Репозиторий

Для работы с материалами этого курса вам потребуется доступ к репозиторию https://gitlab.slurm.io/terraform/base. Данные для авторизации в Gitlab будут высланы вам на почту, а также продублированы в настройках профиля или над текстом по кнопке "**Доступы**".

Если вы не сможете зарегистрироваться на AWS, вот ссылка на практику переделанную под yandex.cloud - https://gitlab.slurm.io/terraform/base/-/tree/main/practice_yandex_cloud

Пока материал готов 1-ого дня.

Без VPN terraform, к сожалению, работать на текущий момент не будет.

Установка Terraform cli

Ссылка на оф. документацию как установить terraform
- https://learn.hashicorp.com/tutorials/terraform/install-cli

Ссылка на установочные файлы - https://www.terraform.io/downloads

https://releases.hashicorp.com/terraform/

```
unzip terraform_1.1.6_linux_amd64.zip -d /usr/local/bin/
```

По итогу, должна быть реакция после ввода команды `terraform`

```
terraform

Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
```

```
  output          Show output values from your root module
  providers       Show the providers required for this configuration
  refresh         Update the state to match remote systems
  show            Show the current state or a saved plan
  state           Advanced state management
  taint           Mark a resource instance as not fully functional
  untaint         Remove the 'tainted' state from a resource instance
  version         Show the current Terraform version
  workspace       Workspace management

Global options (use these before the subcommand, if any):
  -chdir=DIR      Switch to a different working directory before executing the
                  given subcommand.
  -help           Show this help output, or the help for a specified subcommand.
  -version        An alias for the "version" subcommand.
```

Установка AWS cli

Ссылка на оф. документацию как установить terraform
- https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

По итогу, должна быть реакция после ввода команды `aws help`

```
aws help

aws
^^^
Description
***********
The AWS Command Line Interface is a unified tool to manage your AWS
services.
Synopsis
********

   aws [options] <command> <subcommand> [parameters]

Use *aws command help* for information on a specific command. Use *aws
help topics* to view a list of available help topics. The synopsis for
each command shows its parameters and their usage. Optional parameters
are shown in square brackets.

Options
*******
"--debug" (boolean)
Turn on debug logging.
"--endpoint-url" (string)

......
```

AWS configure

Подключаемся к своему личному аккаунту AWS `aws configure` по данной ссылке https://edu.slurm.io/courses/aws_terraform_connect/units/1545/lessons/5531/steps/18085

При подключение выбираем регион - `eu-central-1`

Приступаем к созданию инстанса

На этой практике поймете взаимодействие AWS и terraform

Создаем папку

```
mkdir terraform
```

Заходим в папку

```
cd terraform
```

Создадим файл

```
touch main.tf
```

Заполняем его

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }

  required_version = ">= 0.14.5"
}

provider "aws" {
  profile = "default"
  region  = "eu-central-1"
}

resource "aws_instance" "app_server" {
  ami           = "ami-05b308c240ae70bb6"
  instance_type = "t2.micro"

}
```

Инициализация рабочего каталога

```
terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.27"...
- Installing hashicorp/aws v3.74.3...
- Installed hashicorp/aws v3.74.3 (self-signed, key ID 34365D9472D7468F)
```

```
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Запускаем создание инстанса в AWS

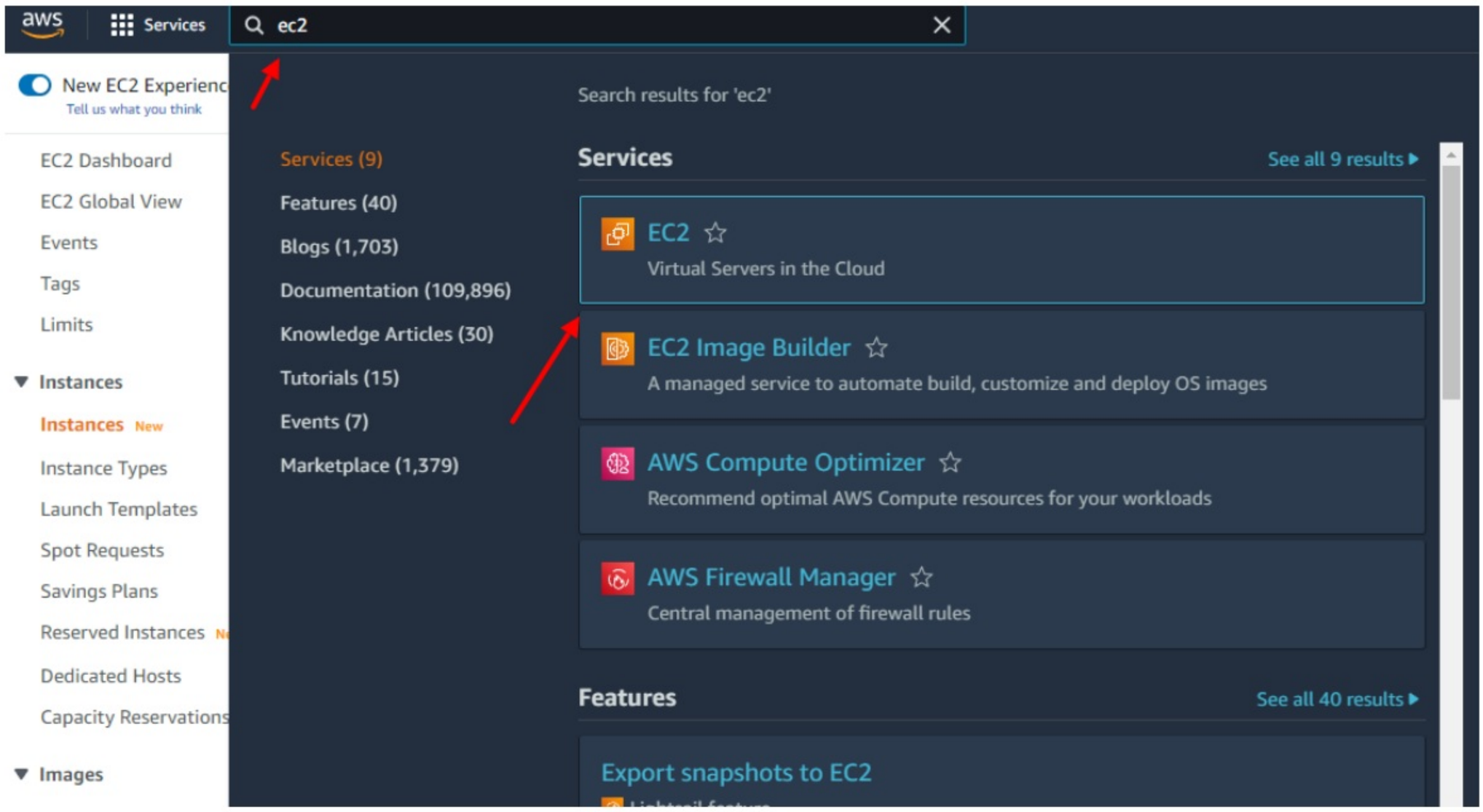
```
terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.app_server will be created
  + resource "aws_instance" "app_server" {
      + ami                                  = "ami-05b308c240ae70bb6"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)

...

  Enter a value: yes
```

Набираем `yes`

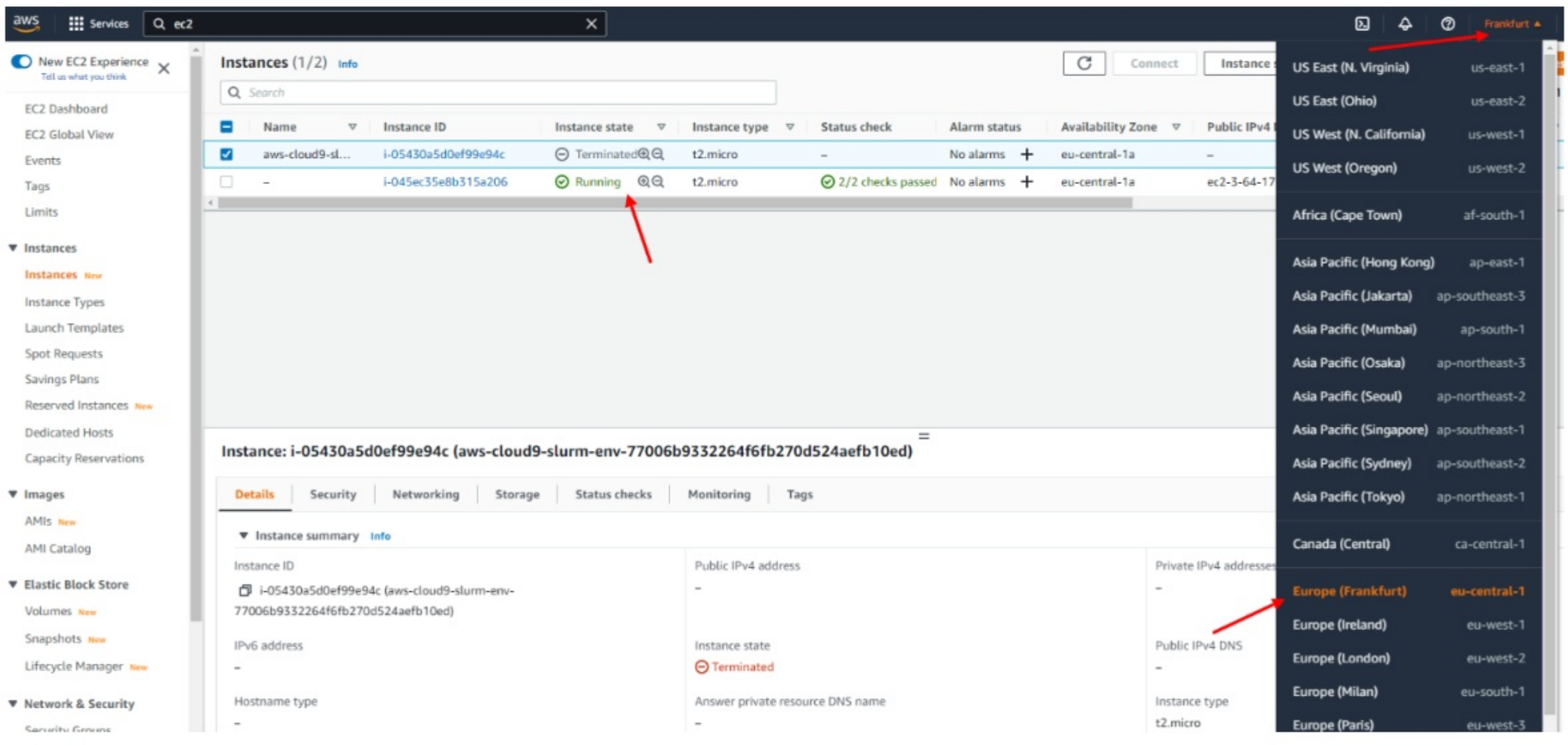
```
aws_instance.app_server: Creating...
aws_instance.app_server: Still creating... [10s elapsed]
aws_instance.app_server: Still creating... [20s elapsed]
aws_instance.app_server: Still creating... [30s elapsed]
aws_instance.app_server: Creation complete after 34s [id=i-045ec35e8b315a206]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Заходим в AWS и смотрим созданный инстанс.

Переходим в EC2



Выбираем регион и смотрим наш созданные инстанс.



Теперь удалим его, т.к. он будет нам стоить денег. Но есть бесплатный лимит использование ресурсов AWS. Более подробно [тут]

Удаляем

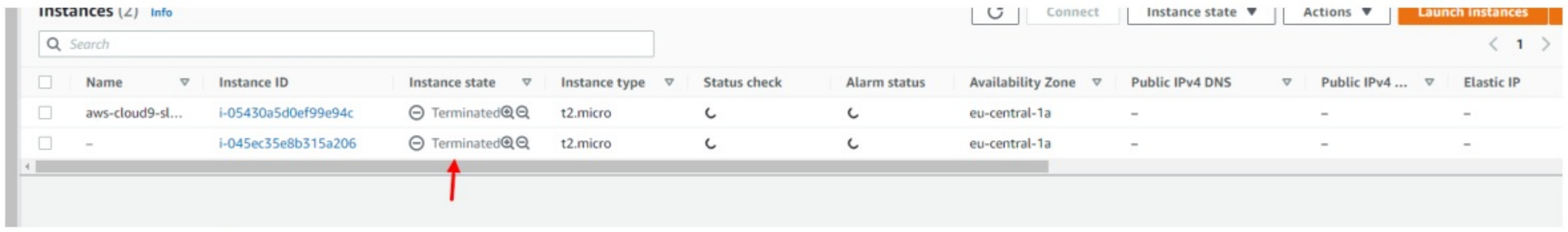
```
terraform destroy

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.app_server will be destroyed
  - resource "aws_instance" "app_server" {
```

```
      - ami                                          = "ami-05b308c240ae70bb6" -> null
      - arn                                          = "arn:aws:ec2:eu-central-
1:472672314710:instance/i-045ec35e8b315a206" -> null
      - associate_public_ip_address                  = true -> null
      - availability_zone                            = "eu-central-1a" -> null
      - cpu_core_count                               = 1 -> null
      - cpu_threads_per_core                         = 1 -> null
      - disable_api_termination                      = false -> null
      - ebs_optimized                                = false -> null
      - get_password_data                            = false -> null
      - hibernation                                  = false -> null
....

  Enter a value: yes
```

Подтверждаем

```
aws_instance.app_server: Destroying... [id=i-045ec35e8b315a206]
aws_instance.app_server: Still destroying... [id=i-045ec35e8b315a206, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-045ec35e8b315a206, 20s elapsed]
aws_instance.app_server: Still destroying... [id=i-045ec35e8b315a206, 30s elapsed]
aws_instance.app_server: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.
```

И проверяем в AWS. Статус **Terminate**



На этом все!