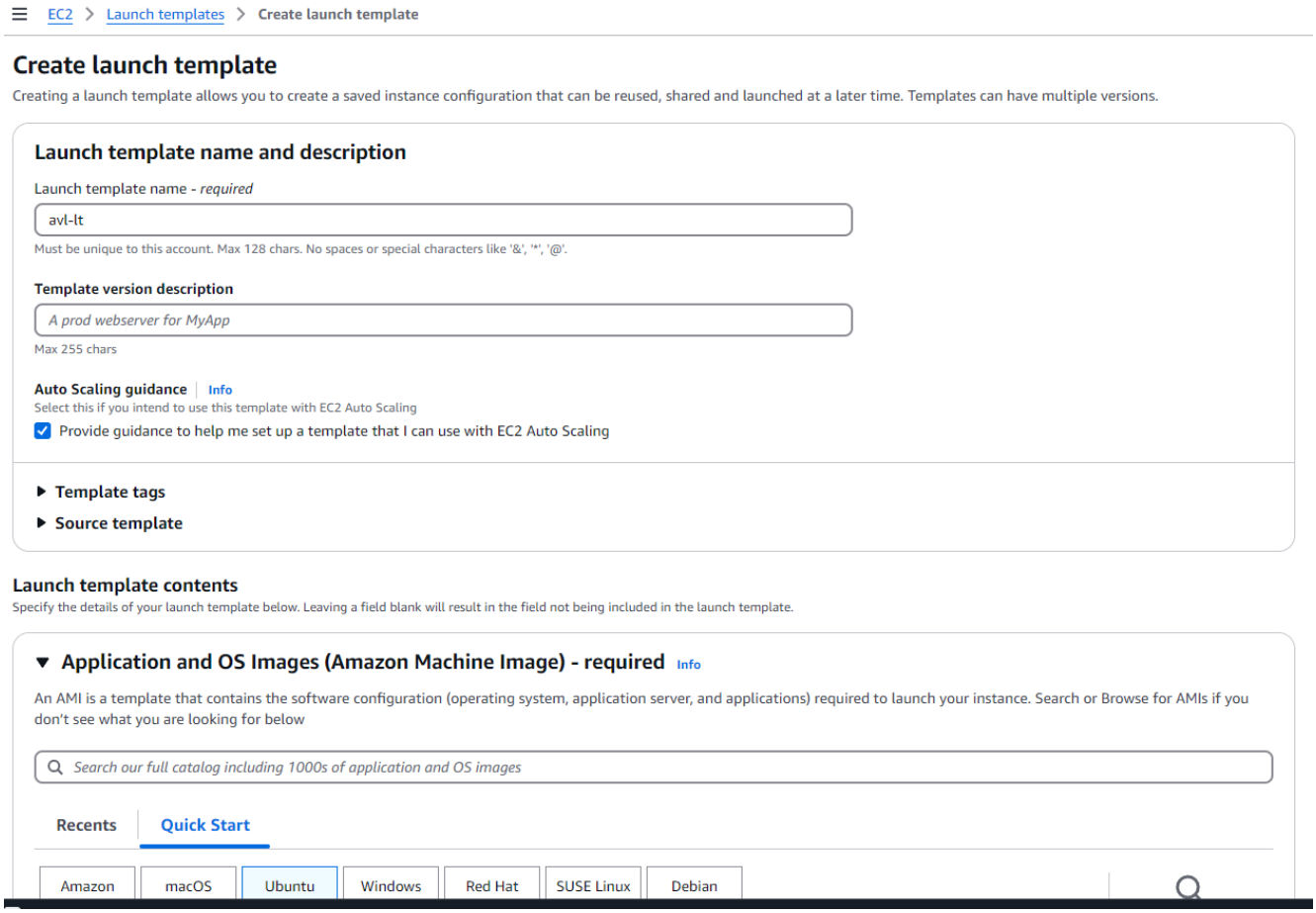
Инстенс на EC 2 у сзданный через GUI уже есть. Зайдем в него и нажемем Actions > Image and Templates > Create Image и создадим image нашей машины.



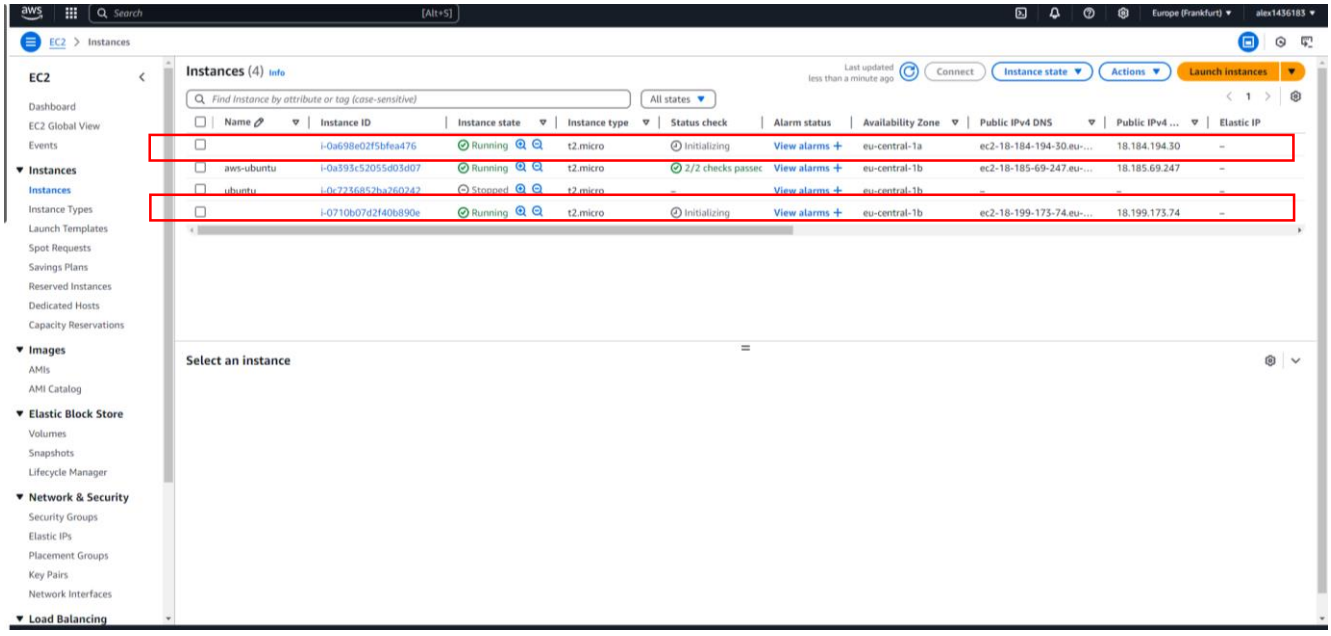
Создаем Auto Scaling group.



Создаем Launch template если у нас нету подходящего ранее созданного.

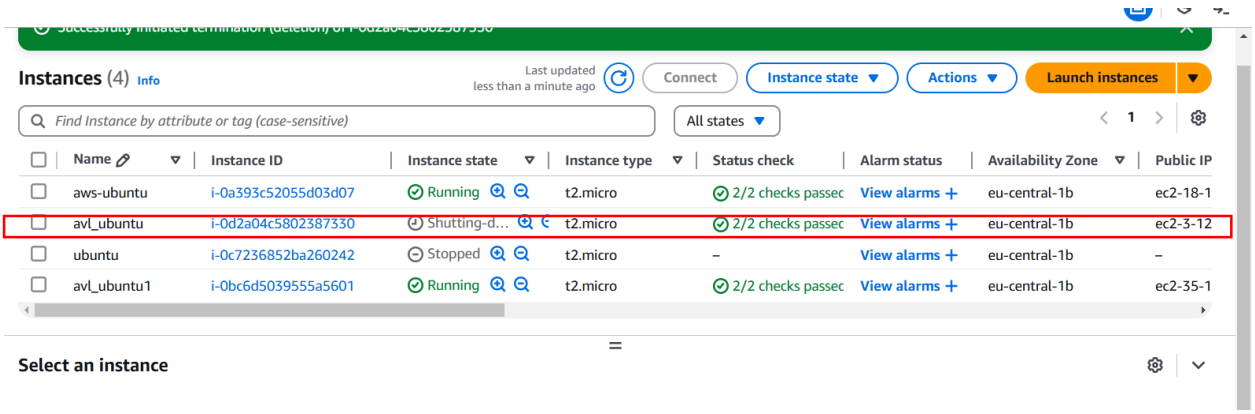


По итогу создания Auto Scaling group у нас запустилось 2 истенса так как мы указали при создании Desired capacity 2.



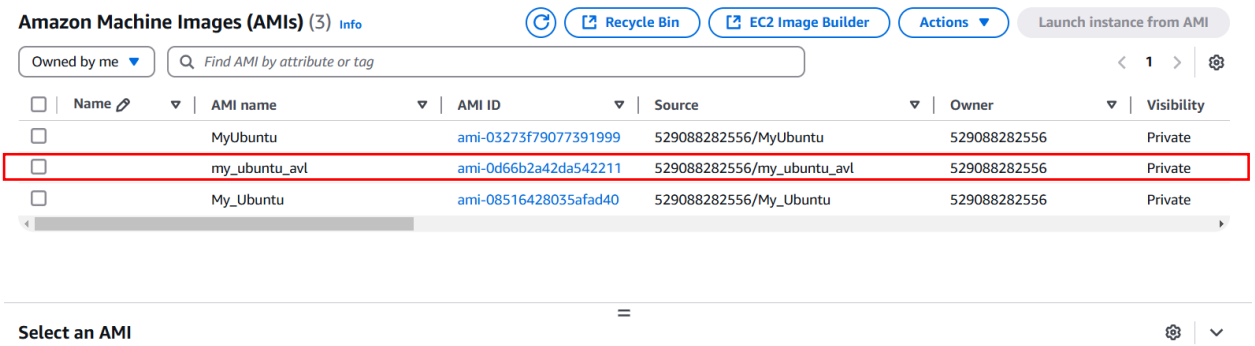
Создаем с помощью CLI Интстанс.

```
PS D:\Terraform> aws ec2 run-instances --image-id ami-0084a47cc718c111a --count 1 --instance-type t2.micro --key-name aws-1 --security-group-ids sg-0c0035be96ba711b7 --subnet-id subnet-0e8f515b5b7d3774
ecifications 'ResourceType=instance,Tags=[{Key=Name,Value=avl_ubuntu1}]'
{
    "ReservationId": "r-0dc65d81c690811b1",
    "OwnerId": "529088282556",
    "Groups": [],
    "Instances": [
        {
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "af8f6a76-61c7-48e7-9bff-284bca1f7a0b",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2025-02-28T10:41:59+00:00",
                        "AttachmentId": "eni-attach-0cfbf250a36d83ffe",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "Status": "attaching",
                        "NetworkCardIndex": 0
                    },
                    "Description": "",
                    "Groups": [
                        {
                            "GroupId": "sg-0c0035be96ba711b7",
                            "GroupName": "launch-wizard-1"
                        }
                    ],
                    "Ipv6Addresses": [],
                    "MacAddress": "06:17:dc:28:53:1f",
                    "NetworkInterfaceId": "eni-0f7acc704b3c99055",
                    "OwnerId": "529088282556",
                    "PrivateDnsName": "ip-172-31-44-183.eu-central-1.compute.internal",
                    "PrivateIpAddress": "172.31.44.183",
                    "PrivateIpAddresses": [
                        {
                            "Primary": true,
                            "PrivateDnsName": "ip-172-31-44-183.eu-central-1.compute.internal",
                            "PrivateIpAddress": "172.31.44.183"
                        }
                    ],
                    "SourceDestCheck": true,
                    "Status": "in-use",
                    "SubnetId": "subnet-0e8f515b5b7d37744",
                    "VpcId": "vpc-010a96ca45520da50",
```

Создаем Image на основе созданного Instance.

```
PS D:\Terraform> aws ec2 create-image --instance-id i-0bc6d5039555a5601 --name "my_ubuntu_avl" --no-reboot
{
    "ImageId": "ami-0d66b2a42da542211"
}
```

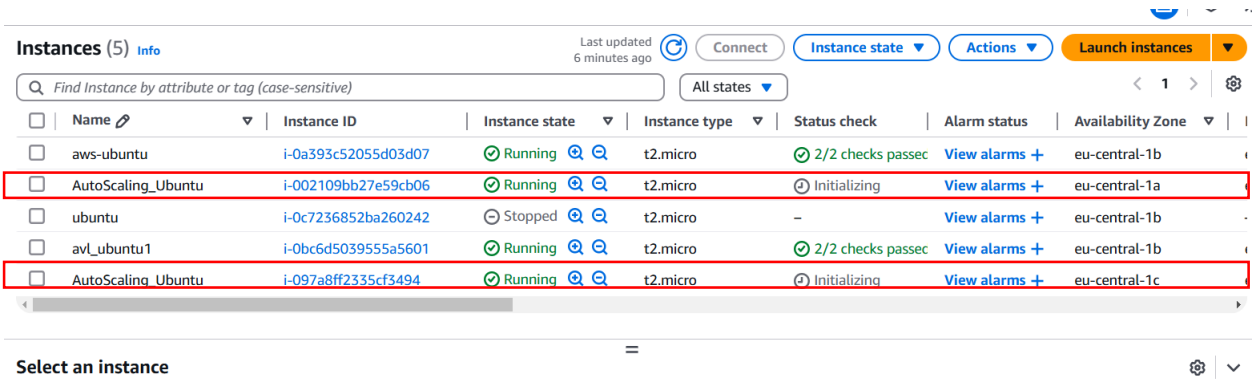


Создаём Auto Scaling Group. Пропустил скрин создания в CLI Launch Template.

```
PS D:\Terraform> aws autoscaling create-auto-scaling-group --auto-scaling-group-name ubunty-asg `
>>     --launch-template "LaunchTemplateName=my-ubuntu-tem,Version=1" `
>>     --min-size 1 --max-size 2 --desired-capacity 2 `
>>     --vpc-zone-identifier "subnet-03207db2b91a2a05e,subnet-00669df1cc78cf133"
```

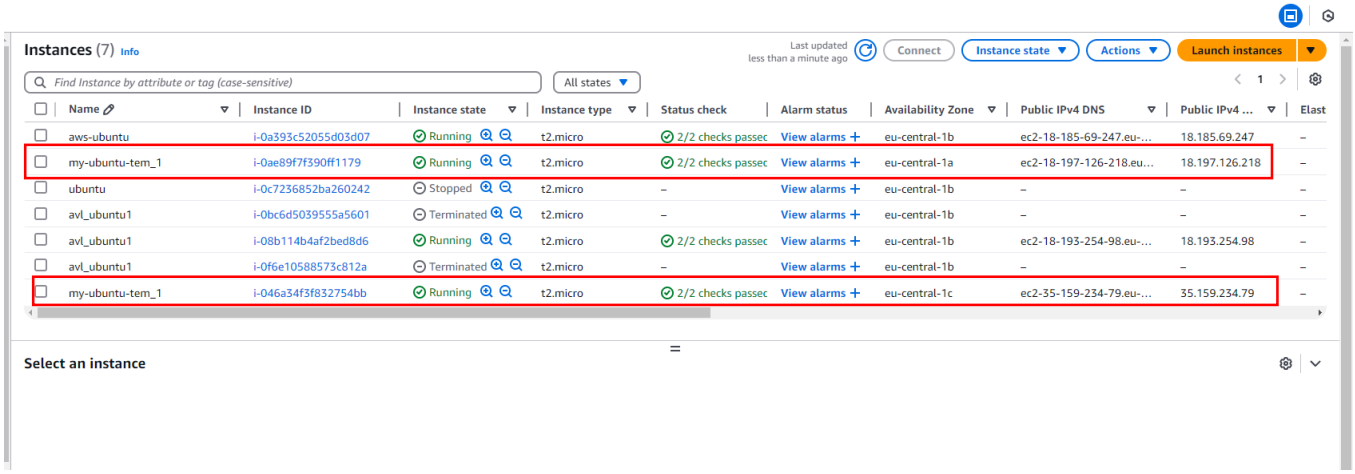
Настраиваем автоматическое масштабирование.

```
PS D:\Terraform> aws autoscaling put-scaling-policy --auto-scaling-group-name ubunty-asg `
>>     --policy-name scale-out-policy --adjustment-type ChangeInCapacity `
>>     --scaling-adjustment 1 --cooldown 300
{
    "PolicyARN": "arn:aws:autoscaling:eu-central-1:529088282556:scalingPolicy:14f3b776-e8e5-4e34-9c5e-3fd9373eb8c2:autoScalingGroupName/
    "Alarms": []
}
```



Приступим к выполнению в терраформе.
Конфиг тераформа прикреплен отдельно.

```
PS D:\Terraform\TEC2> terraform apply
aws_instance.avl_ubuntu1: Refreshing state... [id=i-0f6e10588573c812a]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # aws_ami_from_instance.my_ubuntu_1 will be created
  + resource "aws_ami_from_instance" "my_ubuntu_1" {
      + architecture         = (known after apply)
      + arn                  = (known after apply)
      + boot_mode            = (known after apply)
      + ena_support          = (known after apply)
      + hypervisor           = (known after apply)
      + id                   = (known after apply)
      + image_location       = (known after apply)
      + image_owner_alias    = (known after apply)
      + image_type           = (known after apply)
      + imds_support         = (known after apply)
      + kernel_id            = (known after apply)
      + manage_ebs_snapshots = (known after apply)
      + name                 = "MyUbuntu_1"
      + owner_id             = (known after apply)
      + platform             = (known after apply)
      + platform_details     = (known after apply)
      + public               = (known after apply)
      + ramdisk_id           = (known after apply)
      + root_device_name     = (known after apply)
      + root_snapshot_id     = (known after apply)
      + source_instance_id   = (known after apply)
      + sriov_net_support    = (known after apply)
      + tags_all             = (known after apply)
      + tpm_support          = (known after apply)
      + uefi_data            = (known after apply)
      + usage_operation      = (known after apply)
      + virtualization_type  = (known after apply)

      + ebs_block_device (known after apply)

      + ephemeral_block_device (known after apply)
    }
```


Instances (7) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4 ...
Elast
aws-ubuntu i-0a393c52055d03d07 Running t2.micro 2/2 checks passec View alarms eu-central-1b ec2-18-185-69-247.eu-... 18.185.69.247 –
my-ubuntu-tem_1 i-0ae89f7f390ff1179 Running t2.micro 2/2 checks passec View alarms eu-central-1a ec2-18-197-126-218.eu... 18.197.126.218 –
ubuntu i-0c7236852ba260242 Stopped t2.micro – View alarms eu-central-1b – – –
avl_ubuntu1 i-0bc6d5039555a5601 Terminated t2.micro – View alarms eu-central-1b – – –
avl_ubuntu1 i-08b114b4af2bed8d6 Running t2.micro 2/2 checks passec View alarms eu-central-1b ec2-18-193-254-98.eu-... 18.193.254.98 –
avl_ubuntu1 i-0f6e10588573c812a Terminated t2.micro – View alarms eu-central-1b – – –
my-ubuntu-tem_1 i-046a34f3f832754bb Running t2.micro 2/2 checks passec View alarms eu-central-1c ec2-35-159-234-79.eu-... 35.159.234.79 –
Select an instance