Forget Cloud Front-Ends

Let's Terraform Everything

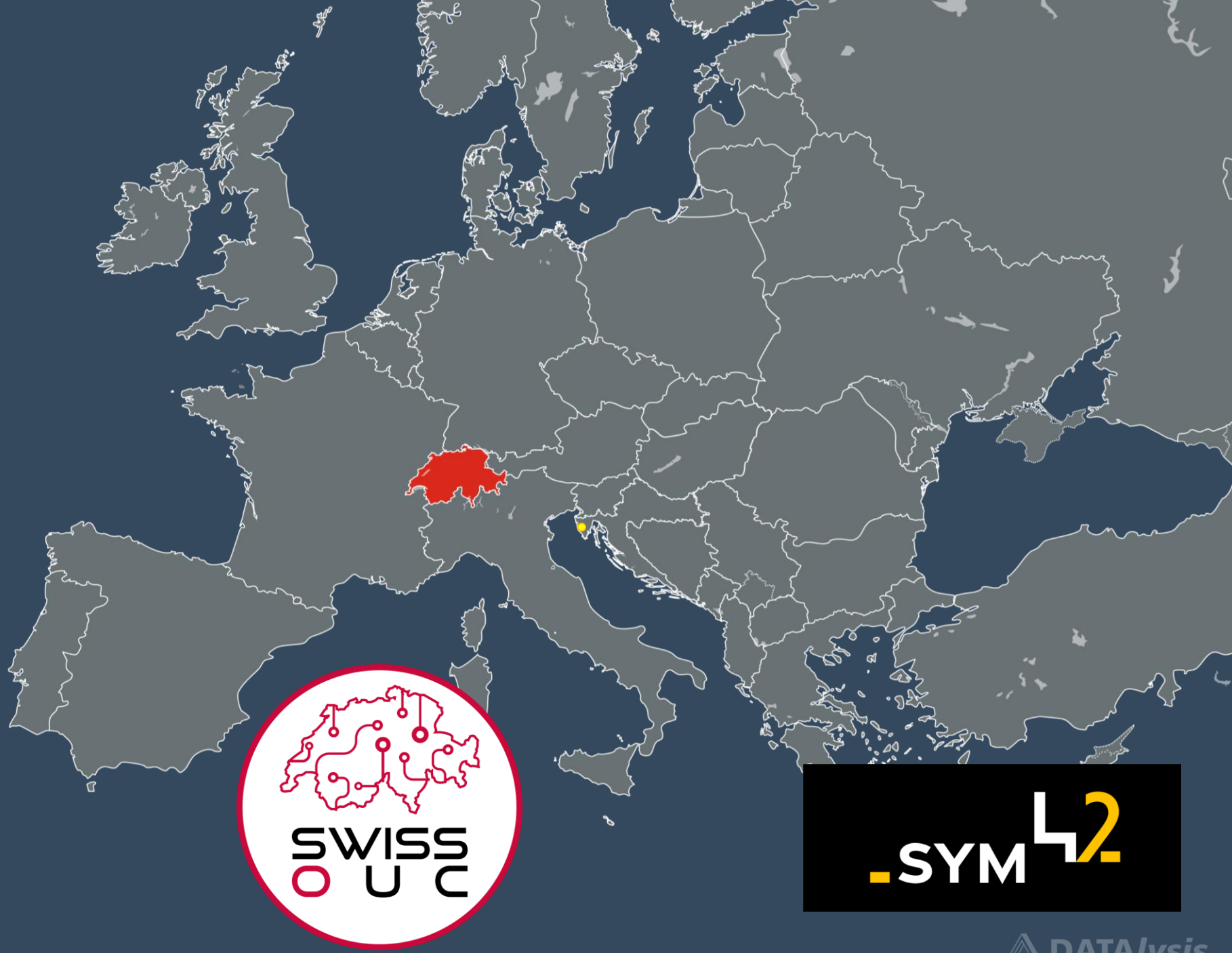
October 18, 2023

hroug '23 annual conference

@G_Ceresa W: www.datalysis.ch E: info@datalysis.ch

Gianni Ceresa

Working with ***data***,
Business Analytics
and EPM tools
for more than
15 years



Oracle ACE Director

SWISS OUC

SYM L2

Oracle ACE

400+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers

Oracle ACE Director

Oracle ACE Pro

Oracle ACE Associate

For more details on Oracle ACE Program:
ace.oracle.com

Oracle ACE

Nominate

yourself or someone you know:

ace.oracle.com/nominate

Connect: **aceprogram_ww@oracle.com** Facebook.com/OracleACEs @oracleace

DISCLAIMER

I'm in no way an expert in Terraform
or claim to be one

- I did try it
- I did use it and keep using it
- So far I didn't crash "the Cloud"
- I also manage to not lose anything I already had in my cloud account (huge achievement...)

Forget Cloud Front-Ends

Let's Terraform Everything

or ... How I did learn Terraform myself

You already did use the Oracle Cloud, right?

To forget cloud front-ends, you first need to have seen it...

Creating a new OCI Compute Instance

What is a compute instance?

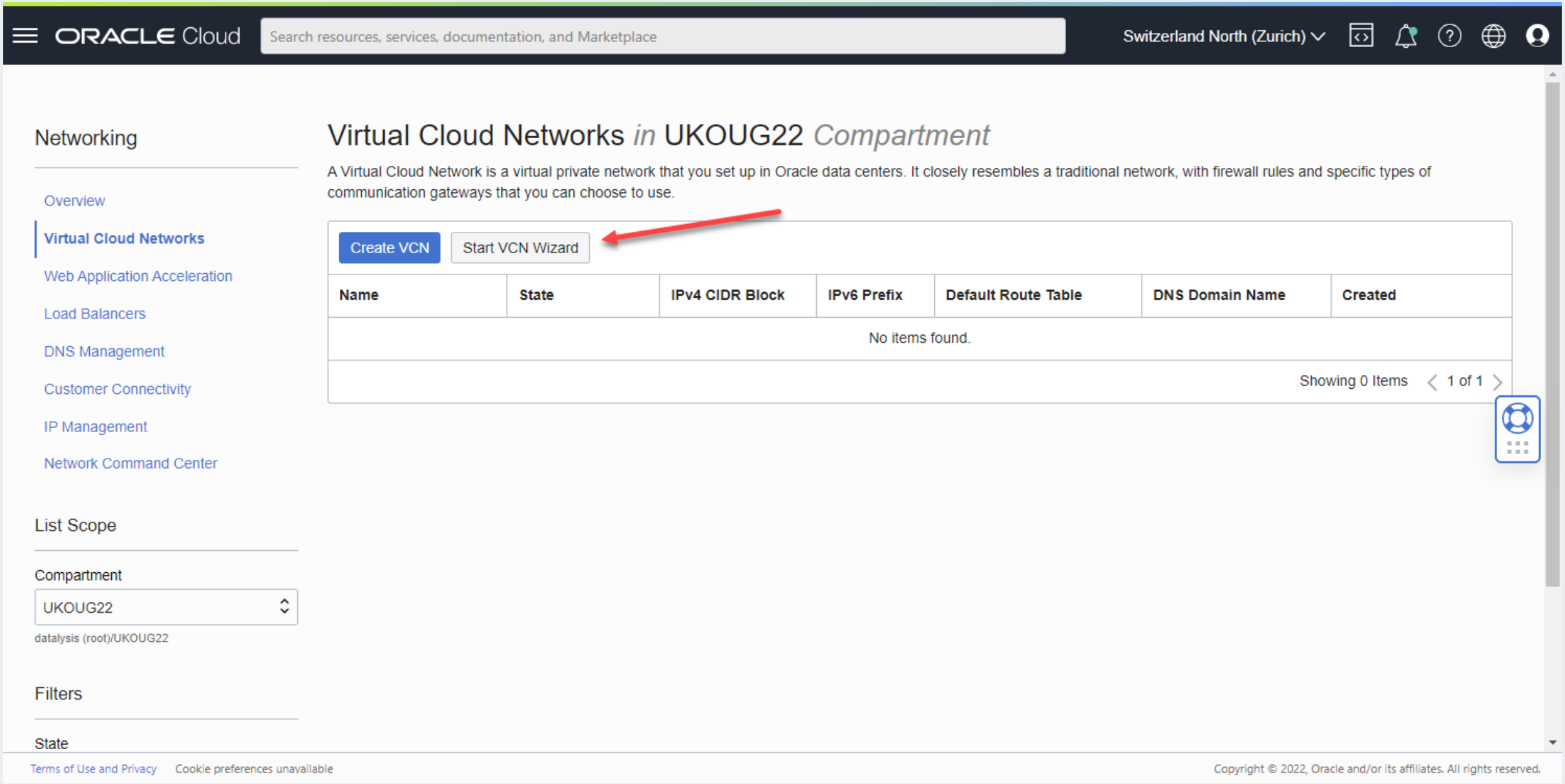
It is like a virtual machine running in the cloud, not much different.

Creating a new OCI Compute Instance

- Requires a Virtual Cloud Network, with at least a subnet
  - Create a new VCN, using the wizard to make it quicker

The screenshots are from 10 months ago, some screens are a bit different now, but all parameters should still be there...

Create a new Virtual Cloud Network (VCN)



Create a new Virtual Cloud Network (VCN)

ORACLE Cloud | Search resources, services, documentation, and Marketplace | Switzerland North (Zurich)

Create a VCN with Internet Connectivity

Help

1. Configuration
2. Review and Create

Configuration

Resource availability checked successfully. Close

Basic Information

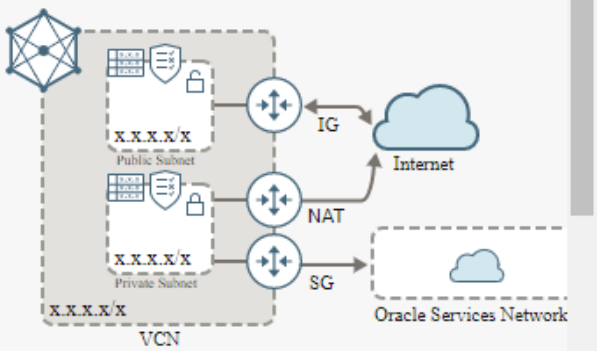
VCN Name
my-new-vcn

Compartment
UKOUG22

datalysis (root)/UKOUG22

Configure VCN and Subnets

VCN with Internet Connectivity

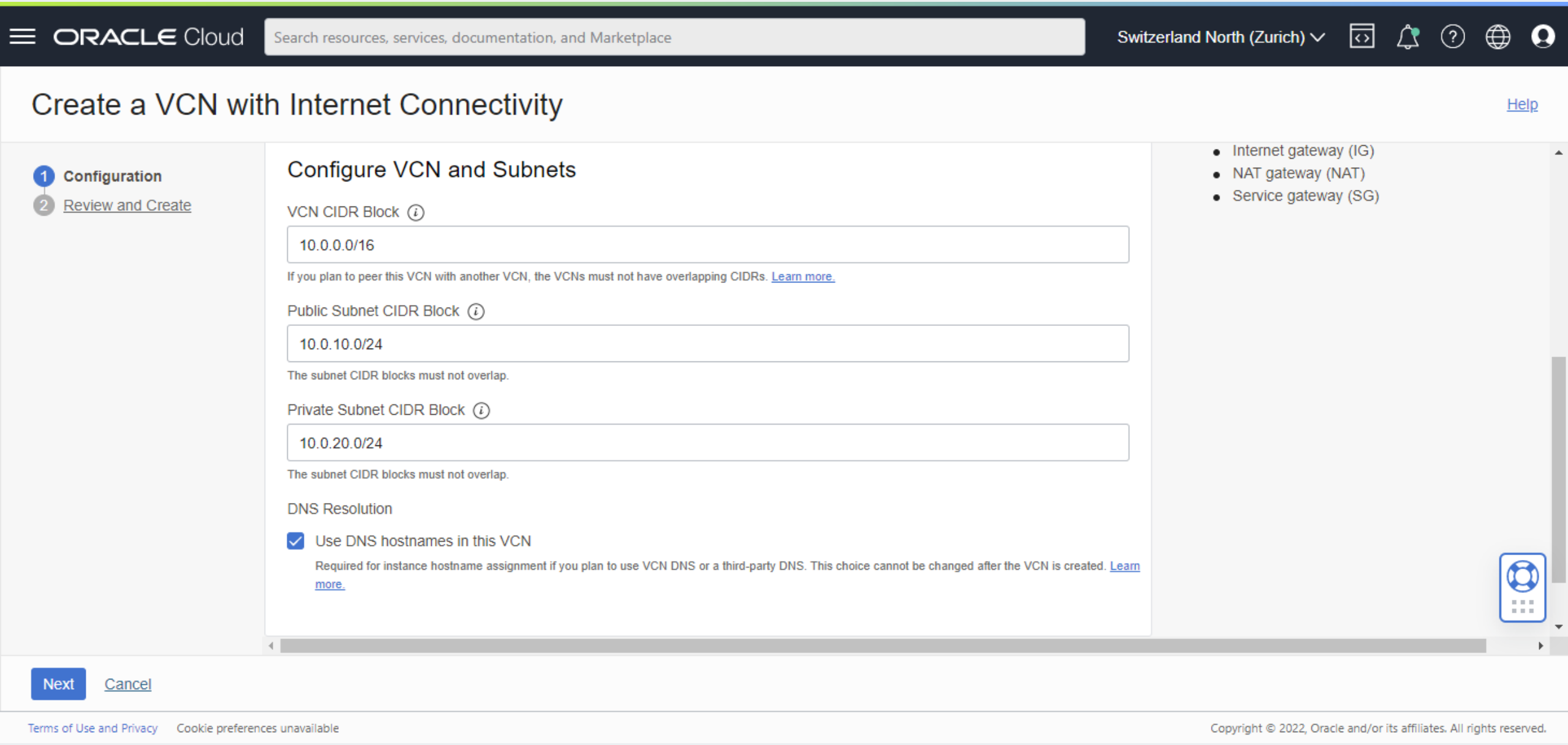


Includes:

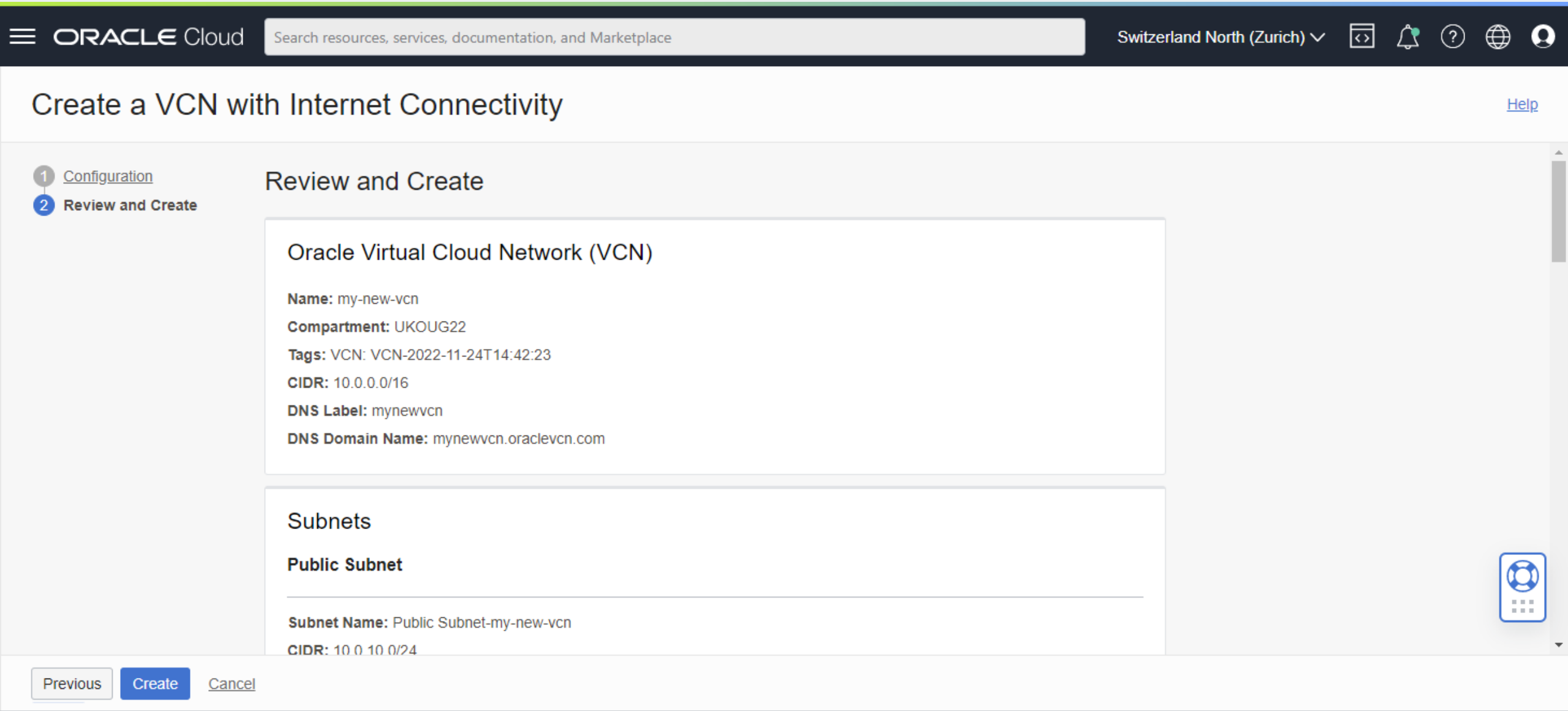
- VCN
- Public subnet
- Private subnet
- Internet gateway (IG)
- NAT gateway (NAT)
- Service gateway (SG)

Next Cancel

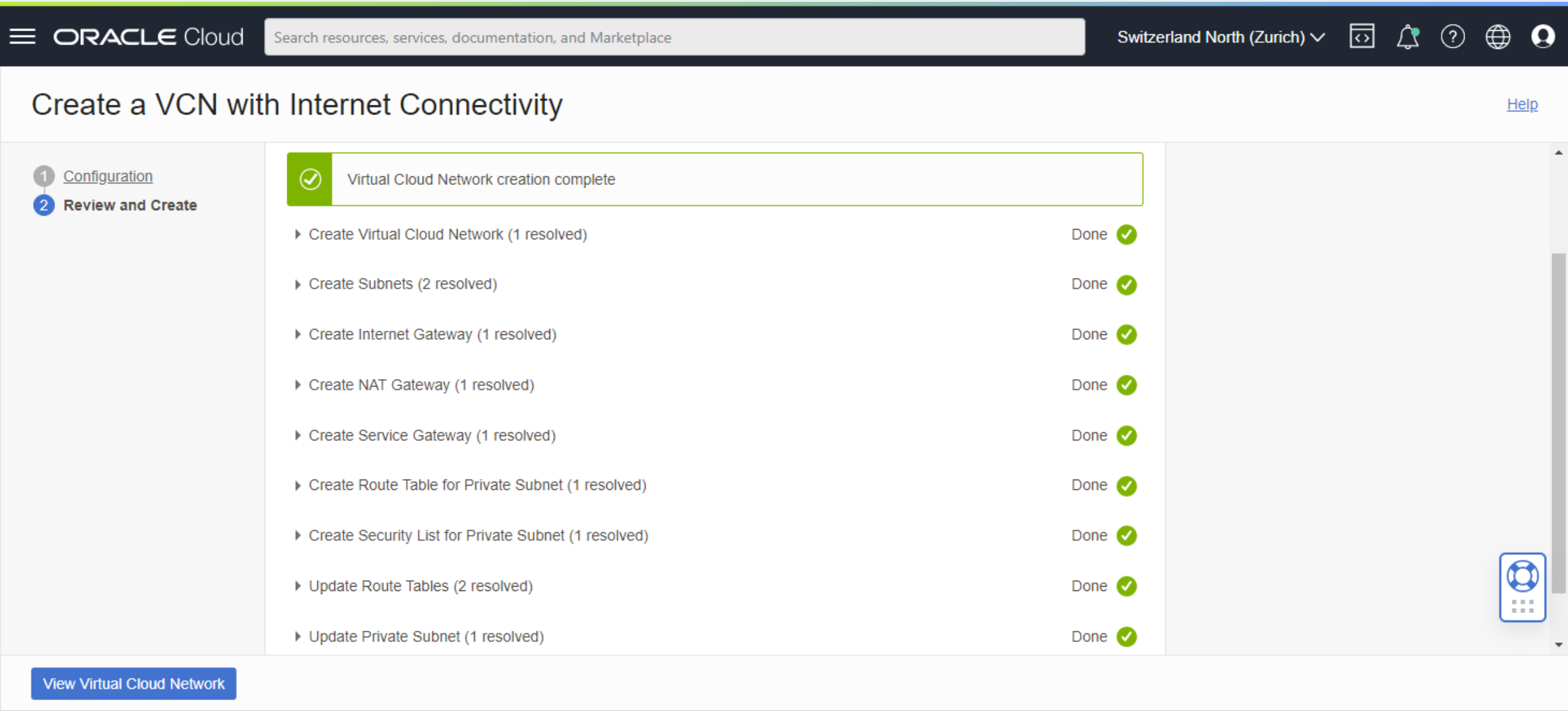
Create a new Virtual Cloud Network (VCN)



Create a new Virtual Cloud Network (VCN)



Create a new Virtual Cloud Network (VCN)

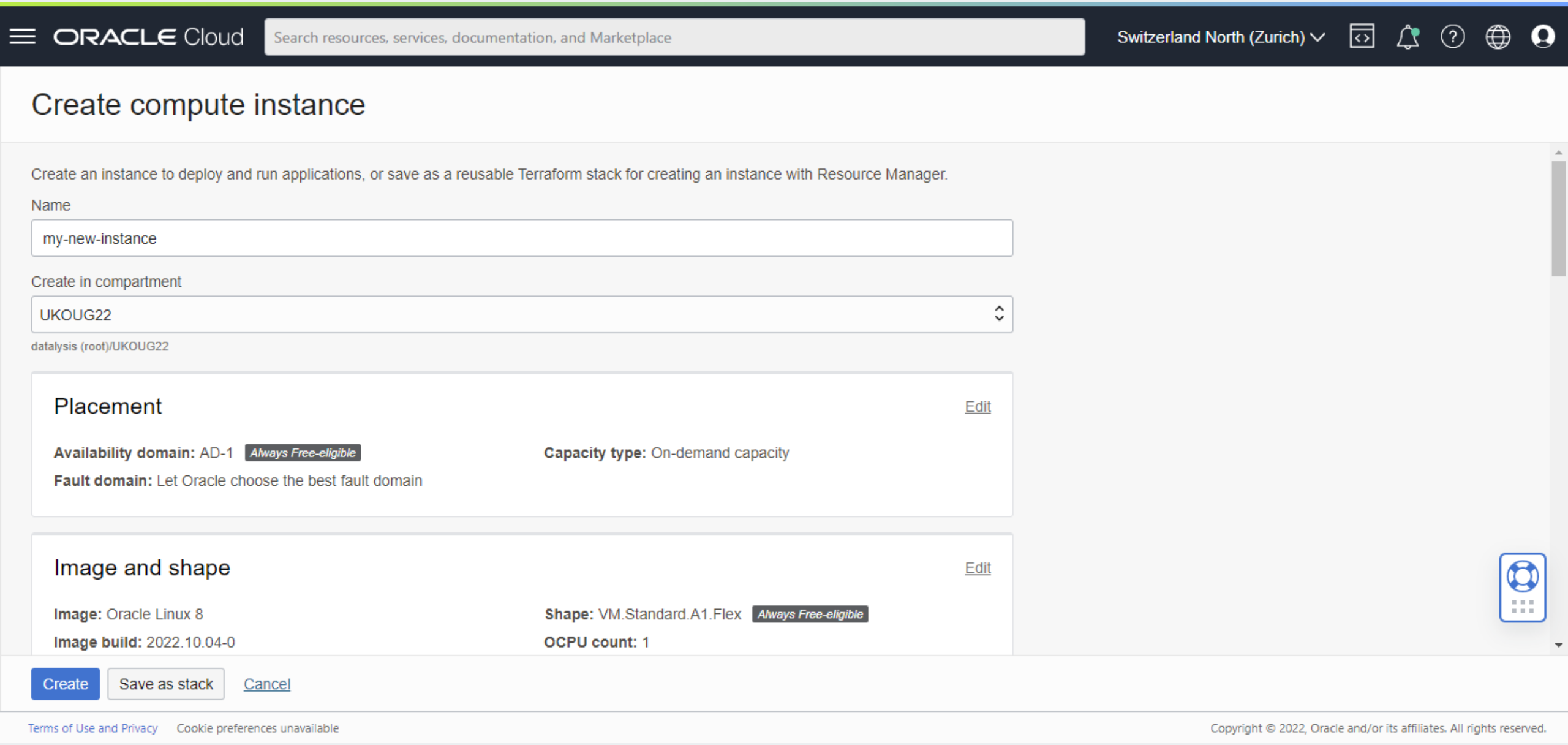


Creating a new OCI Compute Instance

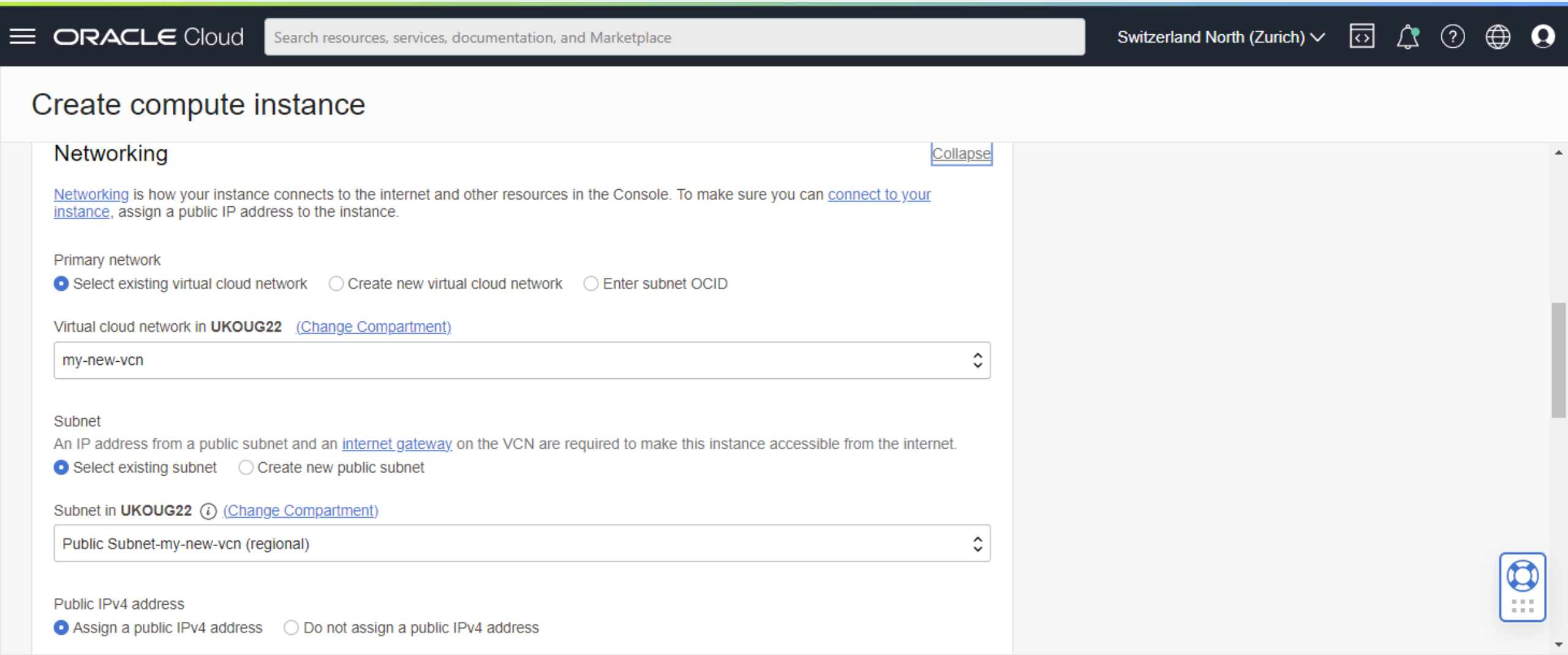
- Requires a Virtual Cloud Network, with at least a subnet
  - Create a new VCN, using the wizard to make it quicker

- Need to select every single option for the compute instance
  - OS (image)
  - Shape (kind of CPU, number of CPUs and RAM)
  - Networking details
  - SSH key
  - ...

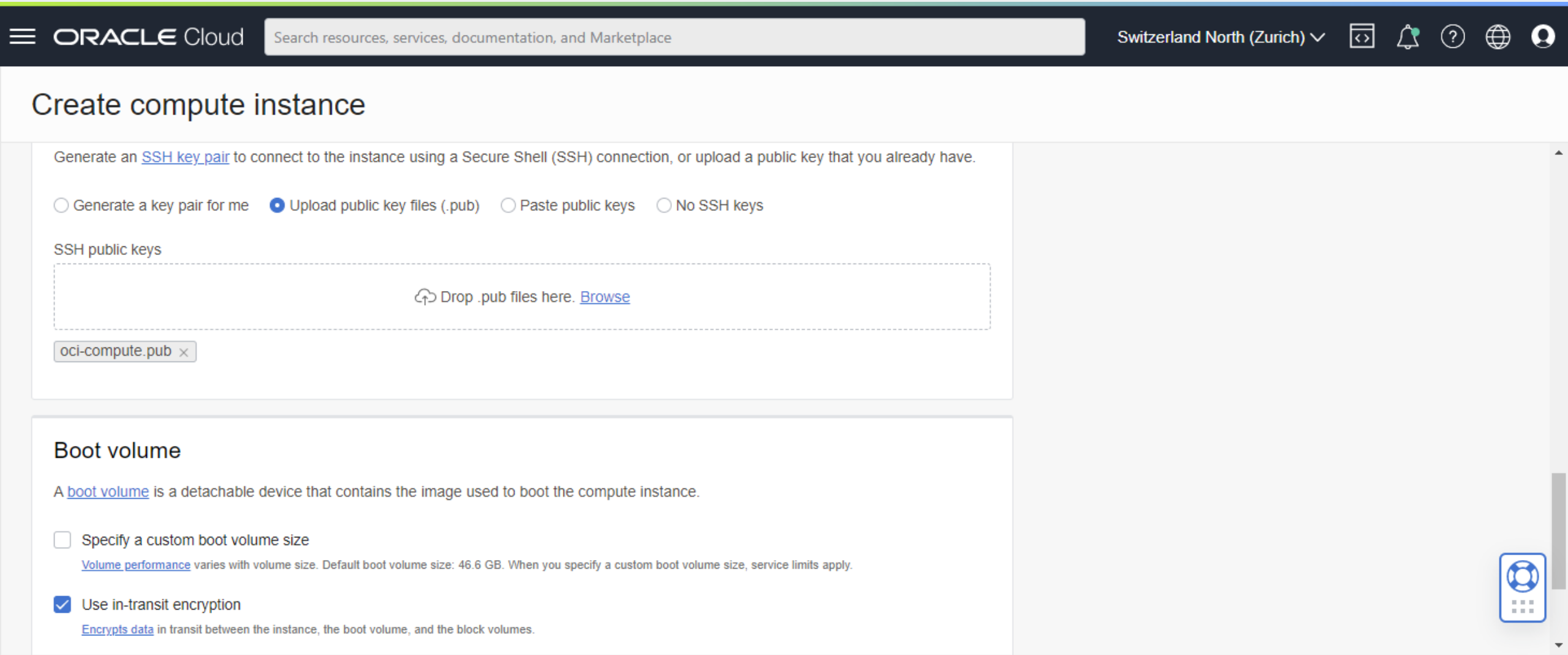
Create a new Compute Instance



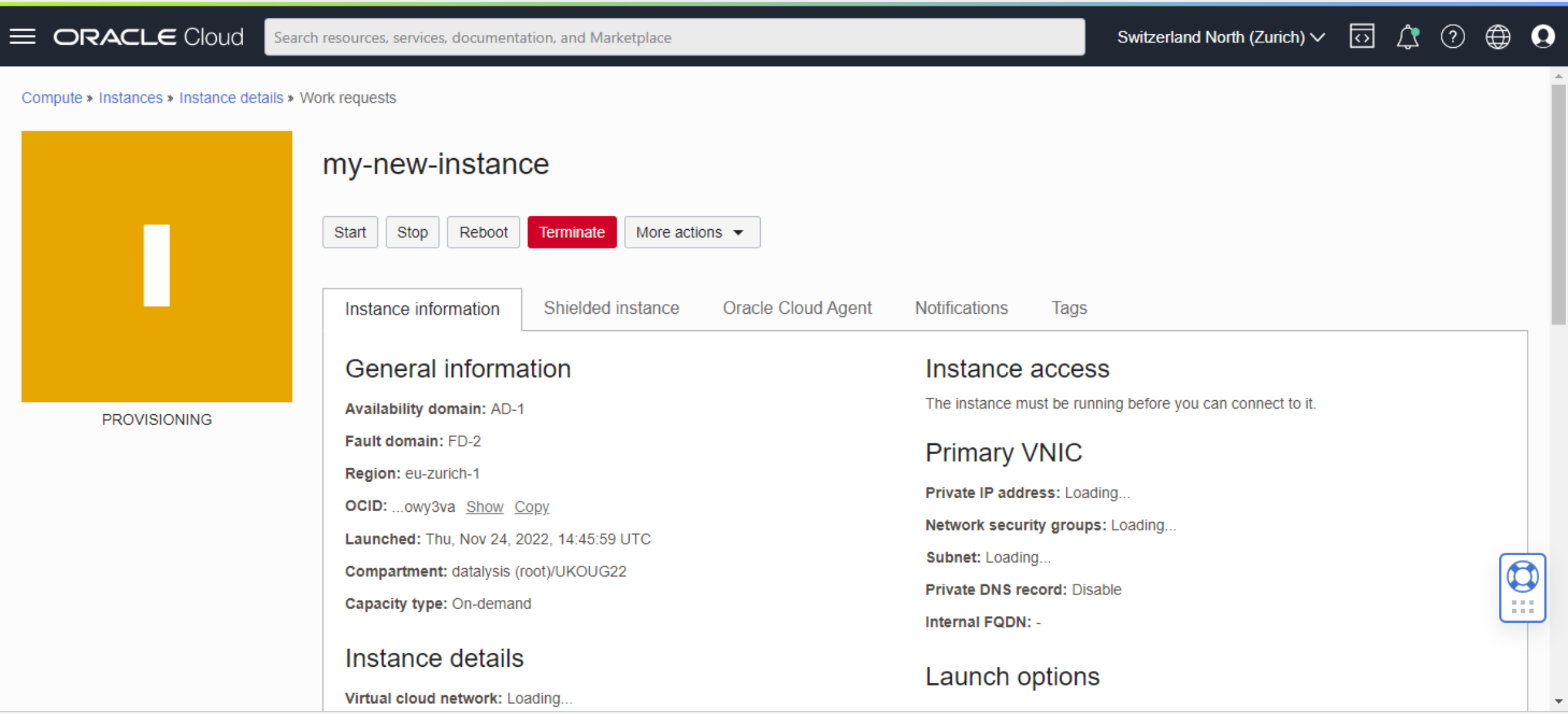
Create a new Compute Instance



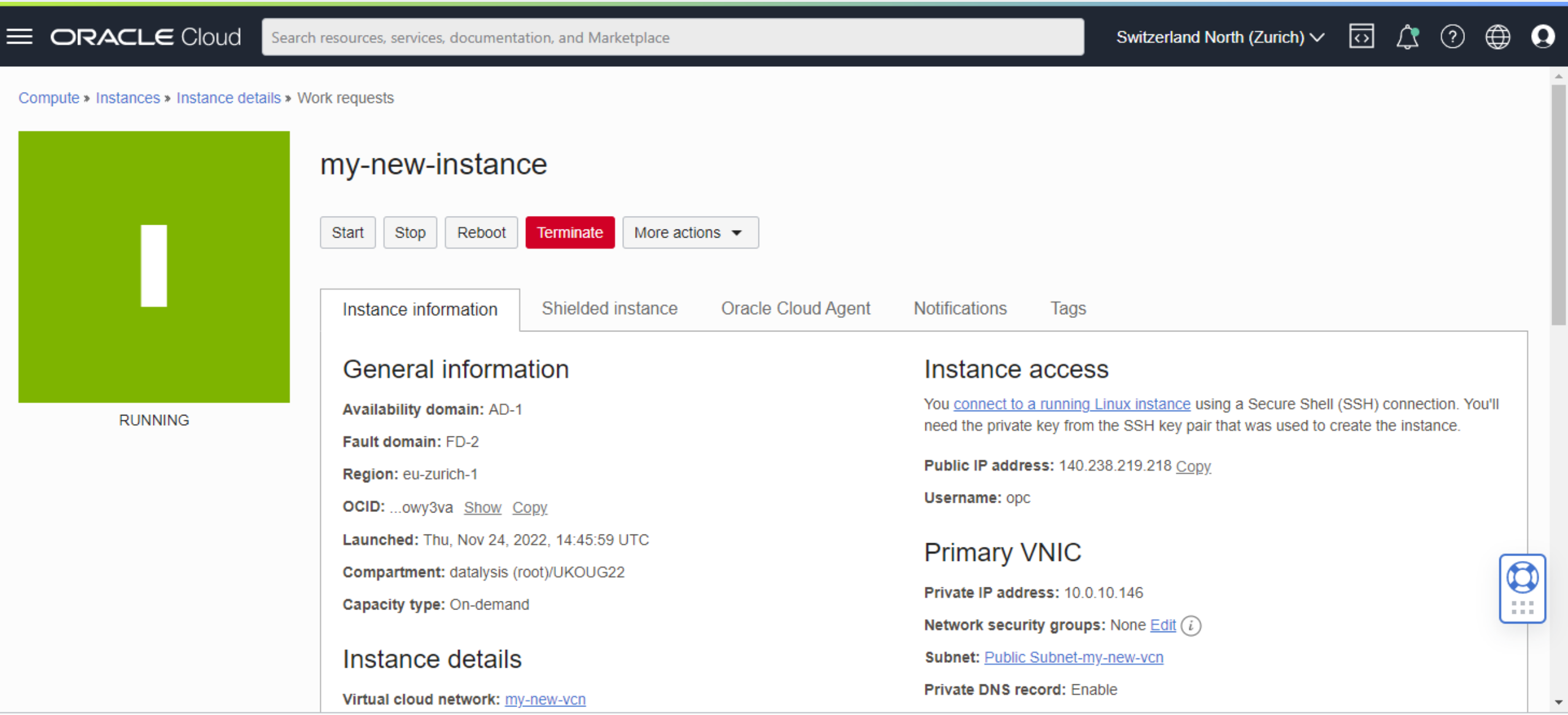
Create a new Compute Instance



Create a new Compute Instance



Create a new Compute Instance



Creating a new OCI Compute Instance

- Requires a Virtual Cloud Network, with at least a subnet
  - Create a new VCN, using the wizard to make it quicker

- Need to select every single option for the compute instance
  - OS (image)
  - Shape (kind of CPU, number of CPUs and RAM)
  - Networking details
  - SSH key
  - ...

When creating 1 compute instance it's fine, when creating 2 compute instances it's still fine ...

Where is the problem?

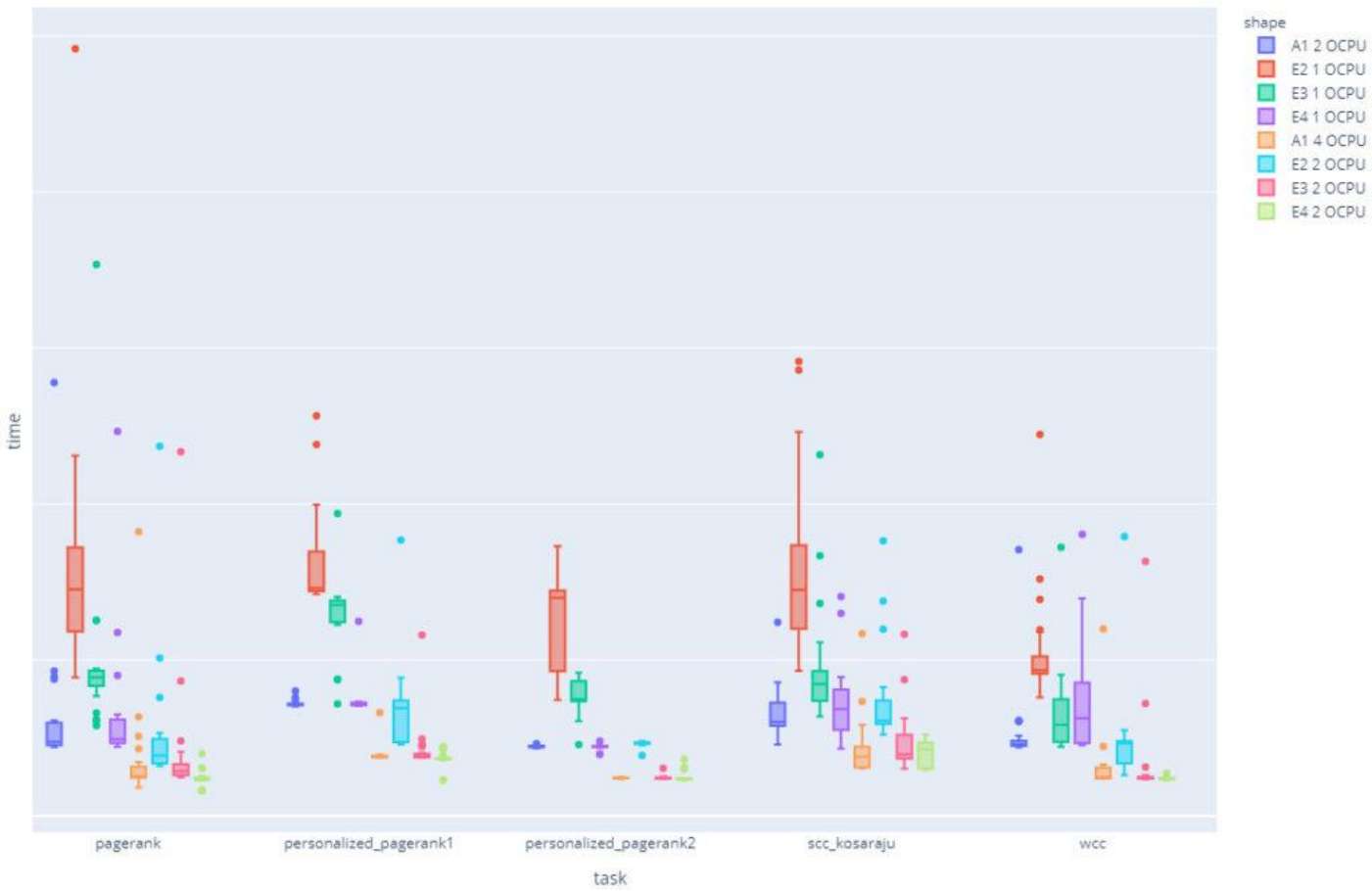
Creating many new OCI Compute Instances

What about when you need to create 8 different shapes of compute instances, install the same environments to execute some tests and repeat all that 5-10 times to compare the results and decide what environment perform the best?

For example:

When Oracle launched the A1 shape with ARM cpus.
How to evaluate how it did perform compared to the existing AMD cpus?

Luckily Terraform exists!



Getting rid of cloud front-ends

Now that you saw a fraction of the cloud front-ends letting you create, manipulate, destroy cloud objects, it's time to look at what could replace them

Terraform: an infrastructure as code tool

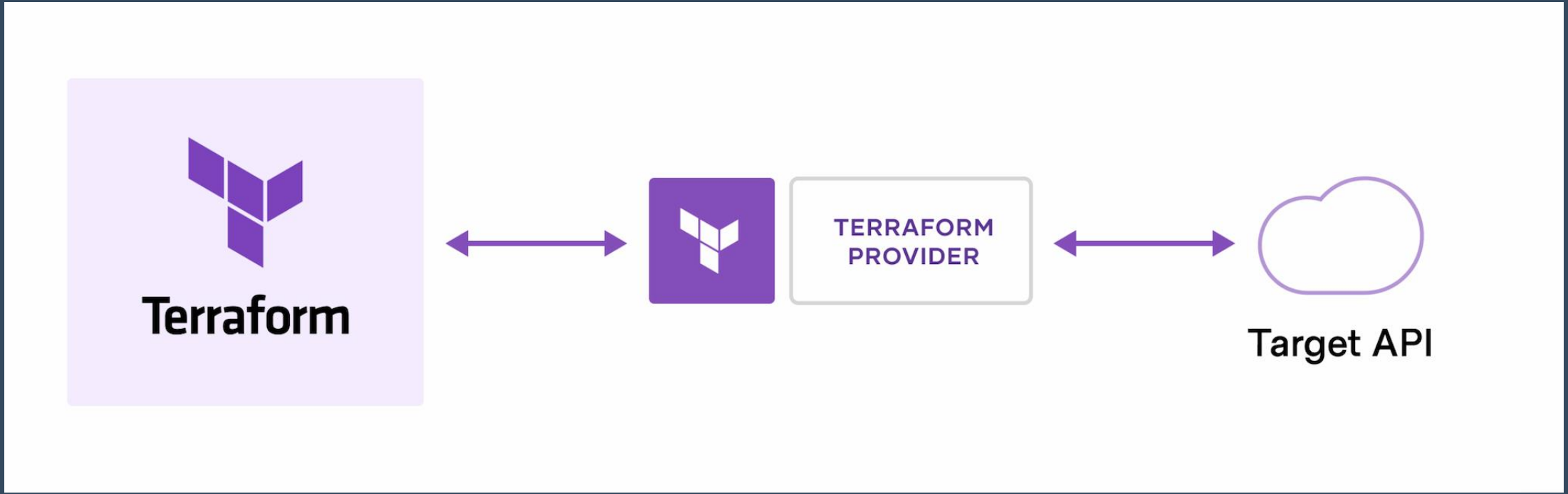
How does Terraform work?

Some important points first:

- Terraform is not an Oracle product, it is from HashiCorp
  - It is cloud-agnostic
  - Oracle Cloud is not the only cloud provider you can use with Terraform

- There is a lot of content available online
  - https://developer.hashicorp.com/terraform/intro
  - https://developer.hashicorp.com/terraform/tutorials/oci-get-started
  - https://docs.oracle.com/en-us/iaas/developer-tutorials/tutorials/home.htm

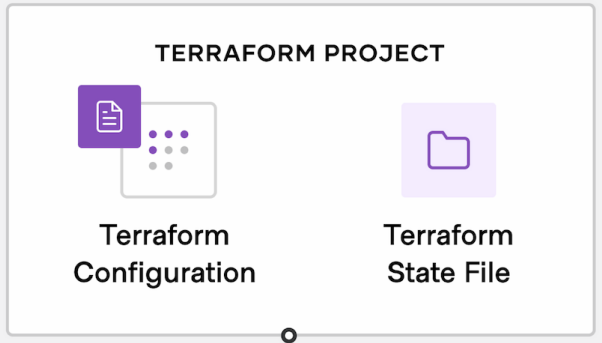
How does Terraform work?



How does Terraform work?

Write

Define infrastructure in configuration files



Plan

Review the changes Terraform will make to your infrastructure

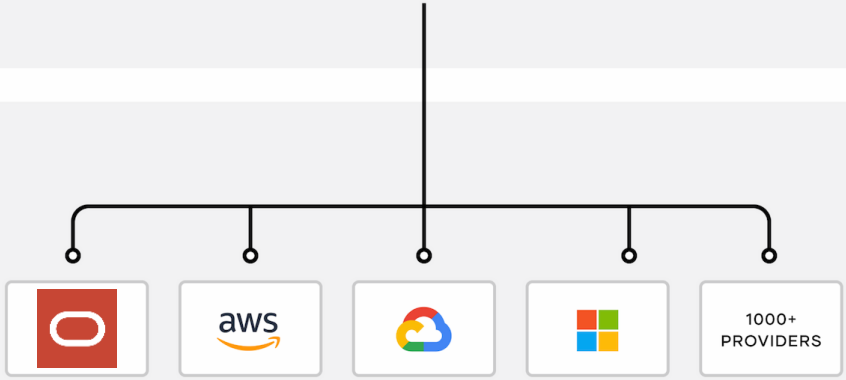
```
$ terraform plan

...

Terraform will perform
the following actions
```

Apply

Terraform provisions your infrastructure and updates the state file.



Terraform important commands

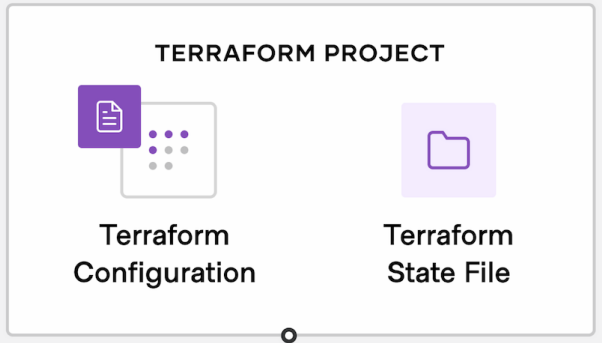
`terraform plan`

- The plan action doesn't perform any change, it only evaluate all the required changes and allows to review what will be created, edited, deleted.
- It is very important because it doesn't represent any risk of breaking things.
- Sometime, when editing an existing object, it will be fully deleted and recreated instead of edited. Because this behaviour can be destructive (think at a compute instance that you did configure and where you are running some application: replacing it would make you lose everything)

How does Terraform work?

Write

Define infrastructure in configuration files



Plan

Review the changes Terraform will make to your infrastructure

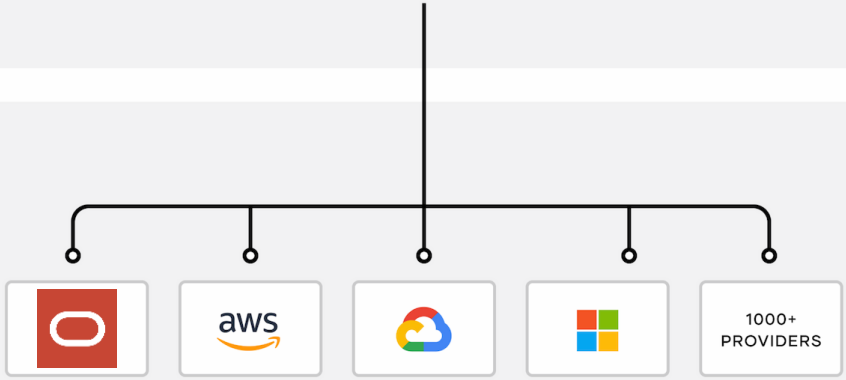
```
$ terraform plan

...

Terraform will perform
the following actions
```

Apply

Terraform provisions your infrastructure and updates the state file.



Terraform important commands

`terraform plan`

- The plan action doesn't perform any change, it only evaluate all the required changes and allows to review what will be created, edited, deleted.
- It is very important because it doesn't represent any risk of breaking things.
- Sometime, when editing an existing object, it will be fully deleted and recreated instead of edited. Because this behaviour can be destructive (think at a compute instance that you did configure and where you are running some application: replacing it would make you lose everything)

`terraform apply`

- Just like plan, by default it does calculate all the required tasks to be performed without applying them directly.
- It does prompt a YES / NO question to confirm if you want to apply the changes for real.

Terraform important commands

`terraform plan`

- The plan action doesn't perform any change, it only evaluate all the required changes and allows to review what will be created, edited, deleted.
- It is very important because it doesn't represent any risk of breaking things.
- Sometime, when editing an existing object, it will be fully deleted and recreated instead of edited. Because this behaviour can be destructive (think at a compute instance that you did configure and where you are running some application: replacing it would make you lose everything)

`terraform apply`

- Just like plan, by default it does calculate all the required tasks to be performed without applying them directly.
- It does prompt a YES / NO question to confirm if you want to apply the changes for real.

Both commands accept a `-destroy` attribute

- The meaning is quite obvious: instead of making the configuration it will destroy it all.

Terraform State

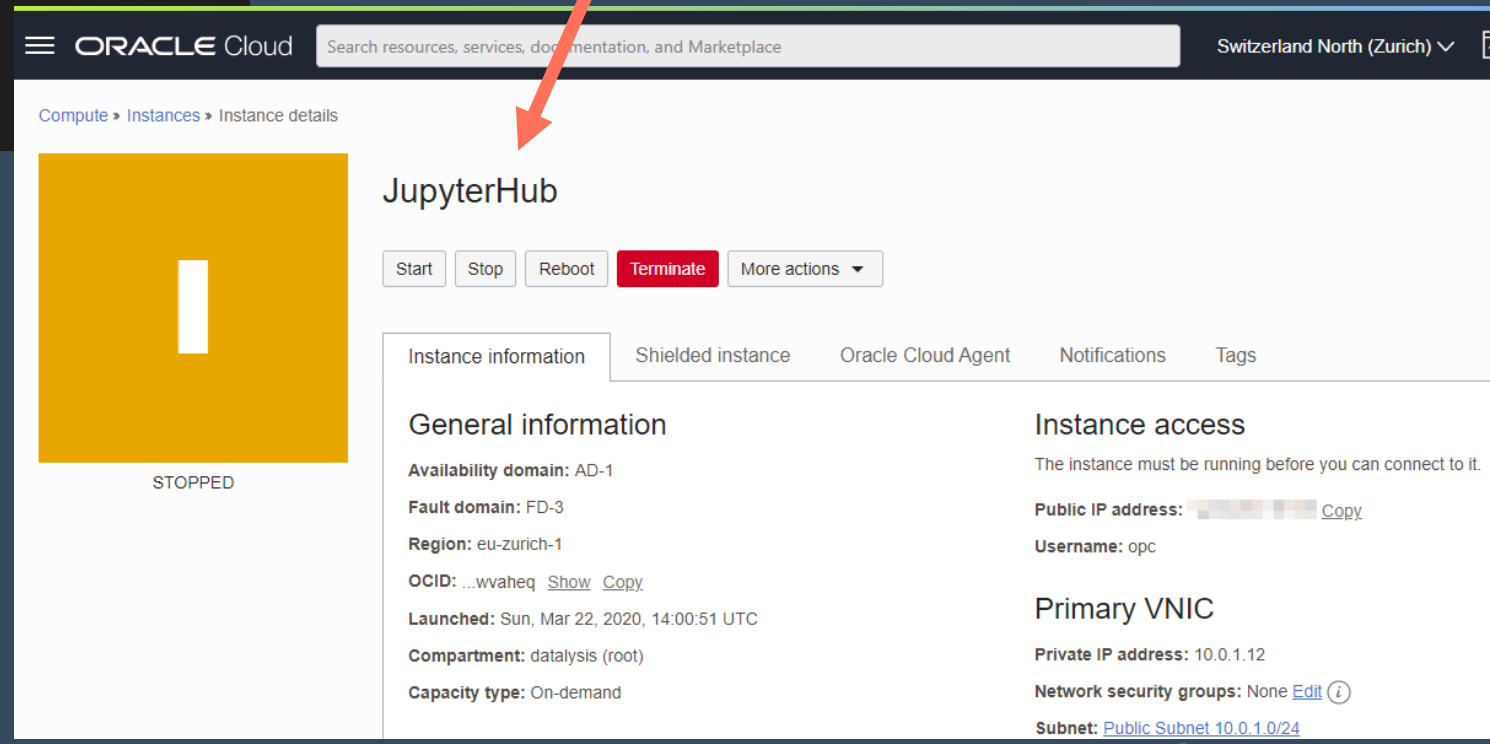
- All your Terraform code is disconnected from reality
  - It does represent elements and dependencies between them
  - It isn't related to a specific, existing, object
- The Terraform State (file) is the connection between your code and reality
  - Terraform identifiers are mapped to real objects (using OCID in Oracle Cloud)
  - Extra metadata is also stored (status etc.)

Terraform State

```
# JupyterHub [VM.Standard.E2.2]
resource "oci_core_instance" "compute_instance-jupyterhub" {
  # Required
  compartment_id      = var.compartment_ocid
  availability_domain = data.oci_identity_availability_domain.EU-ZURICH-1-AD-1.name
  shape               = "VM.Standard.E2.2"

  # Optional
  display_name = "JupyterHub"
  metadata = {
    ssh_authorized_keys = chomp(file(var.ssh_public_keys.compute))
  }

  lifecycle {
    prevent_destroy = true
  }
}
```



Terraform State

terraform.tfstate

terraform.tfstate > [ ] resources > {} 12 > [ ] instances > {} 0 > {} attributes > [ ] instance_options > {} 0 > are_legacy_imds_endpoints_disabled


```
{
  "mode": "managed",
  "type": "oci_core_instance",
  "name": "compute_instance-jupyterhub",
  "provider": "provider[\"registry.terraform.io/hashicorp/oci\"]",
  "instances": [
    {
      "schema_version": 0,
      "attributes": {
        "agent_config": [
          {
            "are_all_plugins_disabled": false,
            "is_management_disabled": false,
            "is_monitoring_disabled": false,
            "plugins_config": []
          }
        ],
        "availability_config": [
          {
            "is_live_migration_preferred": false,
            "recovery_action": "RESTORE_INSTANCE"
          }
        ],
        "availability_domain": "bBBw:EU-ZURICH-1-AD-1",
        "boot_volume_id": "ocid1.bootvolume.oc1.eu-zurich-1.ab5heljrovfxjk22i6n5ehcwerkzfcqbb3bspa7t2ob66bkp4lxtx2hgoxra",
        "capacity_reservation_id": null,
        "compartment_id": "ocid1.tenancy.oc1..aaaaaaaaxkrfbzdi72pukgwprkjer2lro6l7k5kmcips7syvku54mkkv4boq",
        "create_vnic_details": [
          {
            "assign_private_dns_record": false,
            "assign_public_ip": "true",
            "defined_tags": {},
            "display_name": "JupyterHub",
            "freeform_tags": {},
            "hostname_label": "jupyterhub",
            "nsg_ids": [],
            "private_ip": "10.0.1.12",
            "skip_source_dest_check": false,
            "subnet_id": "ocid1.subnet.oc1.eu-zurich-1.aaaaaaaagv2qod6md7ff2z4cq4cyjhxpjql3bqfwpwrqabp7zoyfntbj3yfq",
            "vlan_id": ""
          }
        ],
        "dedicated_vm_host_id": null,
        "defined_tags": {},
        "display_name": "JupyterHub",
        "extended_metadata": {},
        "fault_domain": "FAULT-DOMAIN-3",
        "freeform_tags": {},
        "hostname_label": "jupyterhub",
        "id": "ocid1.instance.oc1.eu-zurich-1.an5heljr73hf3sacwndaci7ftqop3hpqswvbw3hdi23efzcnrkudx7wvaheq",
        "image": "ocid1.image.oc1.eu-zurich-1.aaaaaaaa4nwf5h6nl3u5cdauemg352itja6izecs7ol73z6jftsg4agpdsma",
```

Terraform State

- All your Terraform code is disconnected from reality
  - It does represent elements and dependencies between them
  - It isn't related to a specific, existing, object
- The Terraform State (file) is the connection between your code and reality
  - Terraform identifiers are mapped to real objects (using OCID in Oracle Cloud)
  - Extra metadata is also stored (status etc.)
- Terraform without a State doesn't know what already exists or not in your cloud tenancy
  - It will try to recreate everything
- It is possible to map Terraform identifiers to existing objects OCID by code
  - Making Terraform State aware of already existing objects

```
terraform import oci_core_instance.compute_instance-jupyterhub
ocid1.instance.oc1.eu-zurich-1...wvaheq
```

Where does Terraform fit in the already long list of tools?

There are many tools out there doing similar/same/other things

- Ansible
- python OCI CLI
- Chef, Puppet, ...

If you google this topic you find lot of “opinions”

There isn’t an answer, that I would call universal, to questions like:

- *X vs Y*
- *Should I use X or Y?*
- *Is X better than Y?*

Where does Terraform fit in the already long list of tools?

My personal choice (and opinion) is to pick a tool by type of activity I want to do

Infrastructure

- Terraform
- Manually via the web interface

Installation, configuration, customization of the environment (inside the OS)

- Ansible
- cloud-init
- Custom scripts

Both can be chained

- Terraform executing Ansible on the newly created instances

All this is very nice, but ...

How to learn Terraform?

How to learn and get started with Terraform?

There are tutorials, “step by step” instructions, on both HashiCorp and Oracle website

The documentation is detailed and cover all the options available

You can start from scratch and just try to build your cloud infrastructure from nothing

Or ...

How to learn and get started with Terraform?

A reversed approach:

- Creating a cloud object by hand, in the web interface
  - Allows to see the required values
  - What options are offered
- Export an existing cloud object to a Terraform definition file
- Reading the Terraform code, comparing with what has been entered in the web interface and what says the documentation
- Create a new object in Terraform and explore the actions Terraform wants to perform

Learning based on existing objects, created by you, to have code you know and can fully understand instead of a random "hello world" example

Seeing it in action

Terraform: installation

(I do use an Oracle Linux 9 WSL image to run Terraform)

```
# install the developers repository
sudo dnf install -y oraclelinux-developer-release-el9

# check if Terraform is available
sudo dnf list 'terraform*'

# install Terraform and the Terraform OCI provider
sudo dnf install -y terraform terraform-provider-oci

# the OCI provider is installed in /usr/bin/terraform-provider-
oci_v<version_number>
```

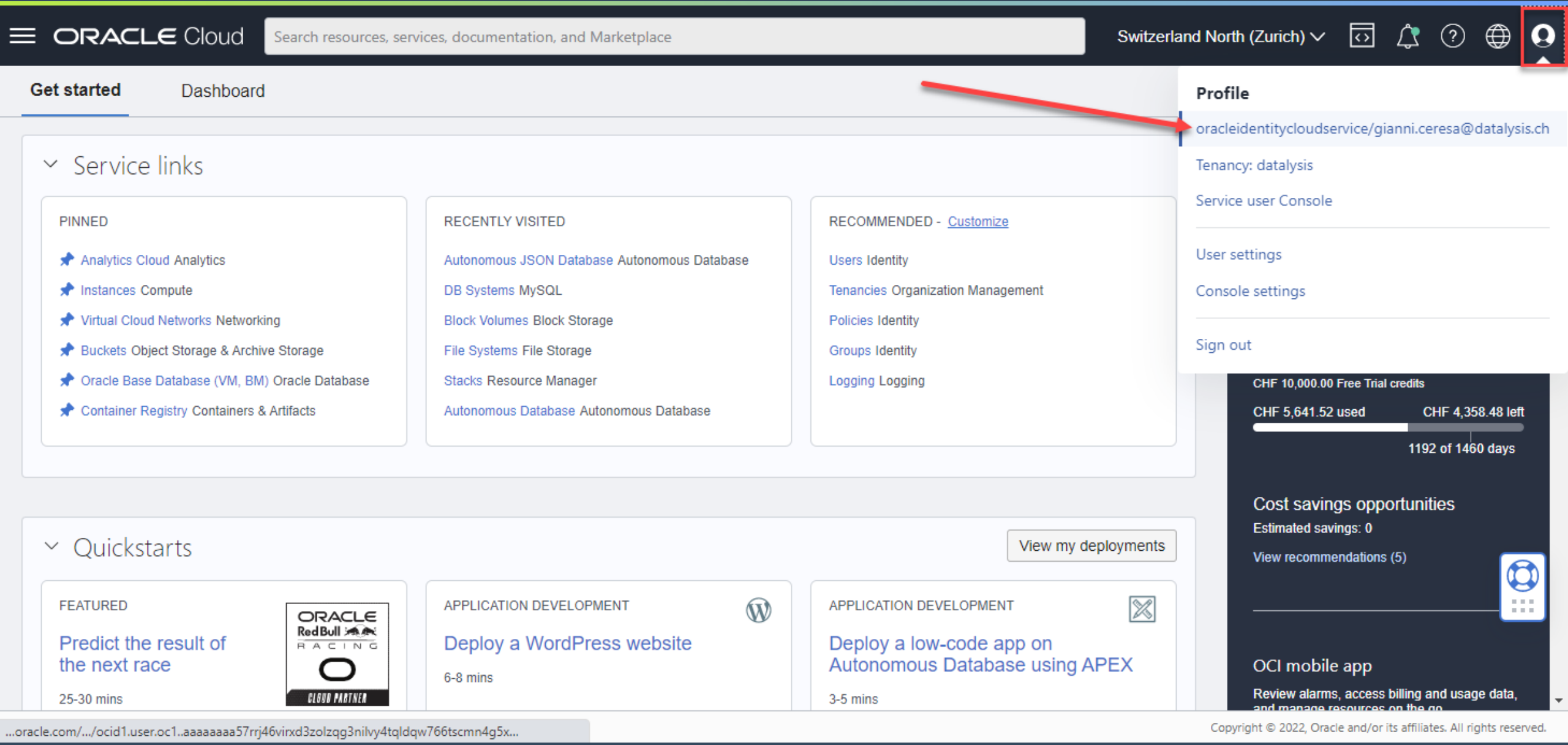
Terraform: authentication

Terraform needs to connect to the cloud provider and because of that an authentication mechanism should be used

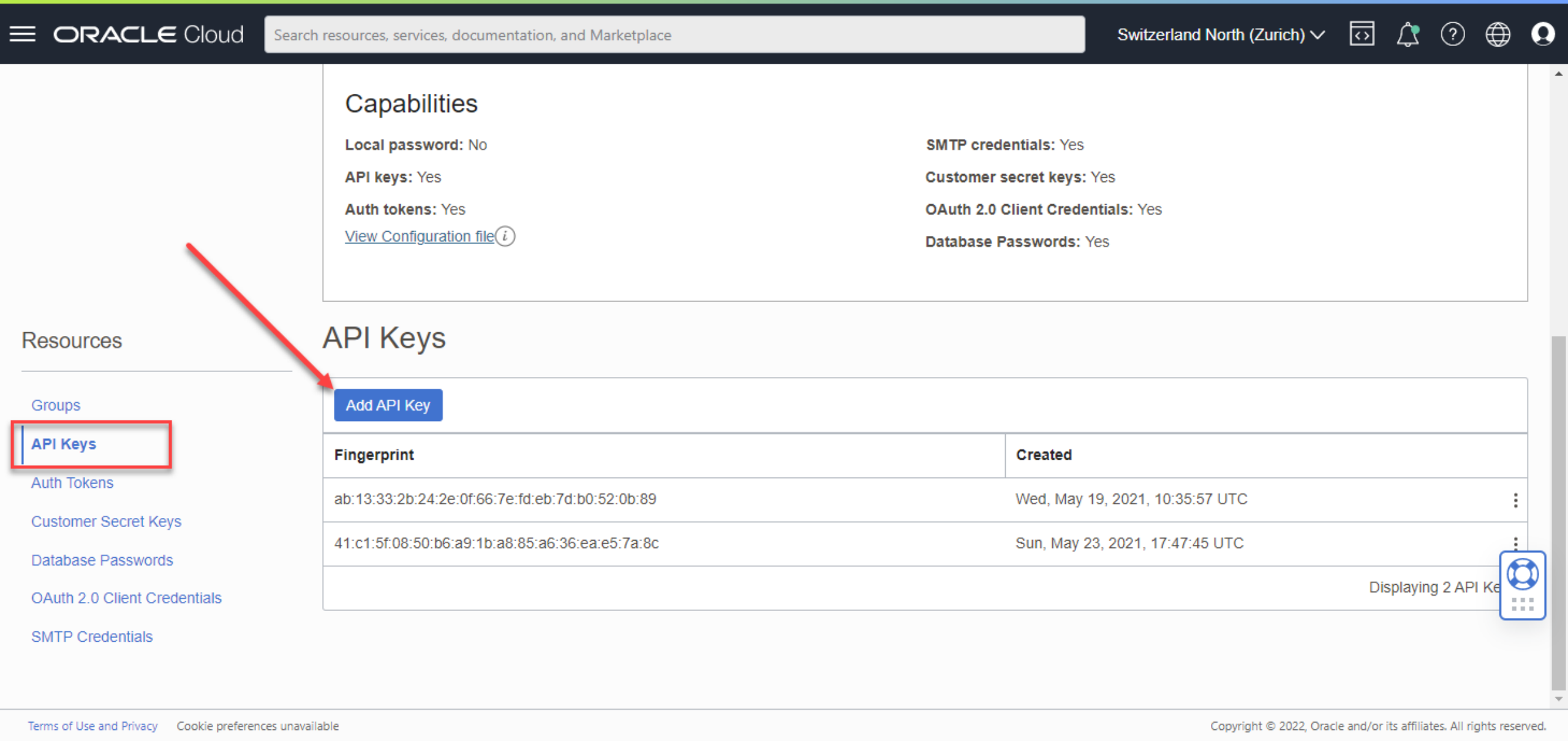
My preference is for an API key:

- Login to your cloud tenancy
- Go to Profile > My Profile > API Keys
- Download a new private key OR upload a public key
- Copy the configuration file preview, edit by adding the correct path to the private key
- Save as ~/.oci/config (the private key can also be saved in ~/.oci with chmod 600 to protect it)
  - This is the default location where Terraform will look for authentication details if not explicitly provided in a different way

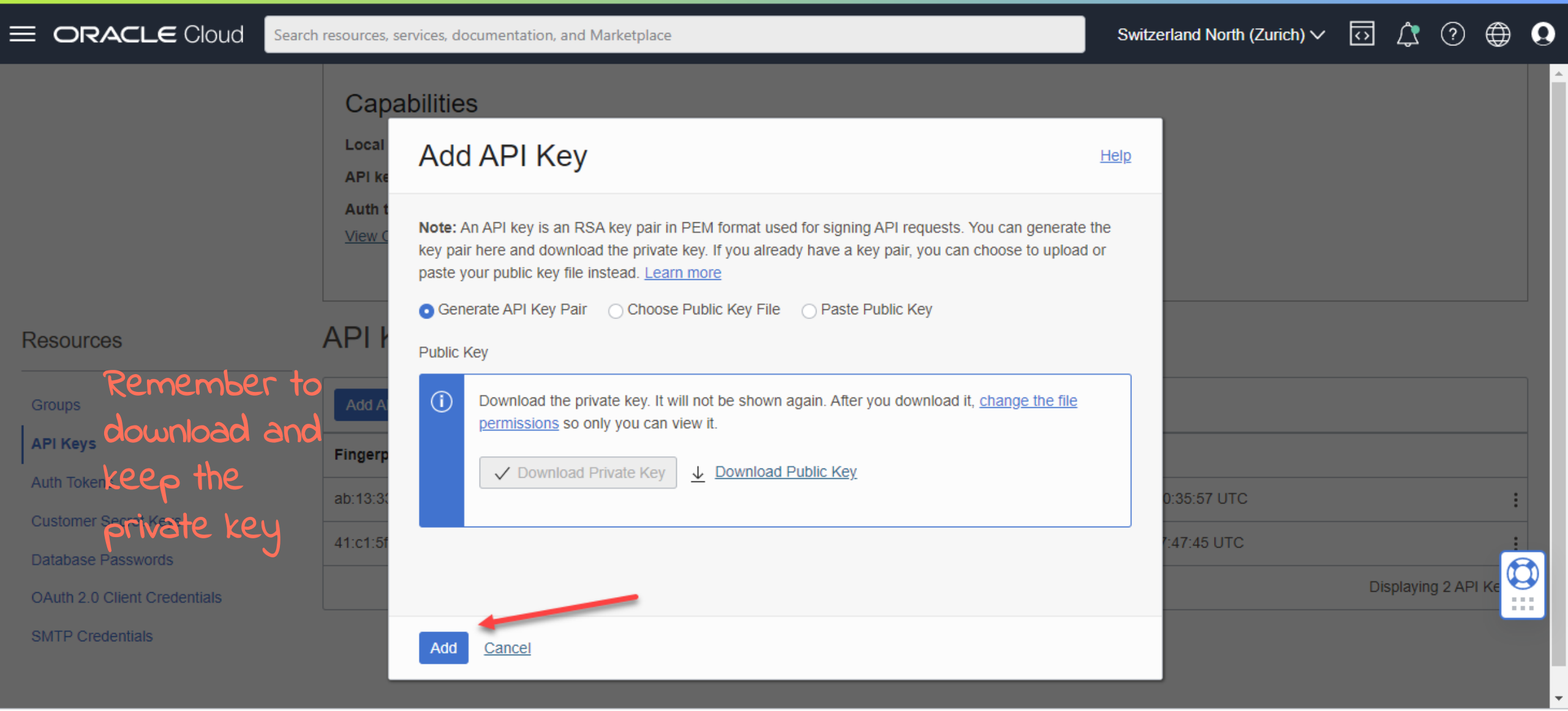
Get an API key and prepare Terraform authentication



Get an API key and prepare Terraform authentication

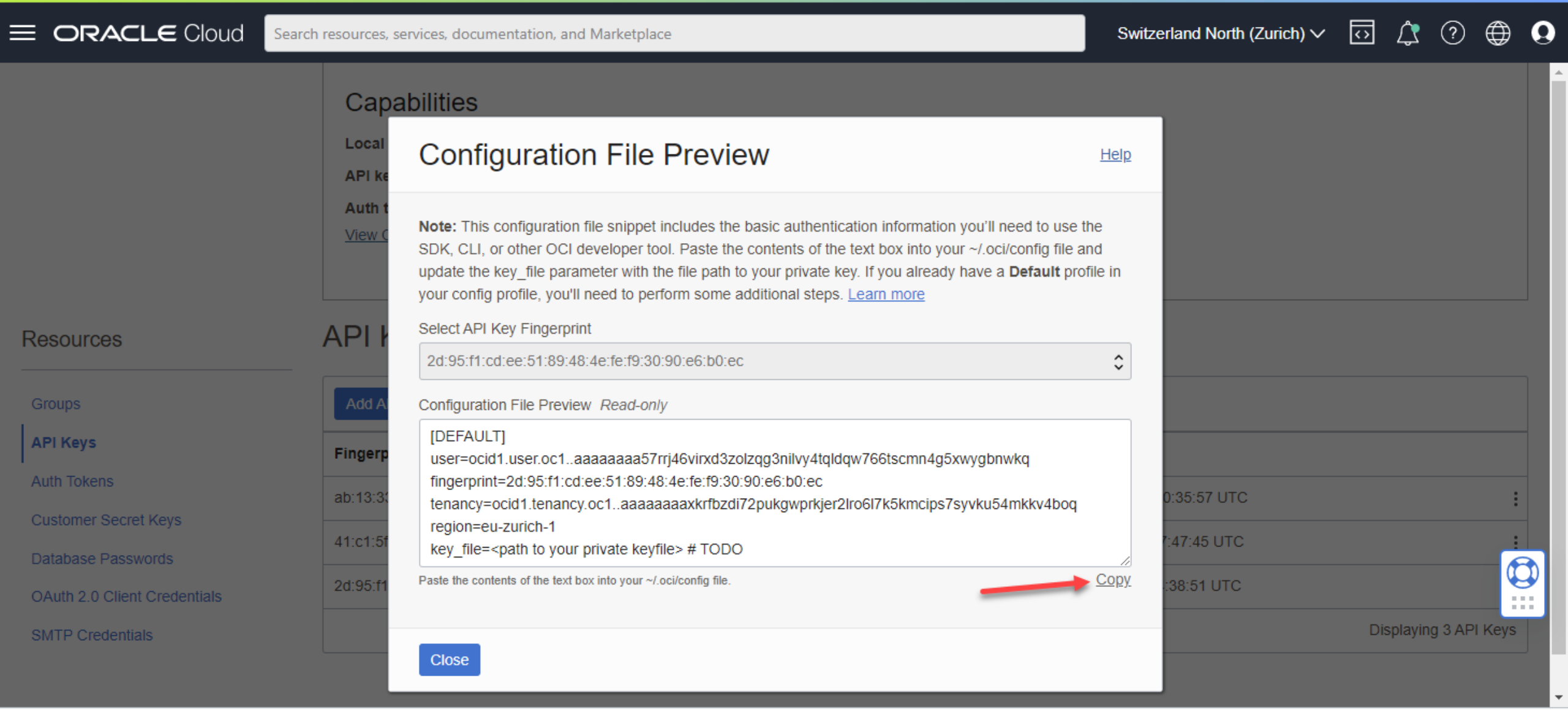


Get an API key and prepare Terraform authentication

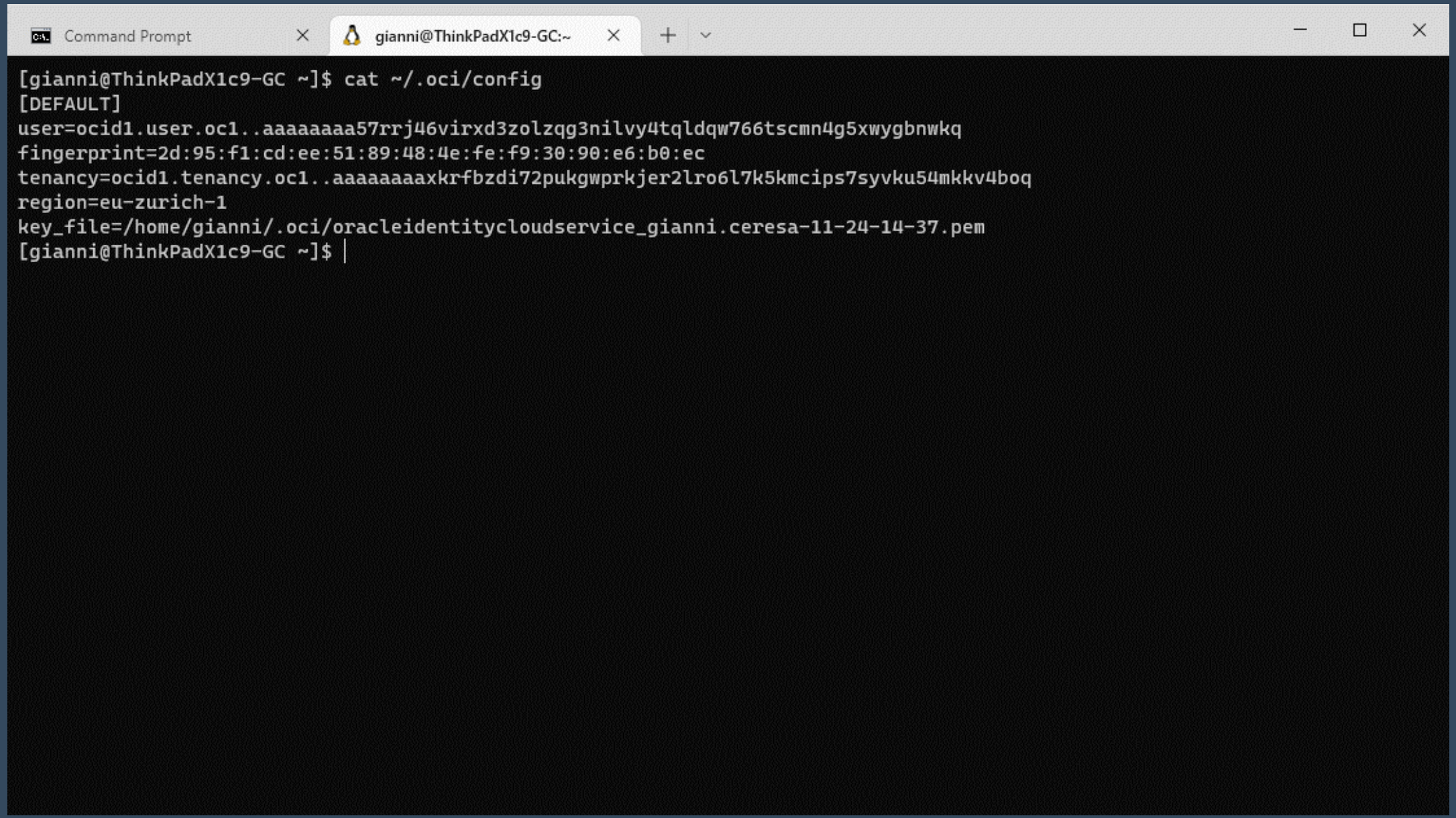


Remember to download and keep the private key

Get an API key and prepare Terraform authentication

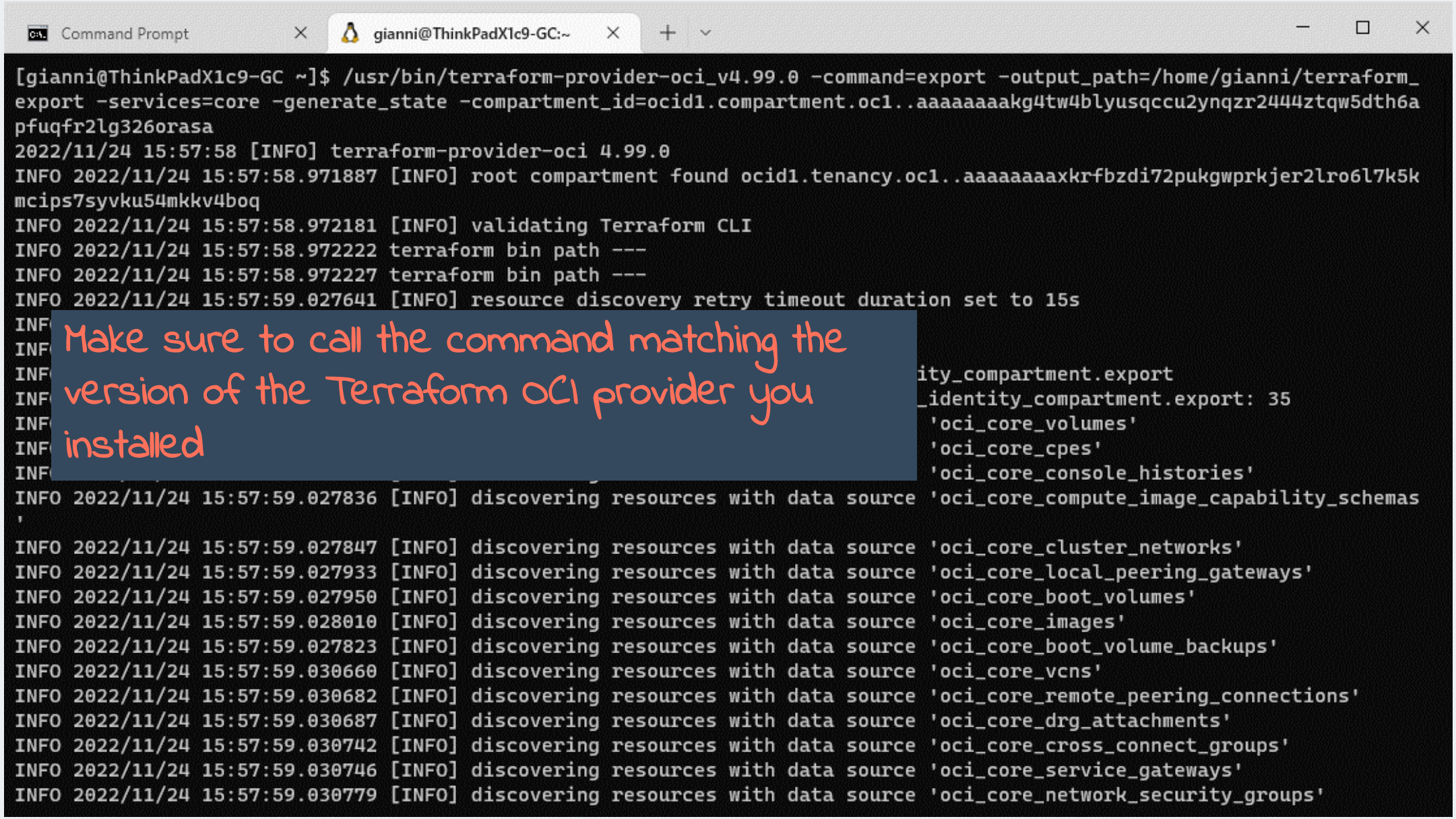


Get an API key and prepare Terraform authentication

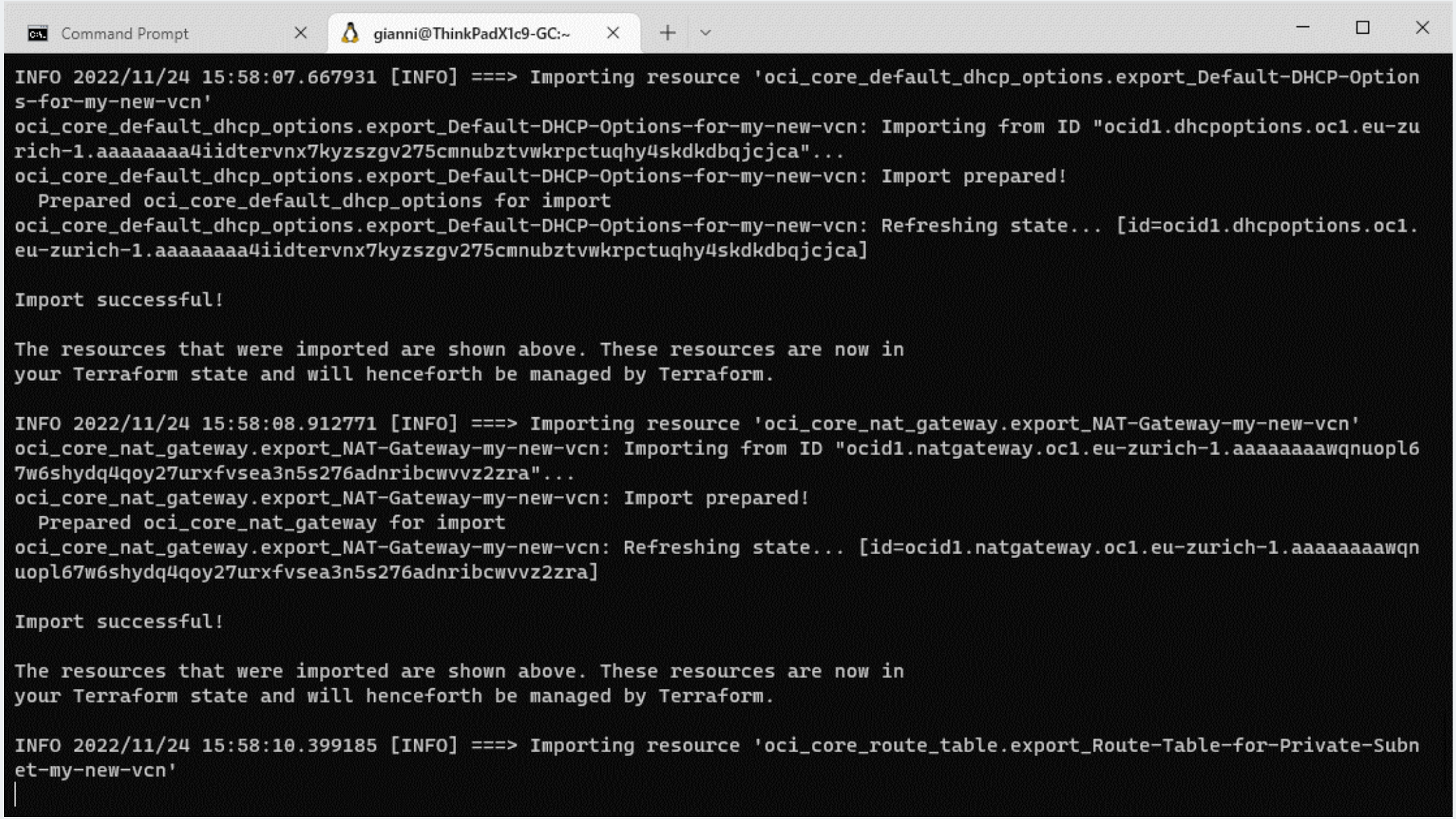


Export the existing cloud objects to a Terraform configuration

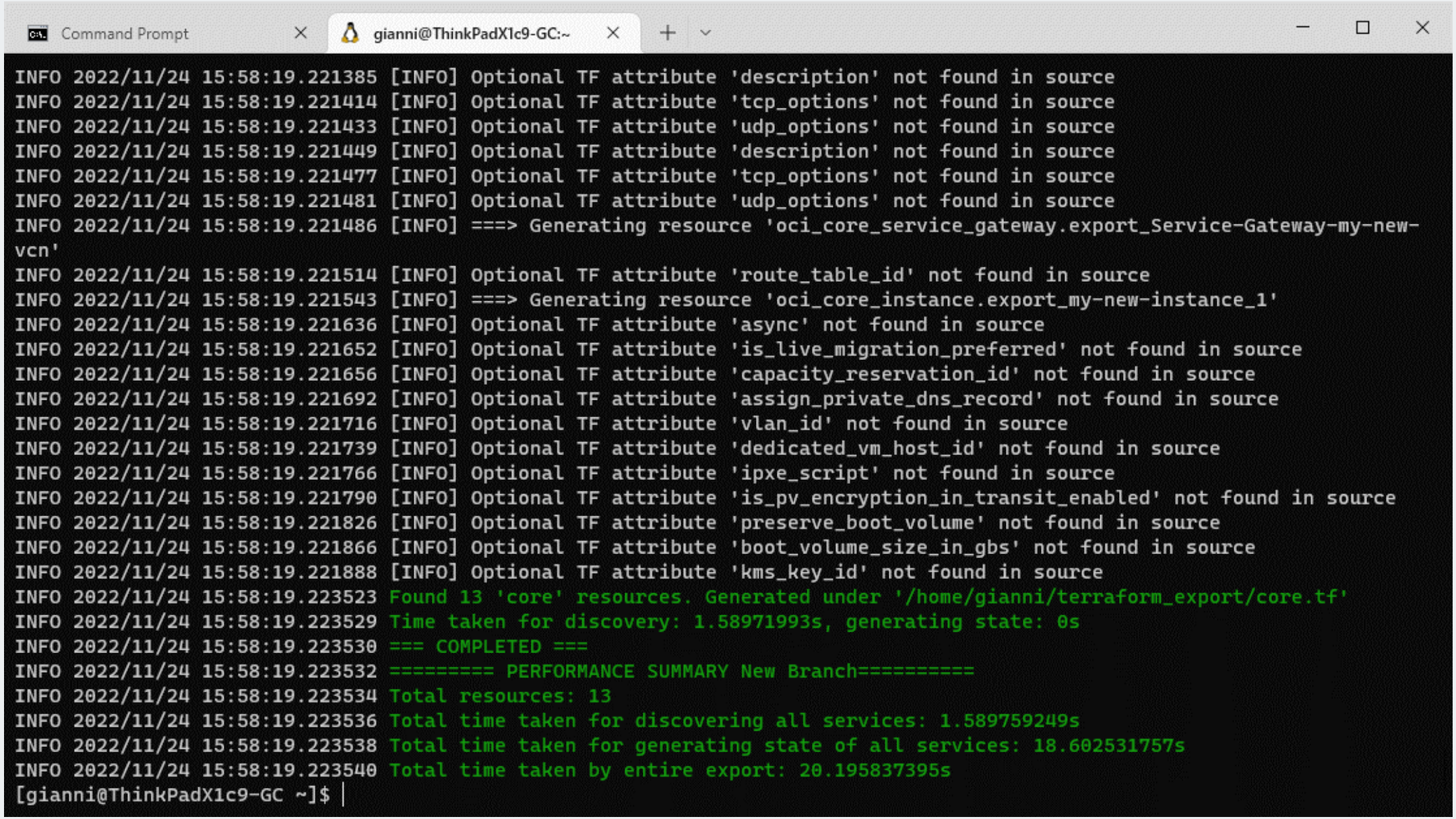
Terraform export/discovery of existing objects (with state file)



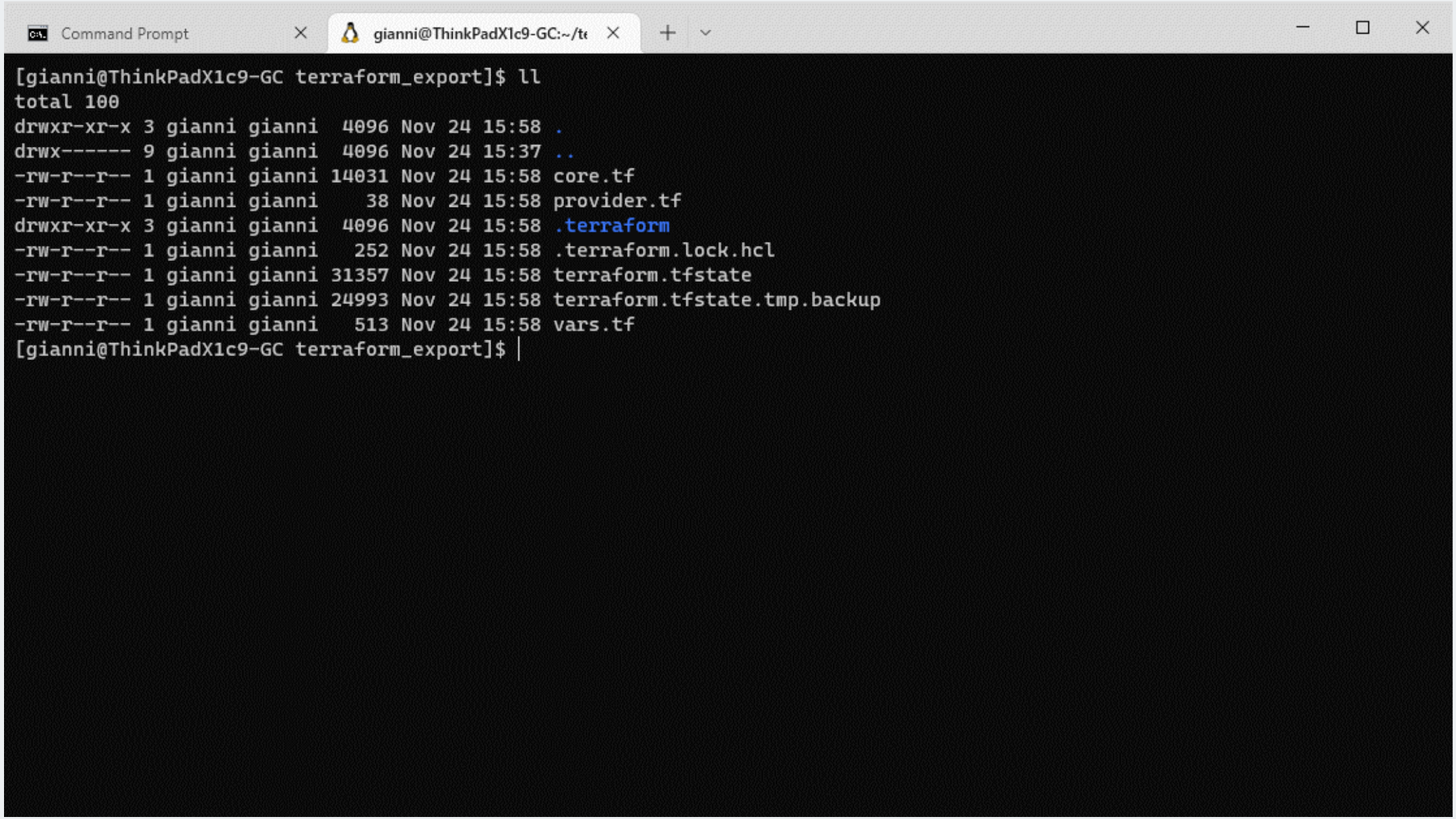
Terraform export/discovery of existing objects (with state file)



Terraform export/discovery of existing objects (with state file)

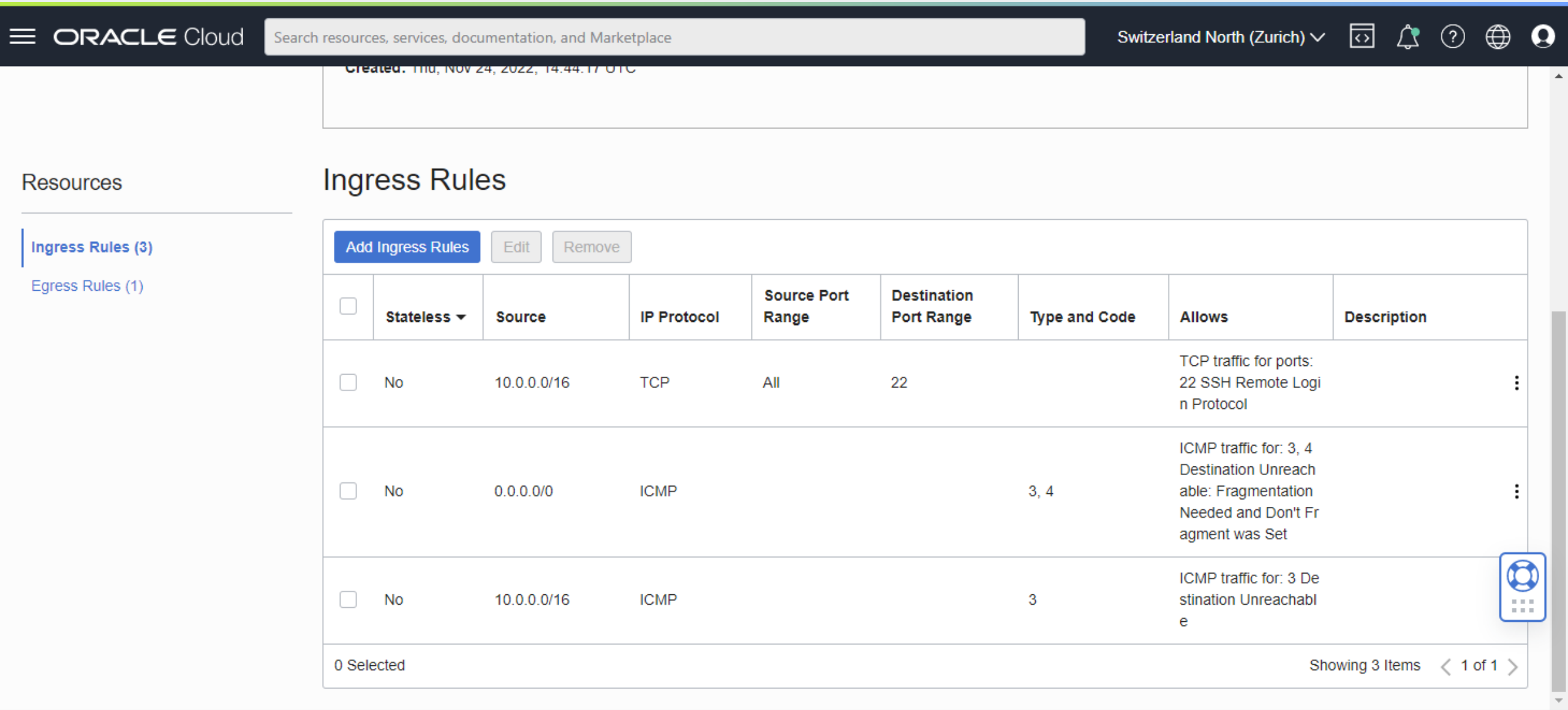


Terraform export/discovery of existing objects (with state file)

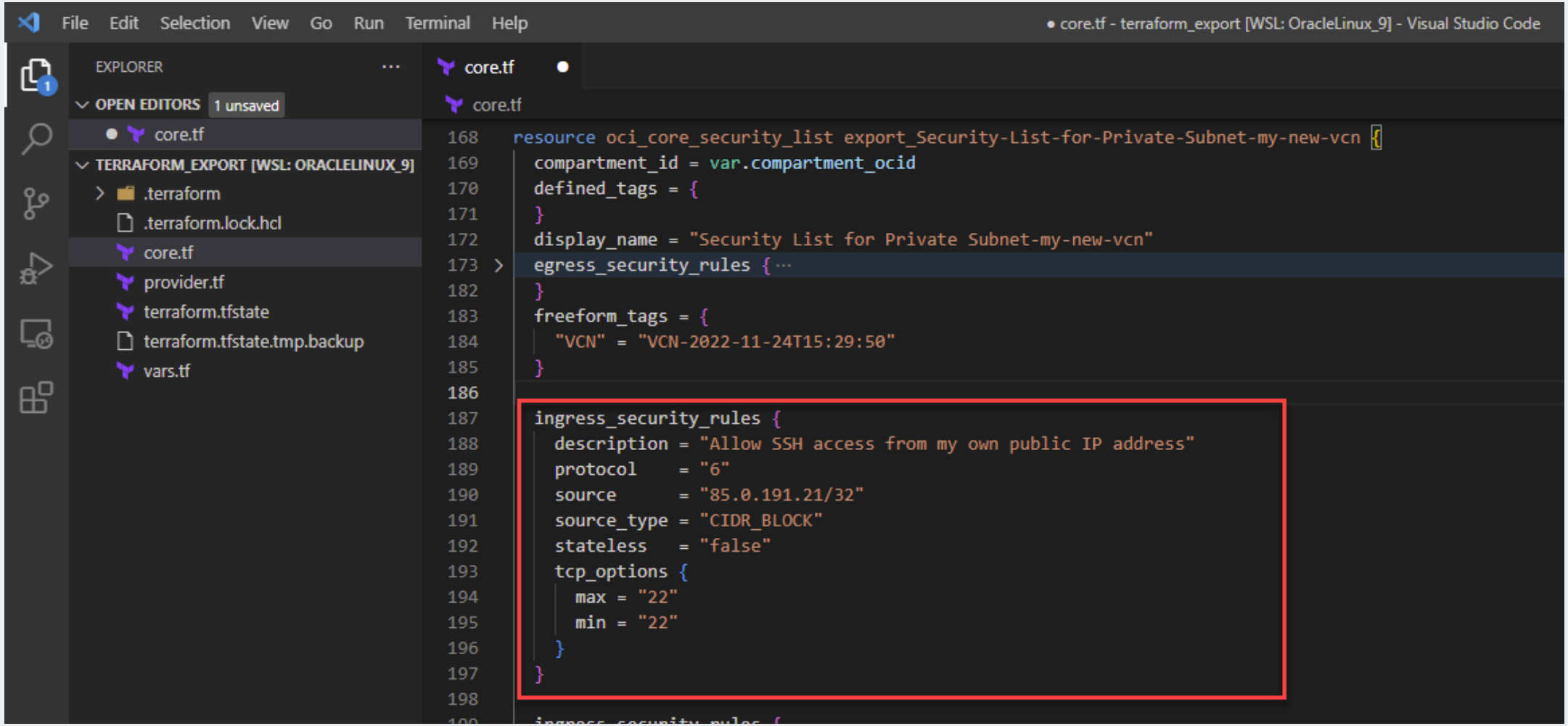


Modify the export configuration to add changes

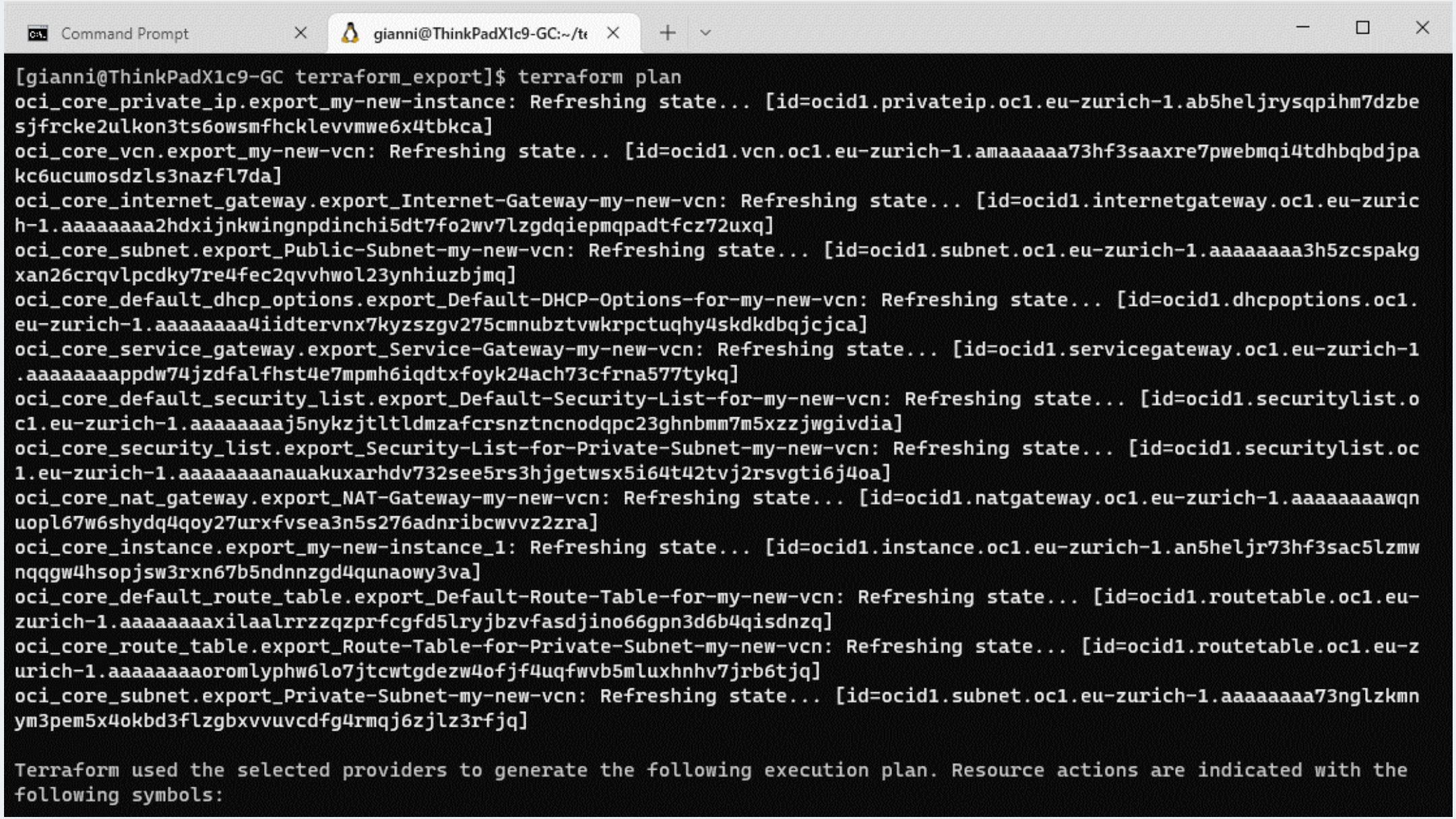
Add new Ingress Rule to an existing VCN Security List



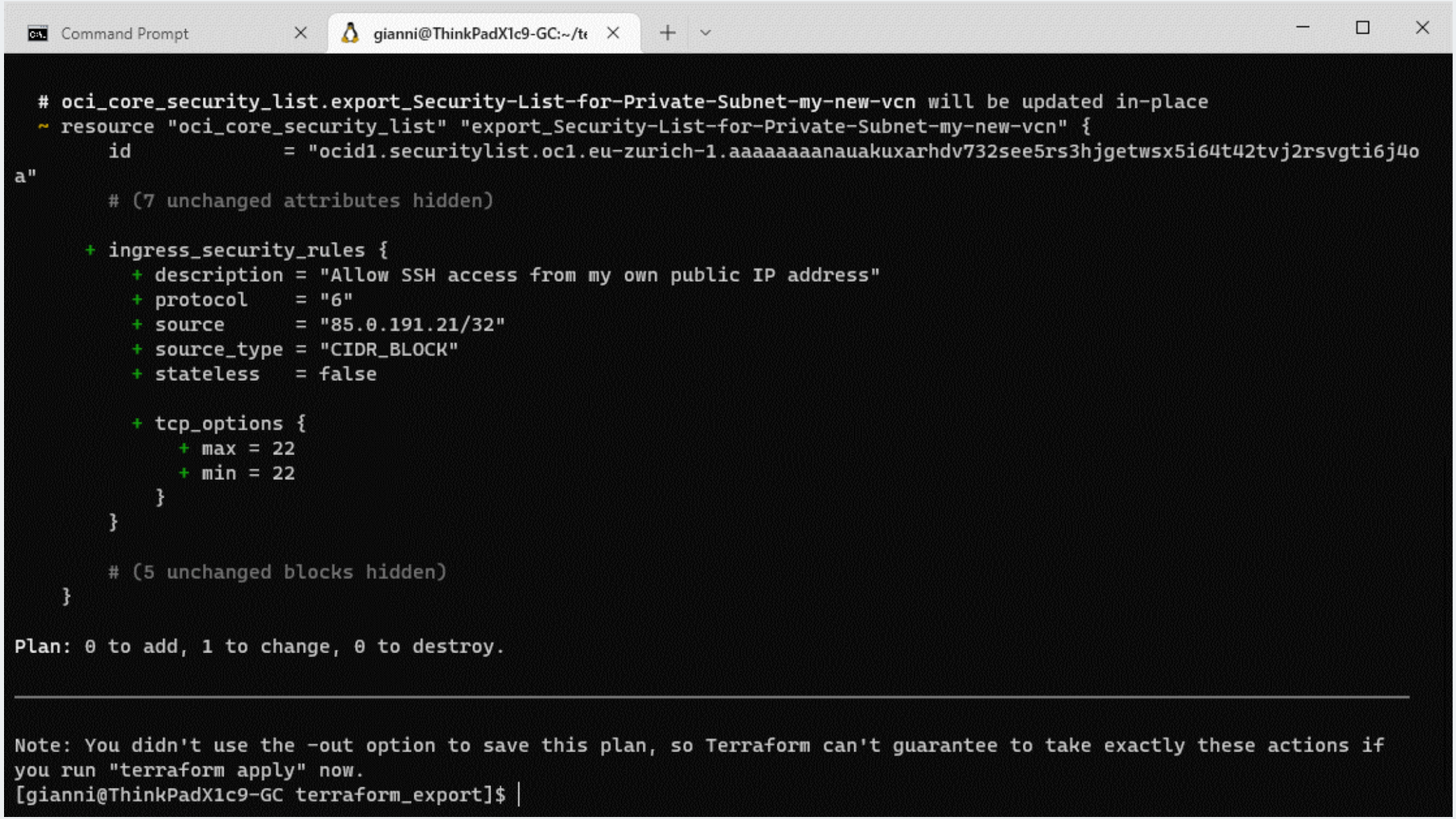
Add new Ingress Rule to an existing VCN Security List



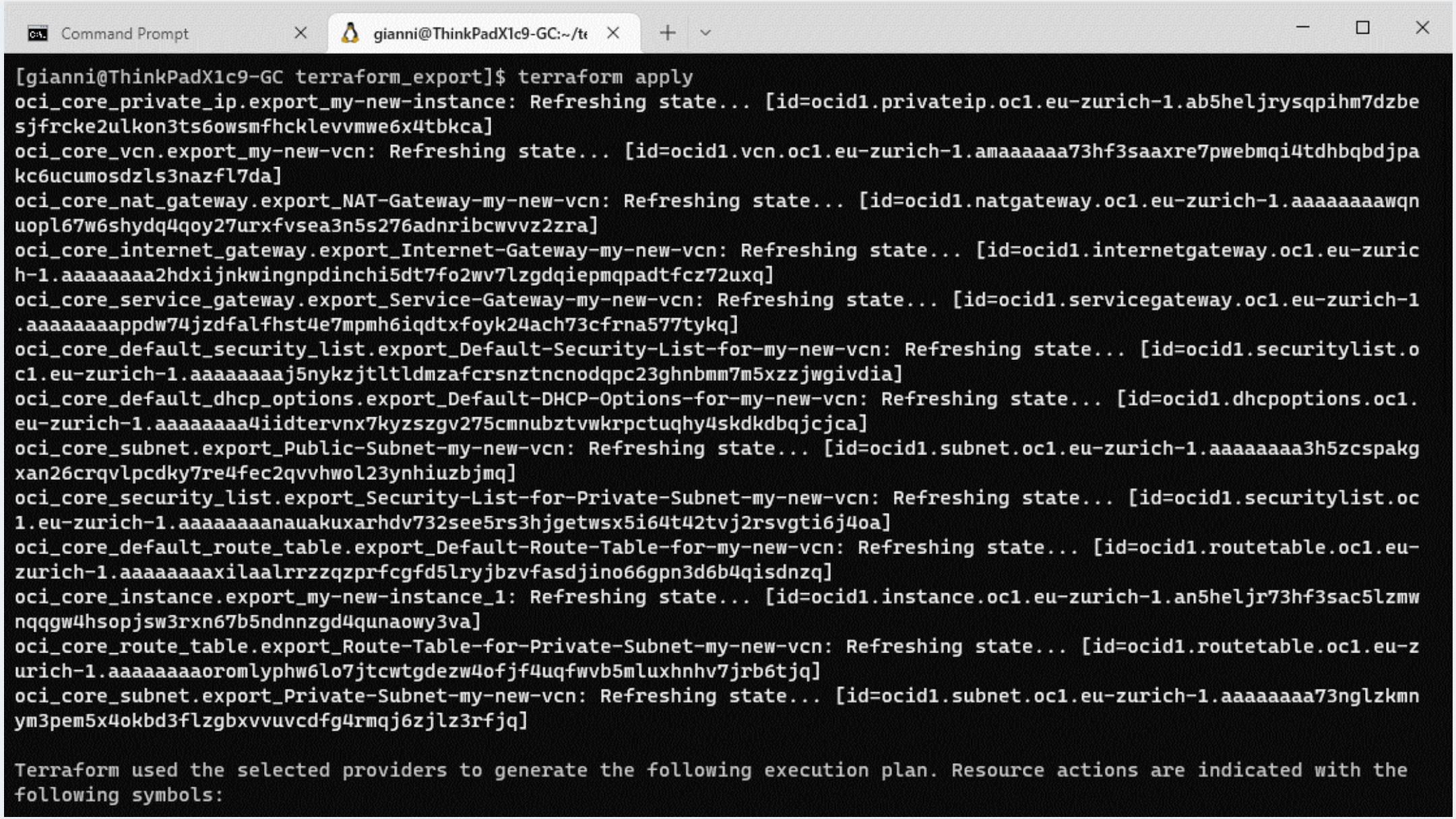
Add new Ingress Rule to an existing VCN Security List



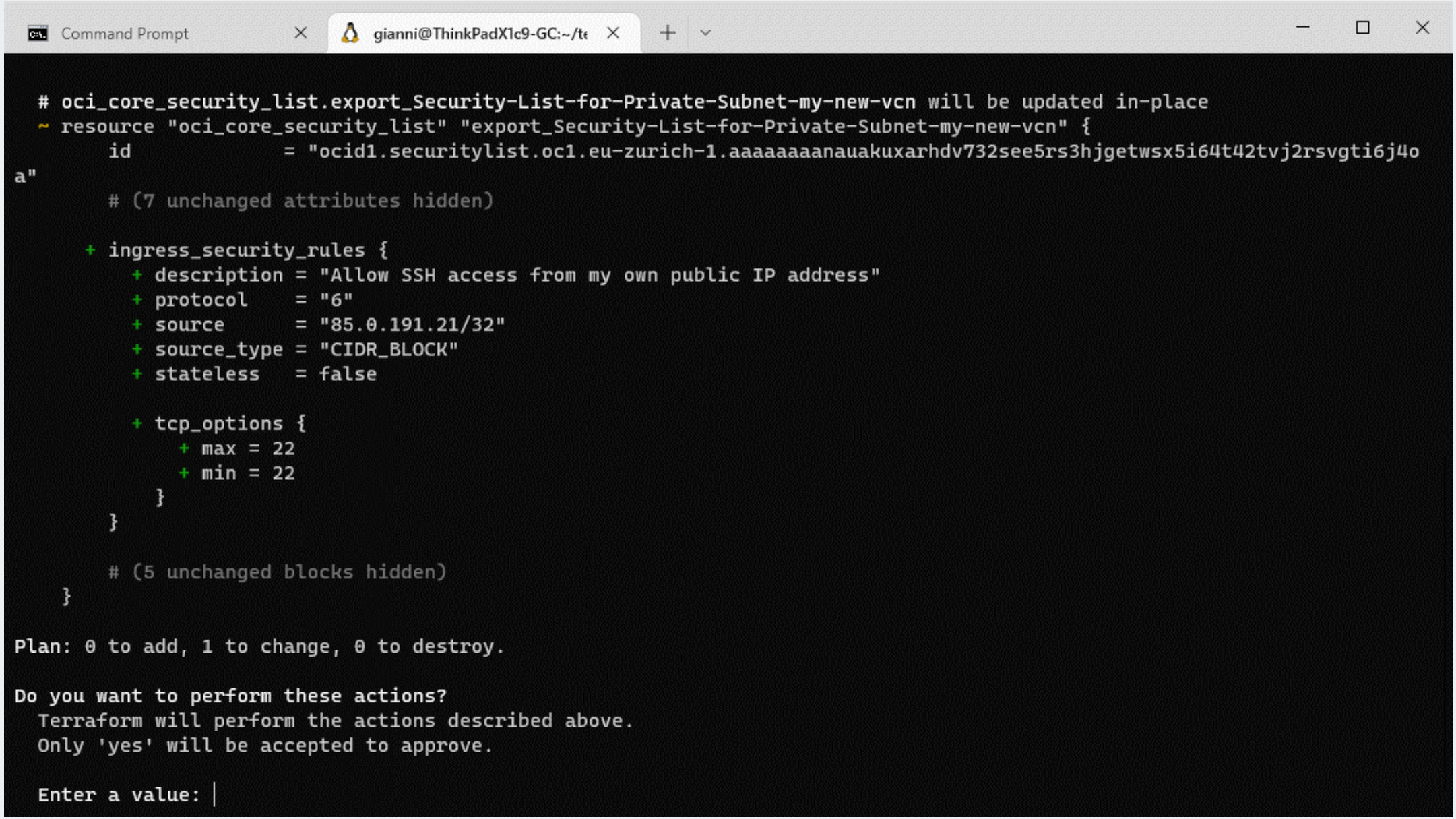
Add new Ingress Rule to an existing VCN Security List



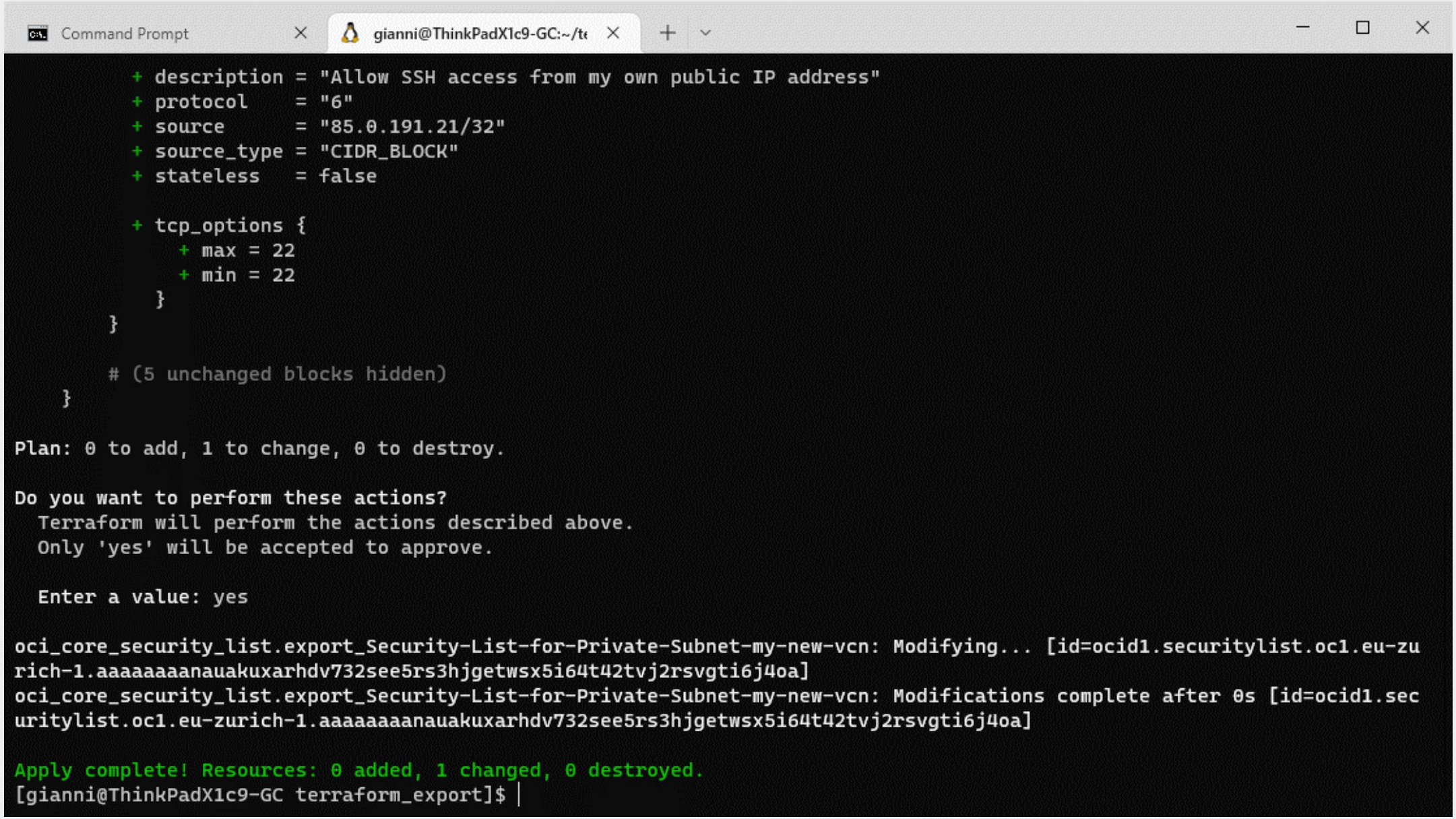
Add new Ingress Rule to an existing VCN Security List



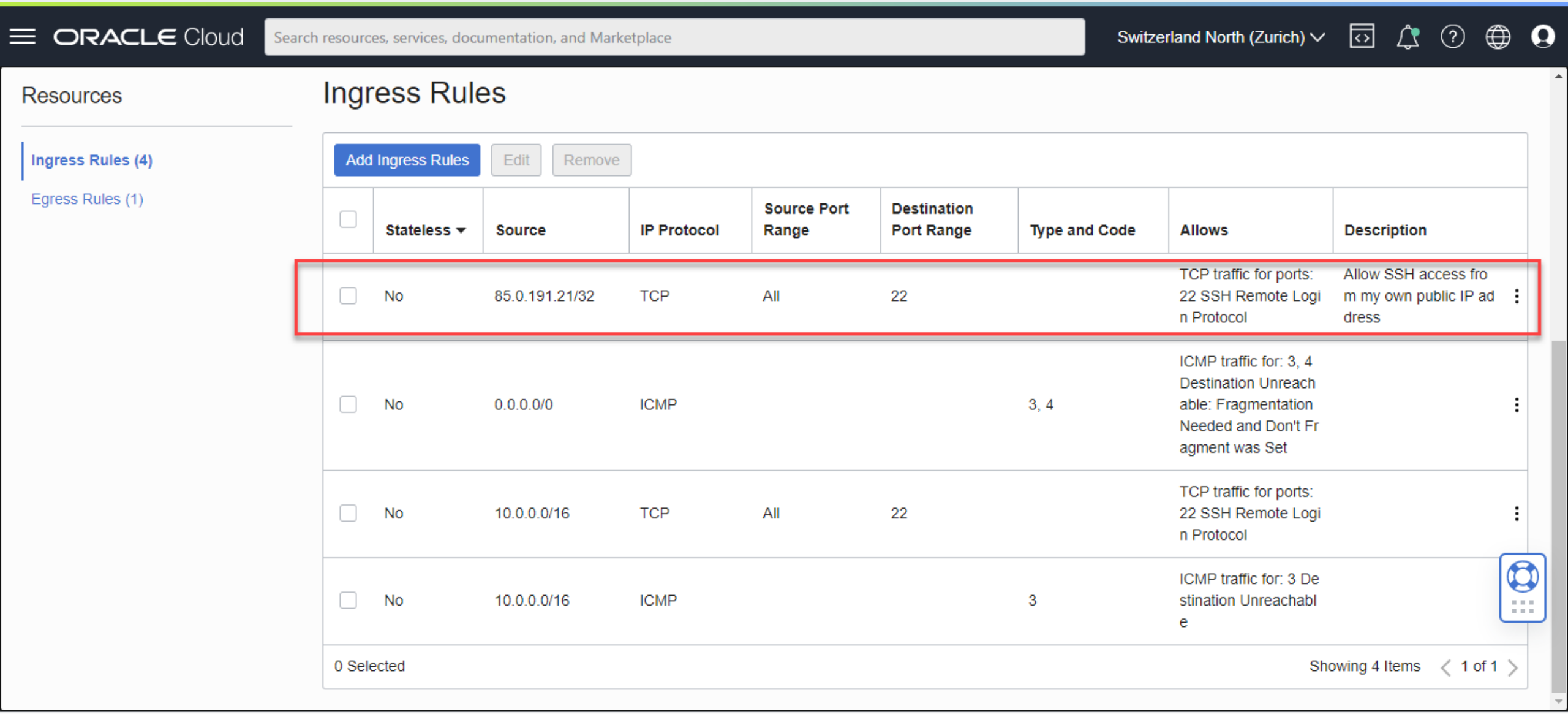
Add new Ingress Rule to an existing VCN Security List



Add new Ingress Rule to an existing VCN Security List



Add new Ingress Rule to an existing VCN Security List



Terraform Modules

Terraform Modules

A module is as simple as a set of Terraform files in a directory

It's a kind of template doing various things, creating multiple things

- Remember the VCN creating wizard? That's what a Terraform Module can do

Allows to reuse things without duplicating again and again the code

List of public existing modules

https://registry.terraform.io/browse/modules?provider=oci

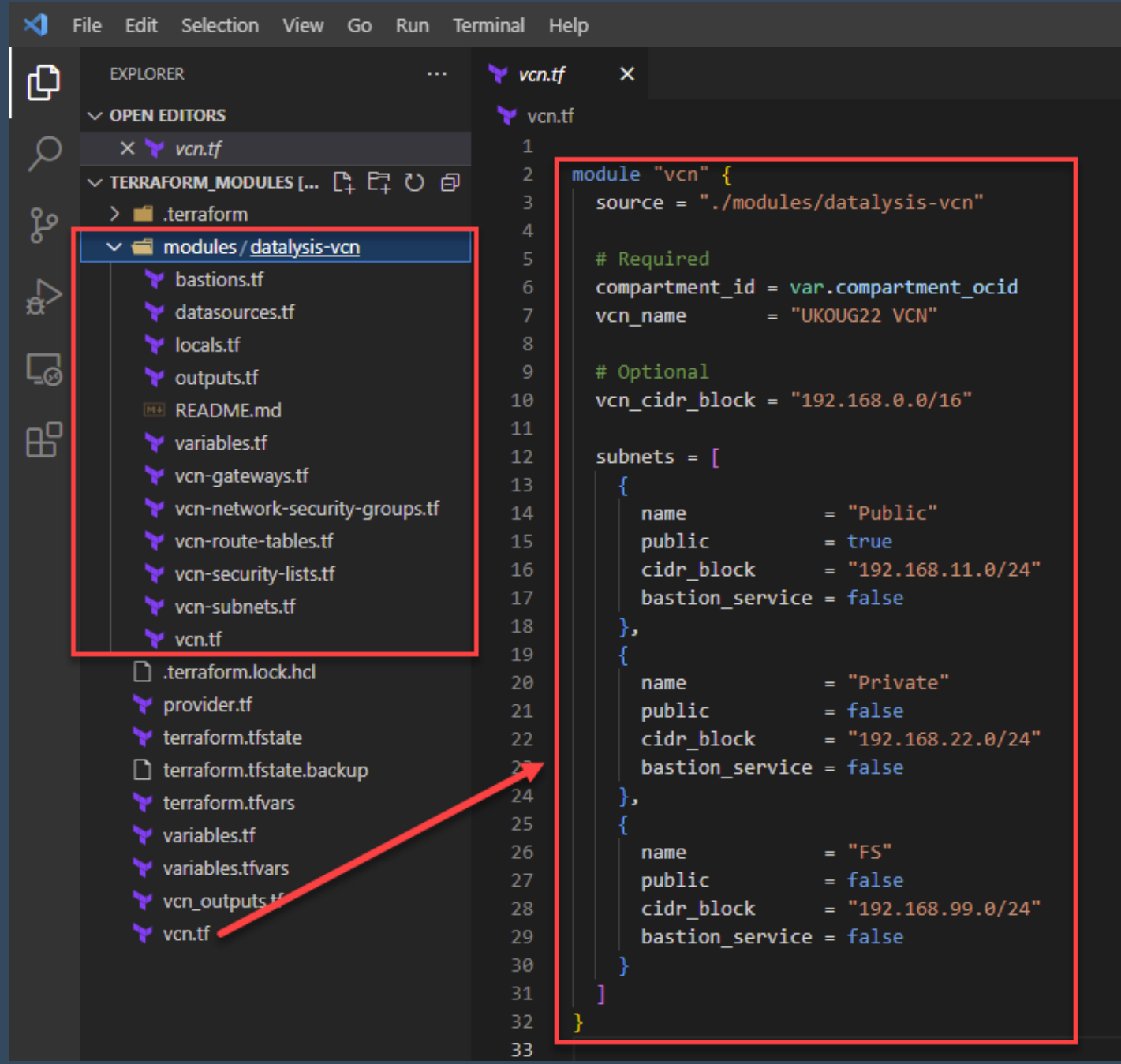
You can write your own...

Terraform Modules

I do like my VCNs done in my own way, with my naming convention and rules...

That is why I created my own VCN module

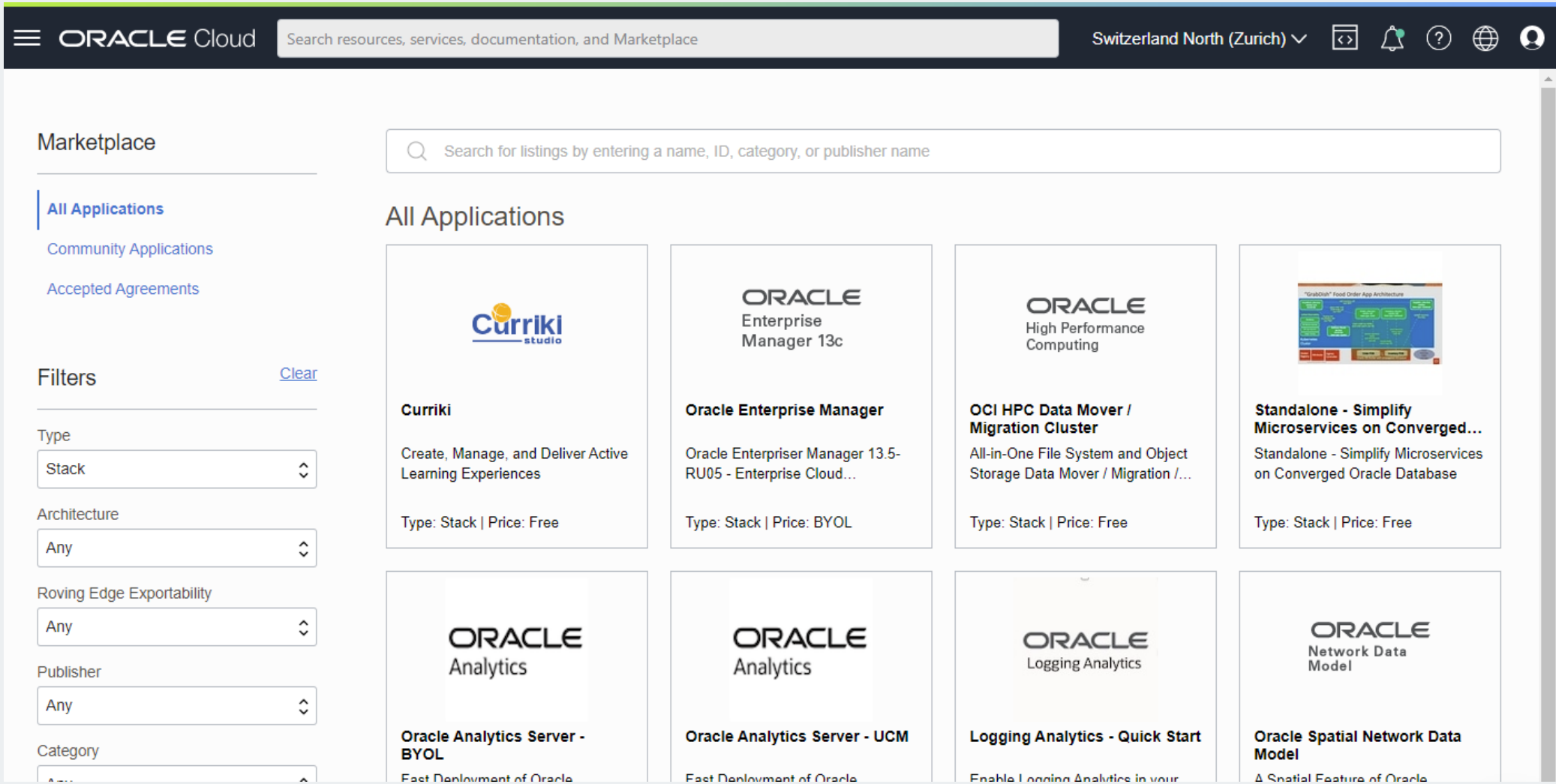
- Setup gateways
- My Security Lists
  - Allowing all connections from my home public IP address (updated automatically through a DDNS lookup by Terraform)
- My Network Security Groups
- ...



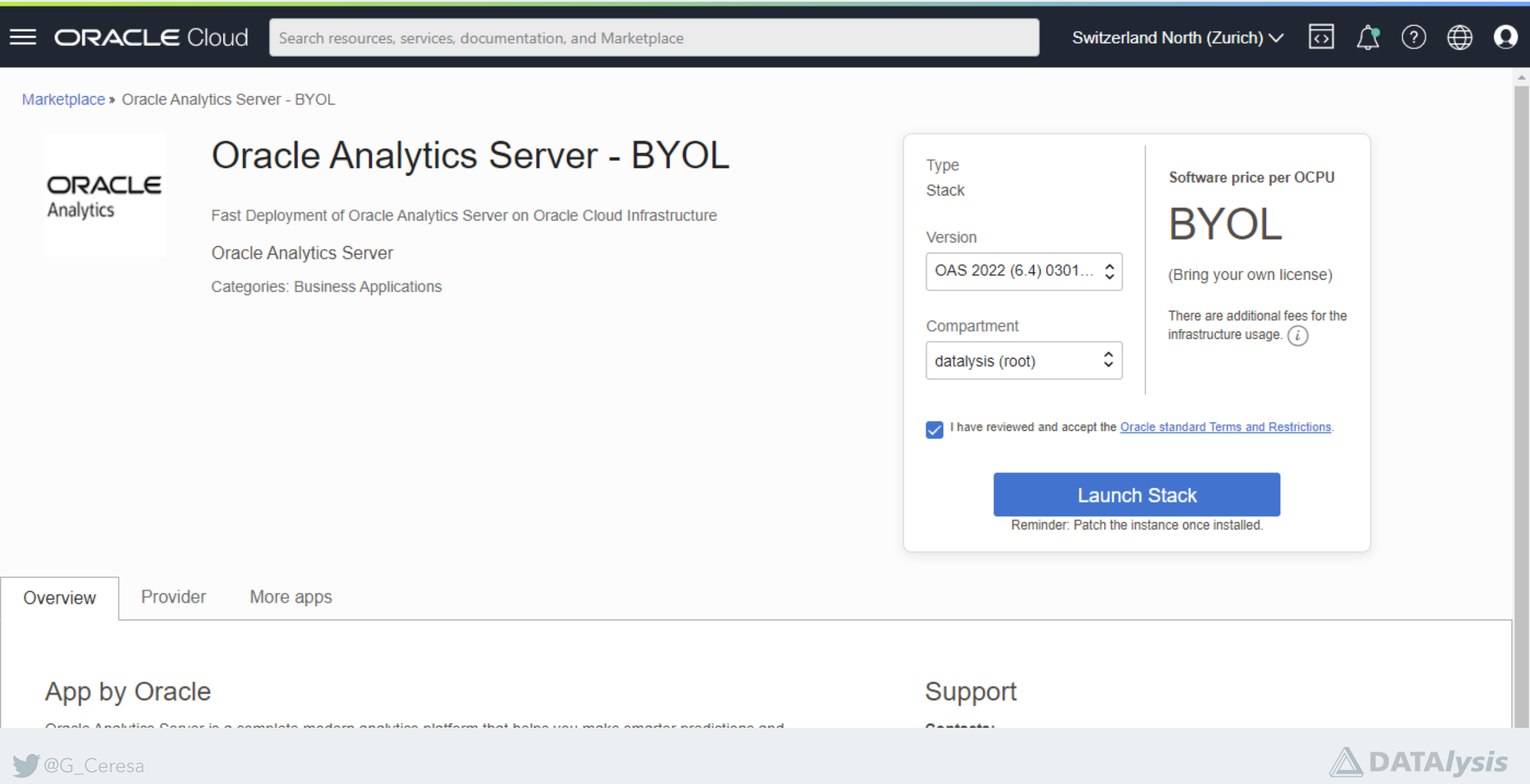
Oracle Cloud Stacks available in the Marketplace

Oracle Cloud Marketplace : many stacks available

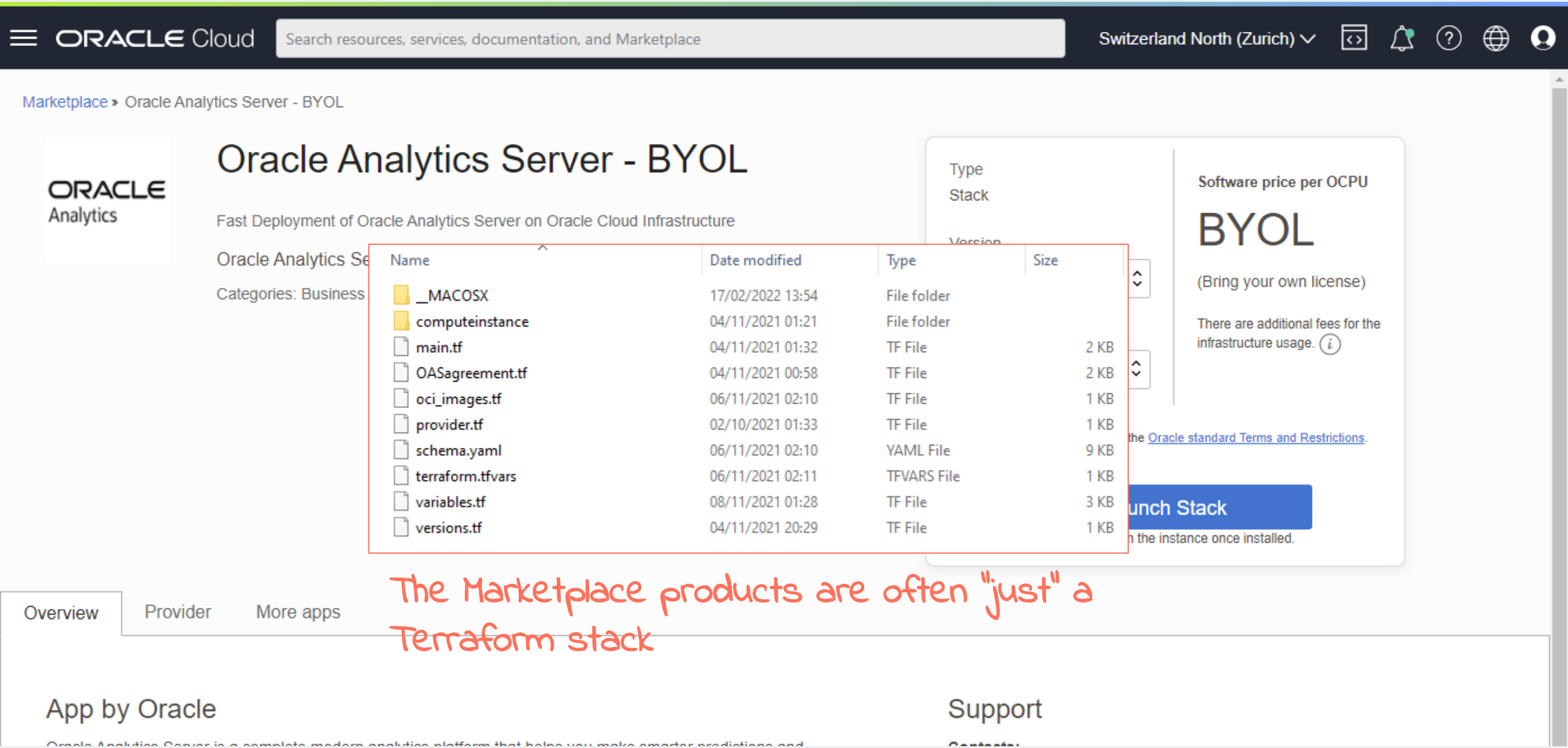
A stack is just a set of Terraform files, often with a form asking to enter values to customize the installation (set Terraform variables values)



Oracle Cloud Marketplace : many stacks available



Oracle Cloud Marketplace : many stacks available



Oracle Cloud Marketplace : many stacks available

ORACLE Cloud | Search resources, services, documentation, and Marketplace | Switzerland North (Zurich)

Create stack

Help

1. **Stack information**
2. Configure variables
3. Review

Your application will launch as part of a stack that includes the infrastructure resources required to ensure that the application deploys and runs properly.

Stack information

Oracle Analytics Server - BYOL

Oracle Analytics Server deployed on Oracle Cloud Infrastructure compute

Oracle Analytics Server will be installed on the compute instance. Do not use the auto-generated SSH private key in a production environment. Instead, generate your own SSH key-pair and upload the public key for the compute instance. To connect to the compute instance, copy the SSH private key content into a file on your machine and run the following command: ssh - i <path to the pem file> opc@<Public IP for Oracle Analytics Server compute instance>

Custom providers

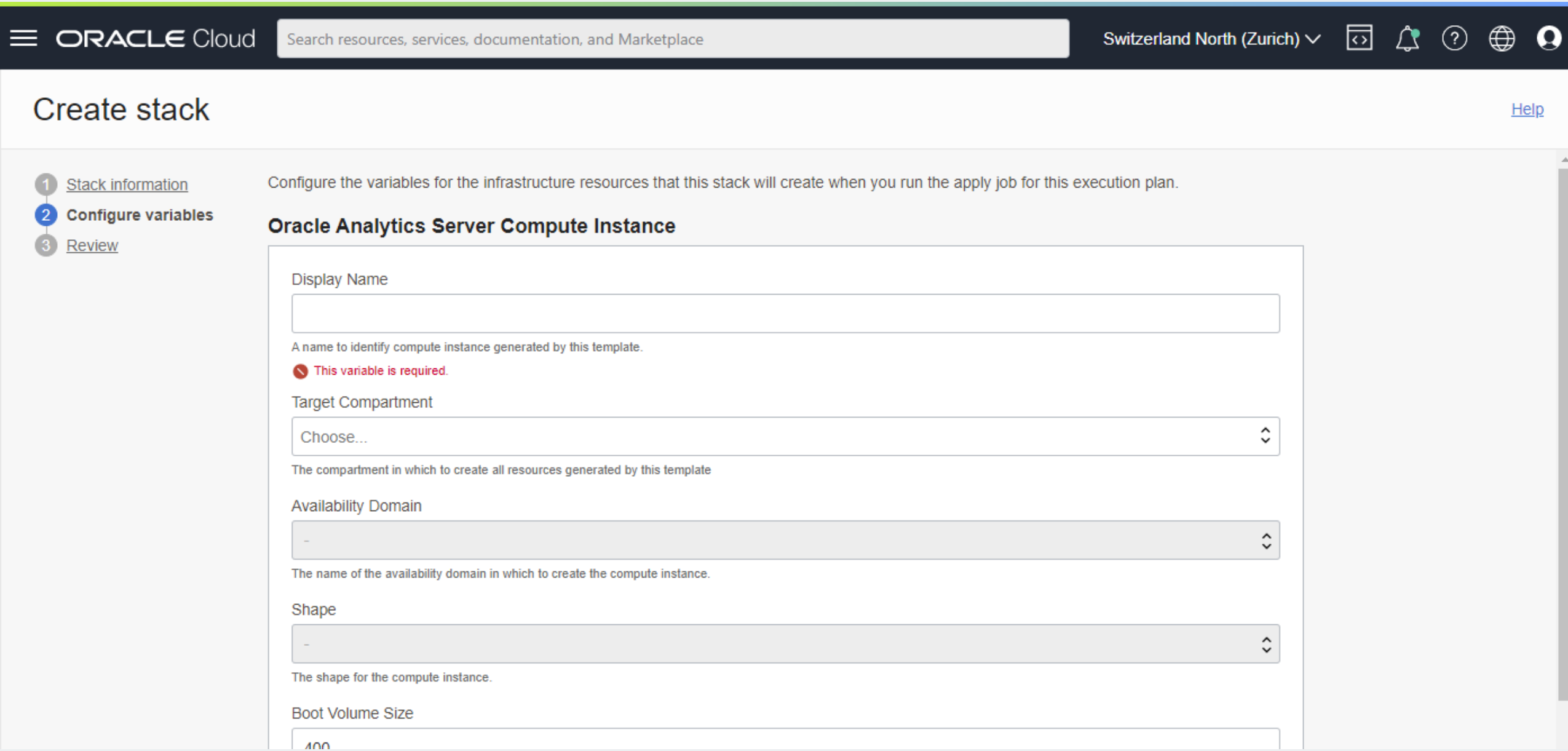
☐ Use custom Terraform providers

Store custom Terraform providers in a bucket.

Name *Optional*

Oracle Analytics Server - BYOL-20221125122555

Oracle Cloud Marketplace : many stacks available



Oracle Cloud Stacks

You can also write your own stack and load and run in Oracle Cloud

Final words...

Is Terraform something you should look into?

- If you do have some cloud objects, Terraform is something you can look into it
- If you do have everything in the cloud (lot of objects), Terraform is something you MUST look into

- Start small, expand more and more in an iterative way
  - Refactoring existing Terraform configuration to automate even more
  - Creating modules for repetitive tasks

- Warning: with an Always Free account it does work, but the account has limits on the number of resources you can create
  - Terraform can’t override those limits and will fail in doing the task