Ivan Danyliuk

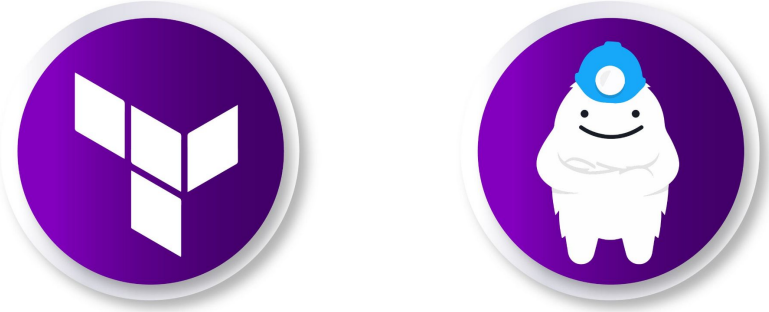
Terraform & Terragrunt

Terraform & Terragrunt

Prometheus,
Prometheus Node Exporter

Grafana

Terraform & Terragrunt



Terraform & Terragrunt

Terraform is a popular infrastructure-as-code software tool built by HashiCorp. I used it to provision all kinds of infrastructure and services.

Terragrunt is a thin wrapper of Terraform maintained by Gruntwork allowing to

- keep your configurations DRY
- work with multiple Terraform modules
- manage remote state.

Ivan Danyliuk

Terragrunt

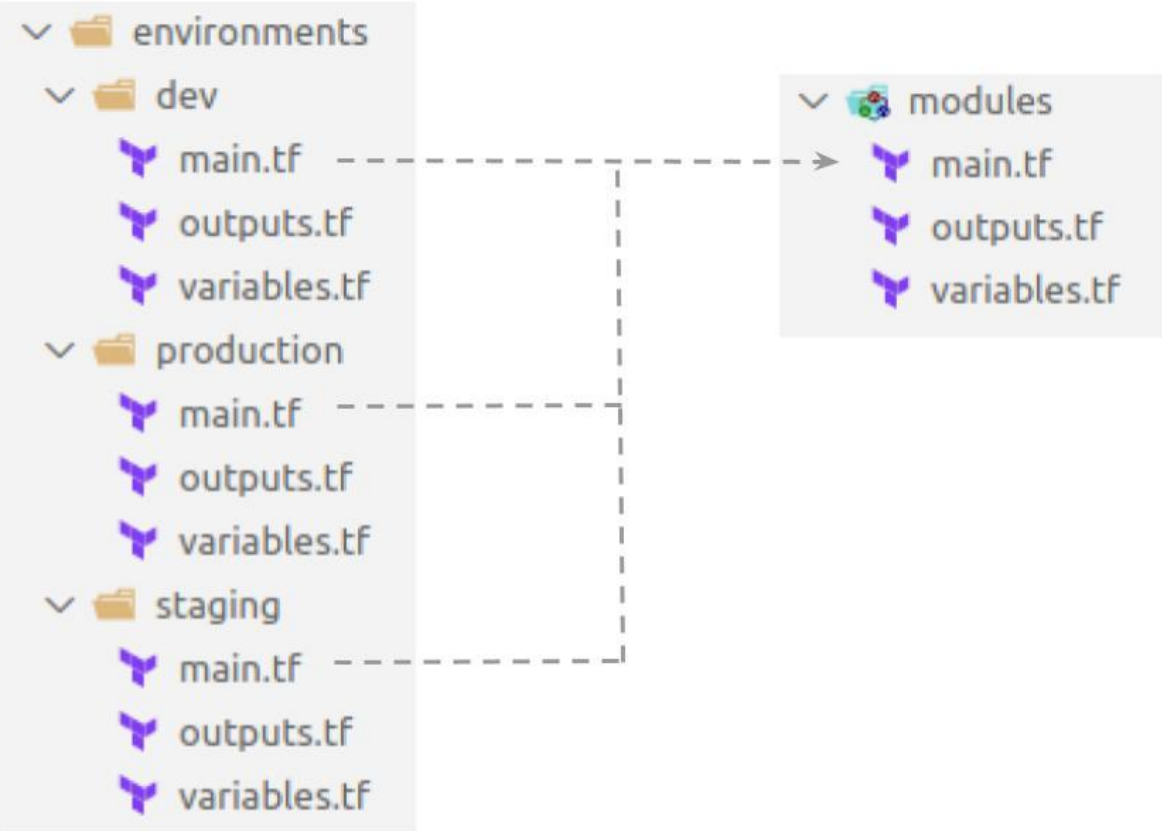
DRY - Don’t Repeat Yourself, instead of Write Everything Twice.

Key features

- Keep your backend configuration DRY
- Keep your provider configuration DRY
- Keep your Terraform CLI arguments DRY
- Promote immutable, versioned Terraform modules across environments

Ivan Danyliuk

Infrastructure environments



Problem: code repetitions

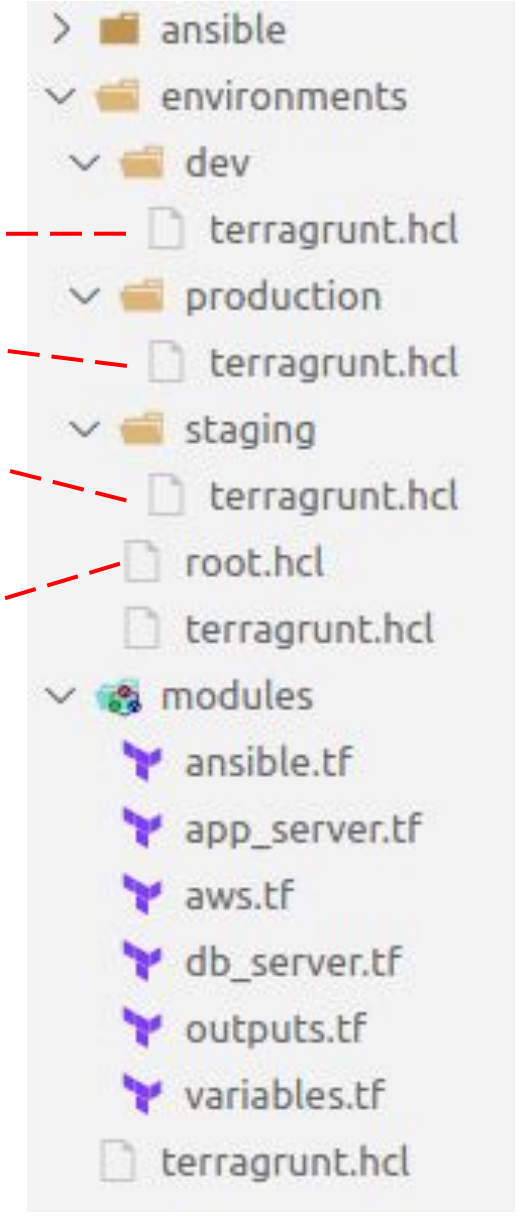
Solution: form general code as modules and reuse them in every environment creation process

Ivan Danyliuk

Terragrunt project structure

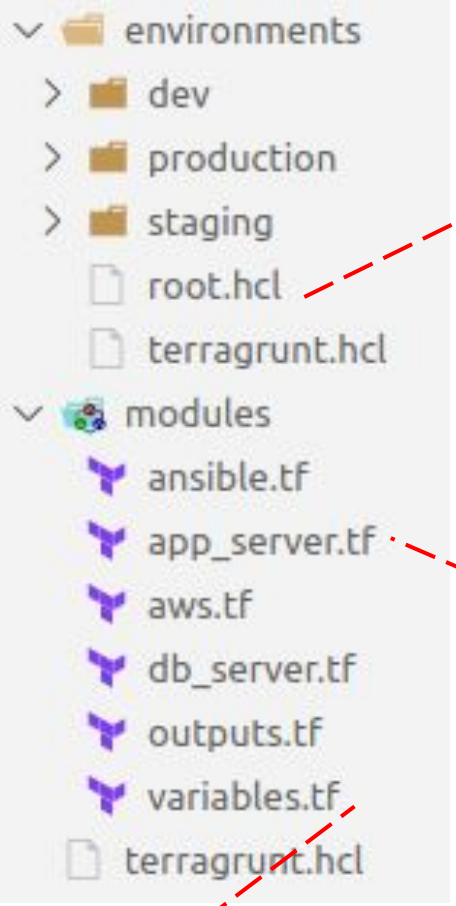
```
include {
  path = find_in_parent_folders("root.hcl")
}
```

```
locals {
  root_deployments_dir       = get_parent_terragrunt_dir()
  relative_deployment_path   = path_relative_to_include()
  deployment_path_components = compact(split("/", local.relative_deployment_path))
  env = reverse(local.deployment_path_components)[0]
}

terraform {
  source = "${local.root_deployments_dir}/..//modules"
}

remote_state {
  backend = "s3"
  config = {
    bucket         = "igeocitizen-terraform-state"
    key            = "${path_relative_to_include()}/terraform.tfstate"
    region         = "eu-north-1"
    encrypt        = true
    dynamodb_table = "igeocitizen-lock-table-${path_relative_to_include()}"
  }
}
inputs = {
  environment = local.env
  root_path   = "${get_parent_terragrunt_dir()}/../ansible/environments/${path_relative_to_include()}"
}
```



Ivan Danyliuk

Terragrunt variables (environment variable)



variable value set by terragrunt

```
1  locals {
2    root_deployments_dir       = get_parent_terragrunt_dir()
3    relative_deployment_path   = path_relative_to_include()
4    deployment_path_components = compact(split("/", local.relative_deployment_path))
5    env = reverse(local.deployment_path_components)[0]
6  }
7
8  terraform {
9    source = "${local.root_deployments_dir}/..//modules"
10 }
11
12 > remote_state {…
21 }
22 inputs = {
23   environment = local.env
24   root_path   = "${get_parent_terragrunt_dir()}/../ansible/environments/${path_relative_to_include()}"
25 }
```

variable using

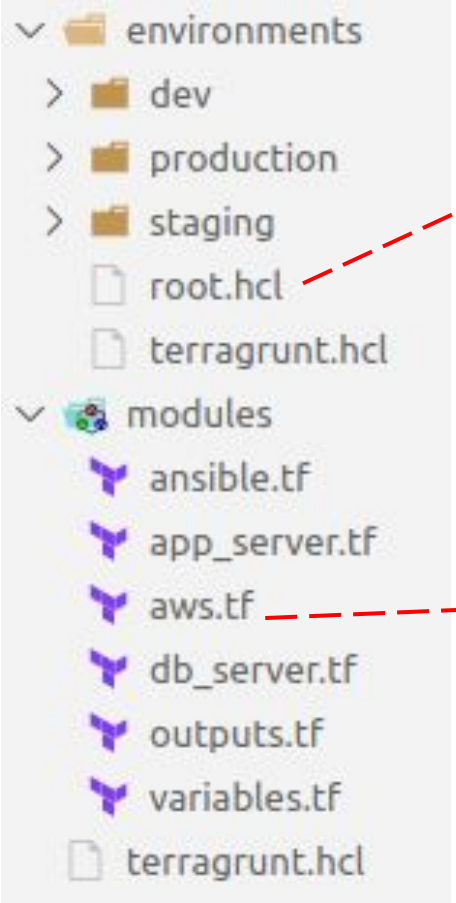
```
2  resource "aws_instance" "App_Ubuntu_Terraform" {
3    ami                    = "ami-000e50175c5f86214"
4    instance_type          = "t3.micro"
5    key_name               = "ATC"
6    vpc_security_group_ids = [aws_security_group.sgh_app.id]
7  > credit_specification {…
9    }
10   tags = {
11     Name    = "App_${var.environment}"
12     Owner   = "Ivan Danyliuk"
13     Project = "Geocitizen"
14   }
15 }
```

variable definition

```
1  variable "environment" {
2    description = "Environment name"
3    type        = string
4  }
```

Ivan Danyliuk

Store Terraform state in a shared location



S3 storage configuration by Terragrunt

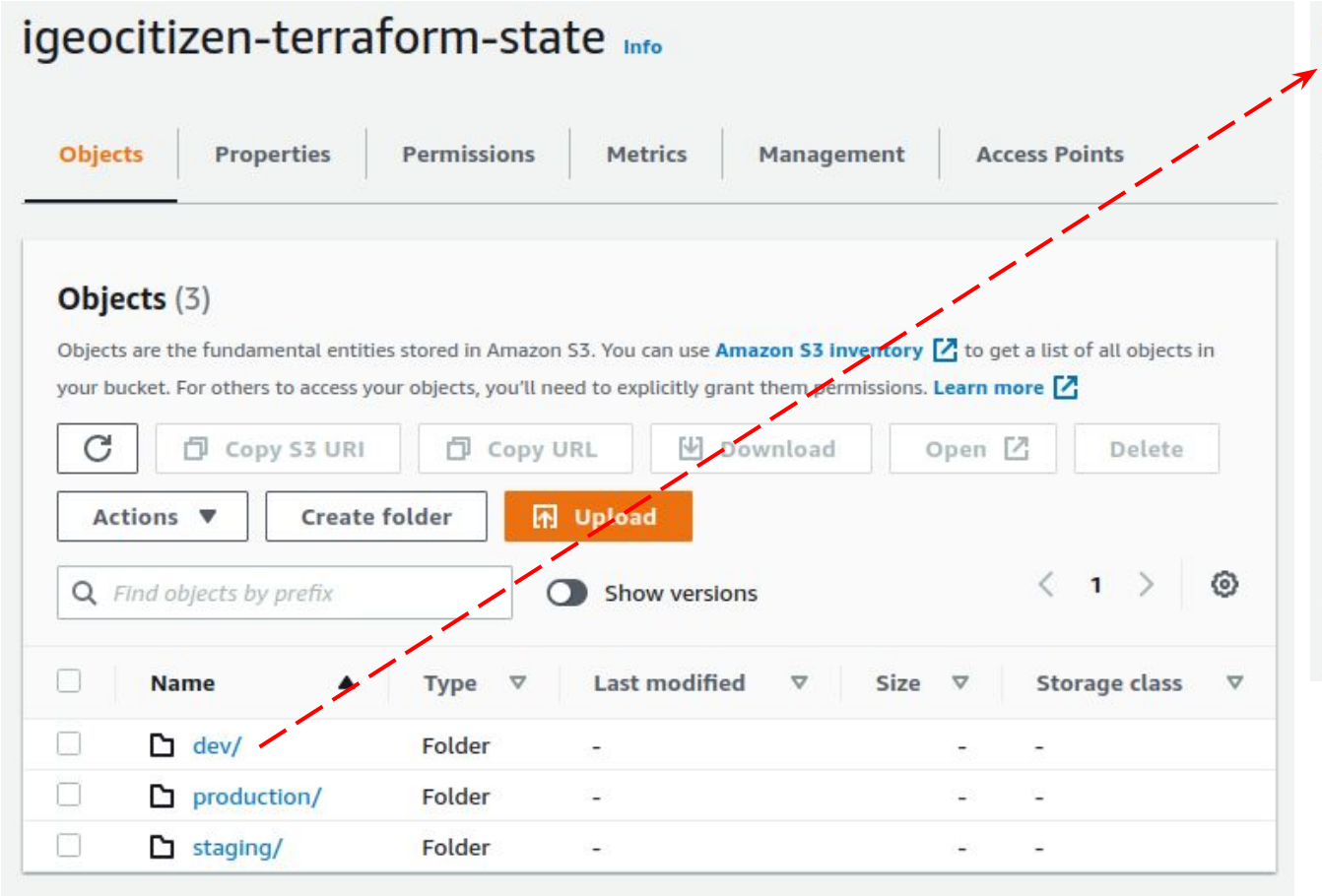
```
remote_state {
  backend = "s3"
  config = {
    bucket         = "igeocitizen-terraform-state"
    key            = "${path_relative_to_include()}/terraform.tfstate"
    region         = "eu-north-1"
    encrypt        = true
    dynamodb_table = "igeocitizen-lock-table-${path_relative_to_include()}"
  }
}
```

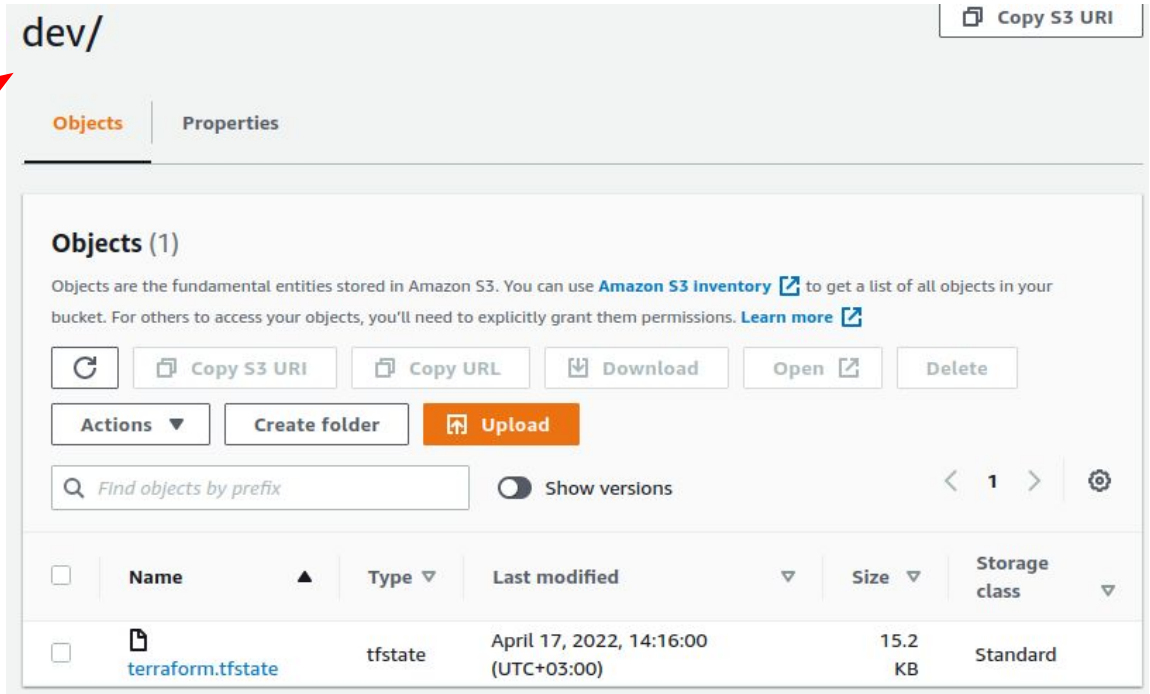
empty S3 storage configuration

```
provider "aws" {
  profile = "Personal"
  region  = "eu-north-1"
}
terraform {
  backend "s3" {}
}
```

Ivan Danyliuk

Store Terraform state in a shared location





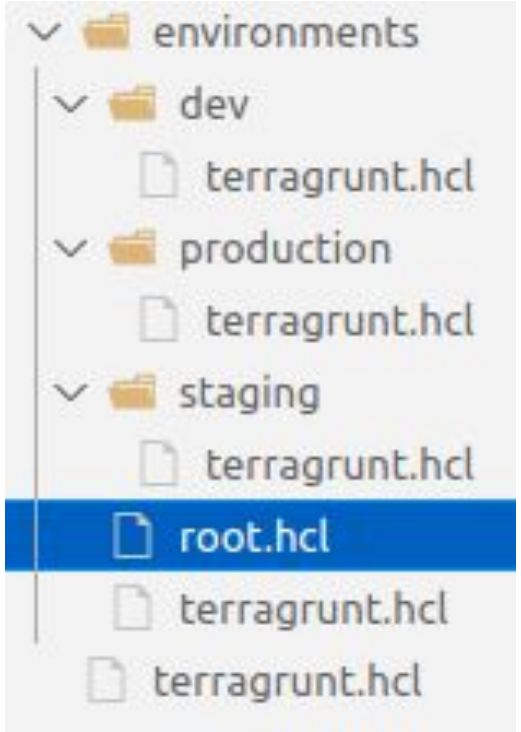
Ivan Danyliuk

Code versioning

Terragrunt allows

- to define Terraform code once
- to promote a **versioned**, immutable “artifact” of that exact same code from environment to environment

```
$ git tag -a "v0.0.1" -m "First release of app module"
$ git push --follow-tags
```




```
1  > locals {…
6    }
7
8    terraform {
9      source = "git@github.com:idanylyuk/terragrunt_test.git//modules?ref=v0.0.2"
10   }
11
12 > remote_state {…
21   }
22 > inputs = {…
25   }
```

Ivan Danyliuk

Terragrunt main commands

$ terragrunt plan
$ terragrunt apply
$ terragrunt destroy

```
ivan@Dell-NB:/media/ivan/SYS/DevOPS/TG/Test3/environments/dev$ terragrunt plan

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create
```

Or variant of these commands for all environments

$ terragrunt run-all plan

or deprecated

$ terragrunt plan-all

```
ivan@Dell-NB:/media/ivan/SYS/DevOPS/TG/Test3$ terragrunt run-all plan
INFO[0000] The stack at /media/ivan/SYS/DevOPS/TG/Test3 will be processed in the following order for command plan:
Group 1
- Module /media/ivan/SYS/DevOPS/TG/Test3/environments/dev
- Module /media/ivan/SYS/DevOPS/TG/Test3/environments/production
- Module /media/ivan/SYS/DevOPS/TG/Test3/environments/staging


Initializing the backend...
```

Ivan Danyliuk

Terragrunt creation of all environments

App_staging	i-0688c25c8c07eae9d	Running		t3.micro	2/2 checks passed	No alarms	+	eu-north-1c	ec2-13-49-148-112.eu-...
Db_staging	i-0a494fb20b0836448	Running		t3.micro	2/2 checks passed	No alarms	+	eu-north-1c	ec2-13-53-140-55.eu-n...
Db_dev	i-02503c9db223c3d8c	Running		t3.micro	2/2 checks passed	No alarms	+	eu-north-1c	ec2-13-53-186-138.eu-...
Db_production	i-071e93ca717f1aa8c	Running		t3.micro	2/2 checks passed	No alarms	+	eu-north-1c	ec2-13-53-120-222.eu-...
App_production	i-0eb8321d848ef3cdd	Running		t3.micro	2/2 checks passed	No alarms	+	eu-north-1c	ec2-13-49-148-38.eu-n...
App_dev	i-0ae5137d5797b517e	Running		t3.micro	2/2 checks passed	No alarms	+	eu-north-1c	ec2-13-53-188-141.eu-...

```
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

AWS_App_Server = "13.53.188.141"
AWS_Db_Server = "13.53.186.138"
Active_Environment = "dev"

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

AWS_App_Server = "13.49.148.38"
AWS_Db_Server = "13.53.120.222"
Active_Environment = "production"

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

AWS_App_Server = "13.49.148.112"
AWS_Db_Server = "13.53.140.55"
Active_Environment = "staging"
```

igeocitizen-terraform-state Info

Objects | Properties | Permissions | Metrics | Manag

Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3

Copy S3 URI | Copy URL | Download

Find objects by prefix

Name	Type
dev/	Folder
production/	Folder
staging/	Folder

Ivan Danyliuk

Prometheus, Node Exporter, Grafana

Prometheus

- open source (Apache 2.0) time series DBMS (Database Management System)
- written in Go
- originally developed by SoundCloud.

In other words, this thing stores metrics.

Node Exporter is a service

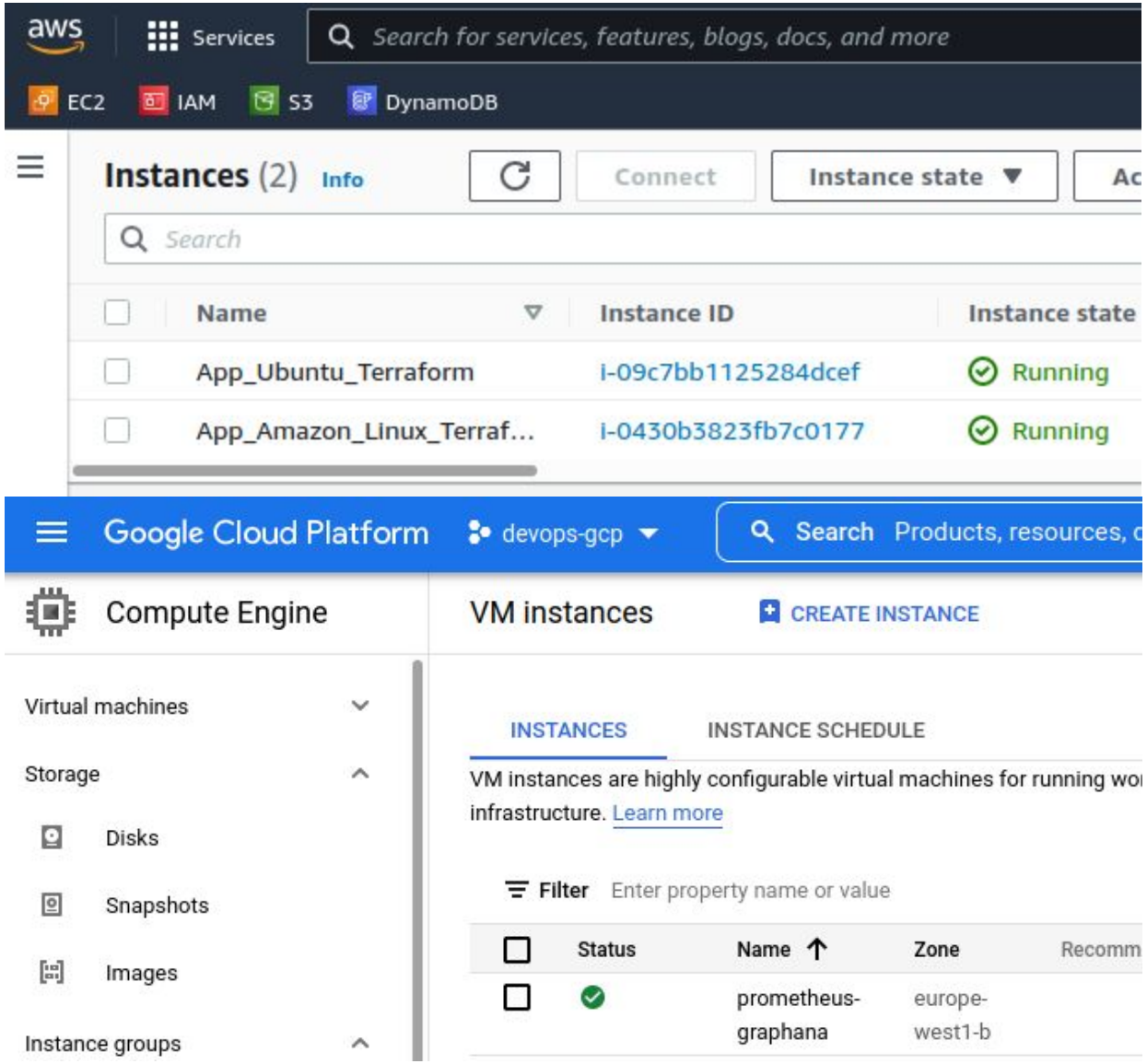
- task is to export information about a machine in a format understandable by Prometheus

Grafana

- Is open source (Apache 2.0) web frontend to various time series database engines
- Grafana draws beautiful graphs for you using information from Prometheus
- Prometheus developers themselves recommend using Grafana

Ivan Danyliuk

Prometheus, Node Exporter, Grafana



Infrastructure:

- Application Geocitizen VMS
  - 2 AWS EC2 Instances
    - Ubuntu 16.04 (App)
    - Amazon Linux (PostgreSQL)
- GCP Instance for Prometheus/Grafana

Ivan Danyliuk

1. Prometheus Node Exporter

After launch, go to the browser at the address http://16.171.35.100:9100/metrics

Mozilla Firefox

16.171.35.100:9100/metrics

16.171.35.100:9100/metrics

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 4.635e-05
go_gc_duration_seconds{quantile="0.25"} 7.992e-05
go_gc_duration_seconds{quantile="0.5"} 0.000176708
go_gc_duration_seconds{quantile="0.75"} 0.000658729
go_gc_duration_seconds{quantile="1"} 0.001330346
go_gc_duration_seconds_sum 0.003868513
go_gc_duration_seconds_count 11
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 8
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.849248e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 2.5492376e+07
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.454983e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 221046
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 4.668336247313072e-06
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
```

Ivan Danyliuk

2. Prometheus

Install Prometheus

Configure it to collect data from the previously installed Node Exporter

$sudo nano /etc/prometheus/prometheus.yml

```
    static_configs:
      - targets: ['35.195.74.140:9090']

  - job_name: node
    # If prometheus-node-exporter is installed, grab stats about the local
    # machine by default.
    static_configs:
      - targets: ['35.195.74.140:9100', '16.171.35.100:9100', '16.171.28.6:9100']
```

Prometheus collect metrics from two services:

- itself ('35.195.74.140:9090')
- Node Exporter instances ('35.195.74.140:9100', '16.171.35.100:9100', '16.171.28.6:9100')

Ivan Danyliuk

2. Prometheus

After launching Prometheus, you can open its user interface in the browser '35.195.74.140:9090'. The "/targets" page displays the status of the systems you are getting

35.195.74.140:9090/targets

Prometheus Alerts Graph Status Help

Targets

Only unhealthy jobs

node (3/3 up) show less

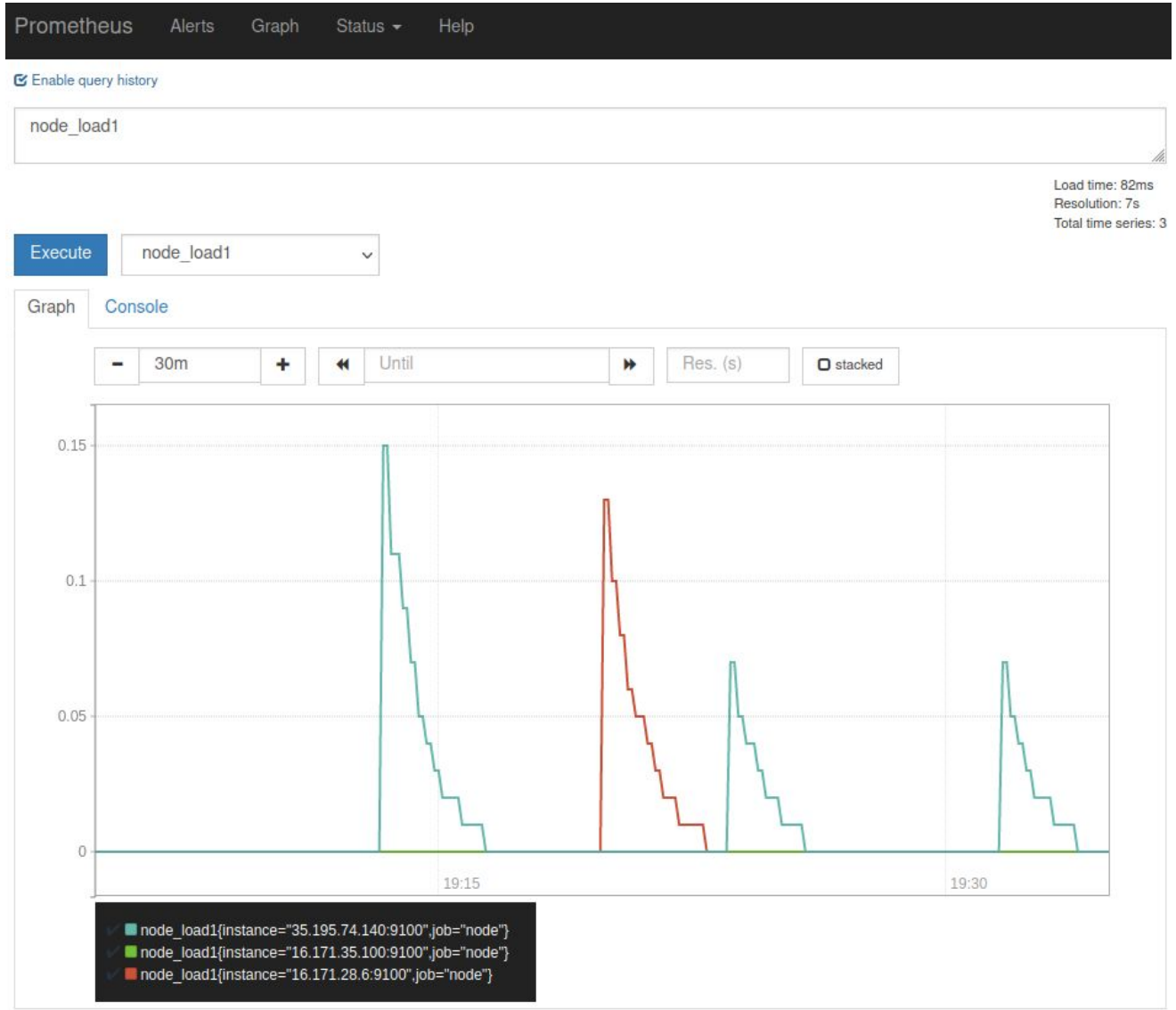
Endpoint	State	Labels	Last Scrape	Error
http://16.171.28.6:9100/metrics	UP	instance="16.171.28.6:9100"	12.935s ago	
http://16.171.35.100:9100/metrics	UP	instance="16.171.35.100:9100"	7.504s ago	
http://35.195.74.140:9100/metrics	UP	instance="35.195.74.140:9100"	12.212s ago	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Error
http://35.195.74.140:9090/metrics	UP	instance="35.195.74.140:9090"	573ms ago	

Ivan Danyliuk

2. Prometheus



Already using only Prometheus, you can query the collected data and build graphs using the tools on the "/graph" page.

It's a good tool for learning the metrics you're collecting and writing complex data queries.

But we will entrust this work to Grafana.

Ivan Danyliuk

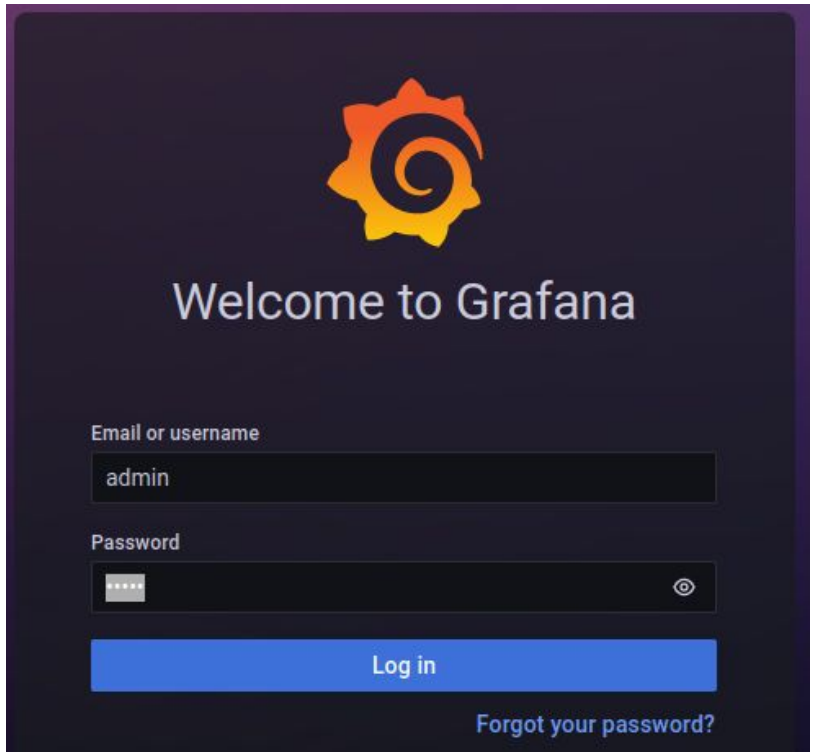
3. Grafana

Grafana is the last component of my solution.

Its task is to connect to Prometheus and display the collected metrics on charts and dashboards. Grafana only interacts with Prometheus, all collected metrics are stored there.

Web interface:

http://35.195.74.140:3000



The default user and password is

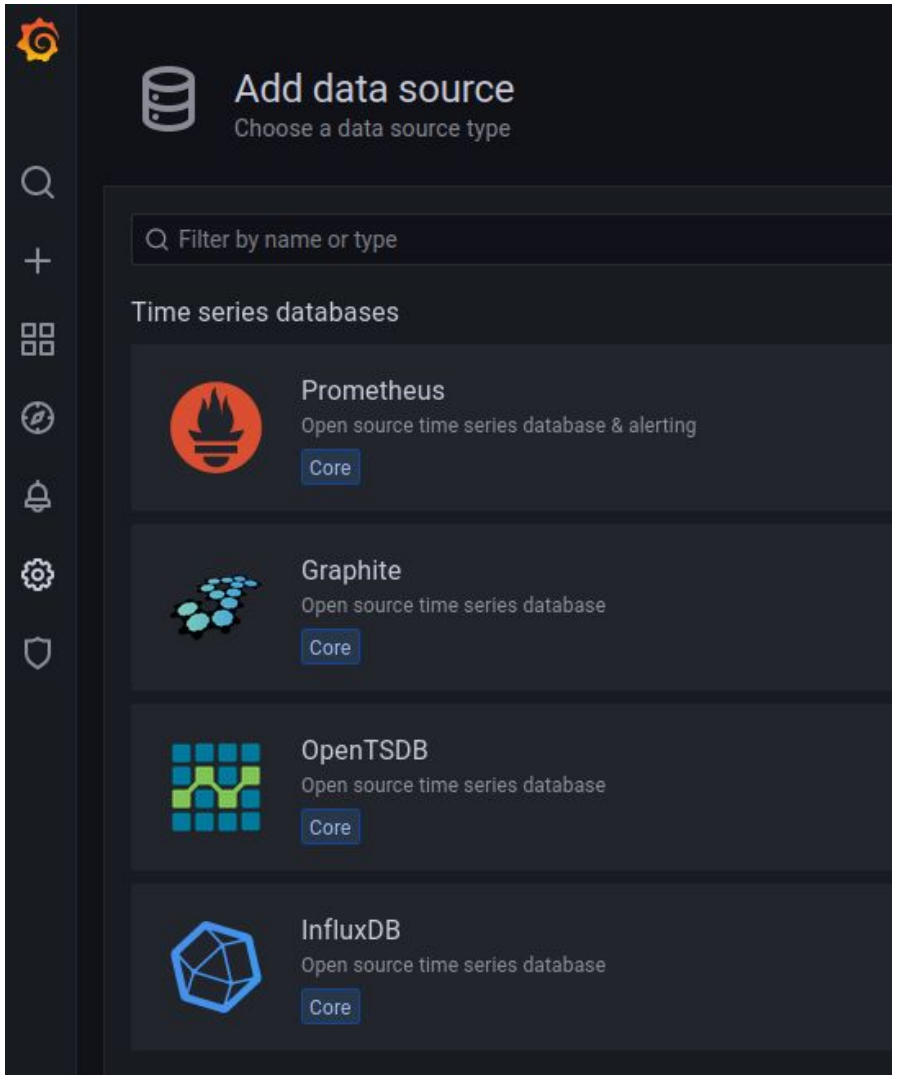
"admin"/"admin".

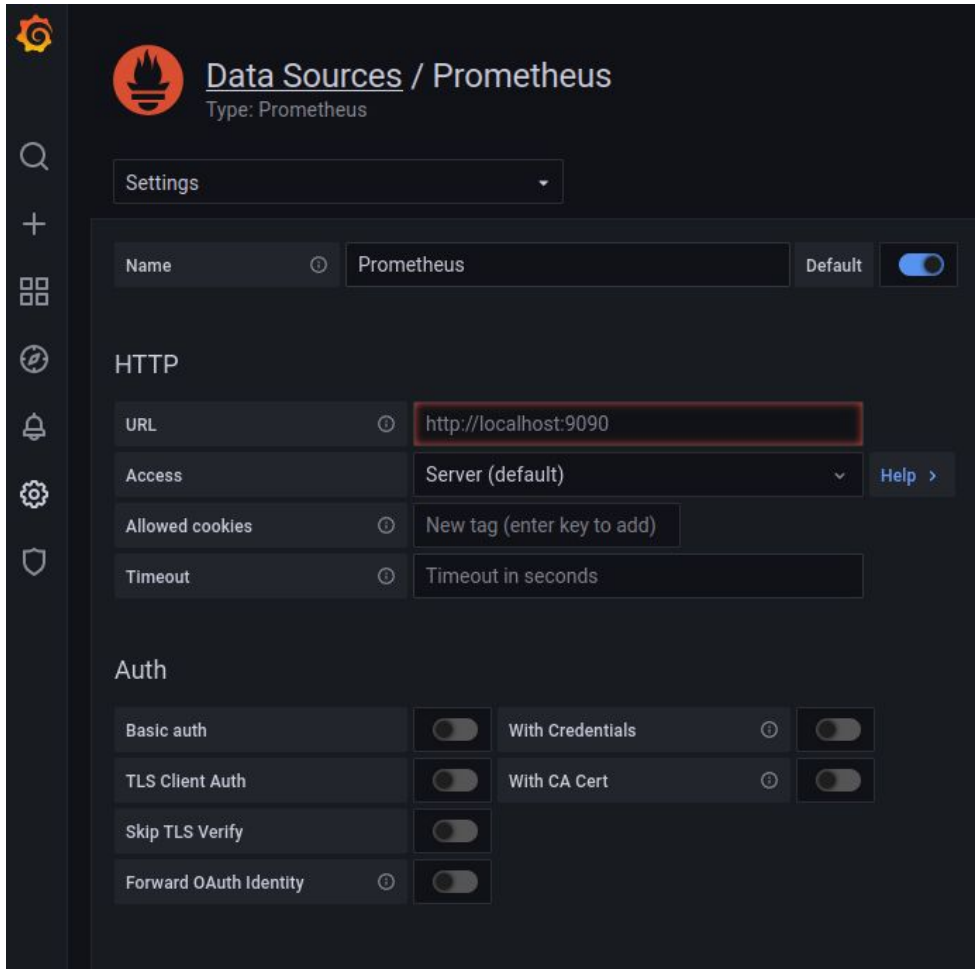
The password will need to be changed upon first login.

Ivan Danyliuk

Grafana configuration

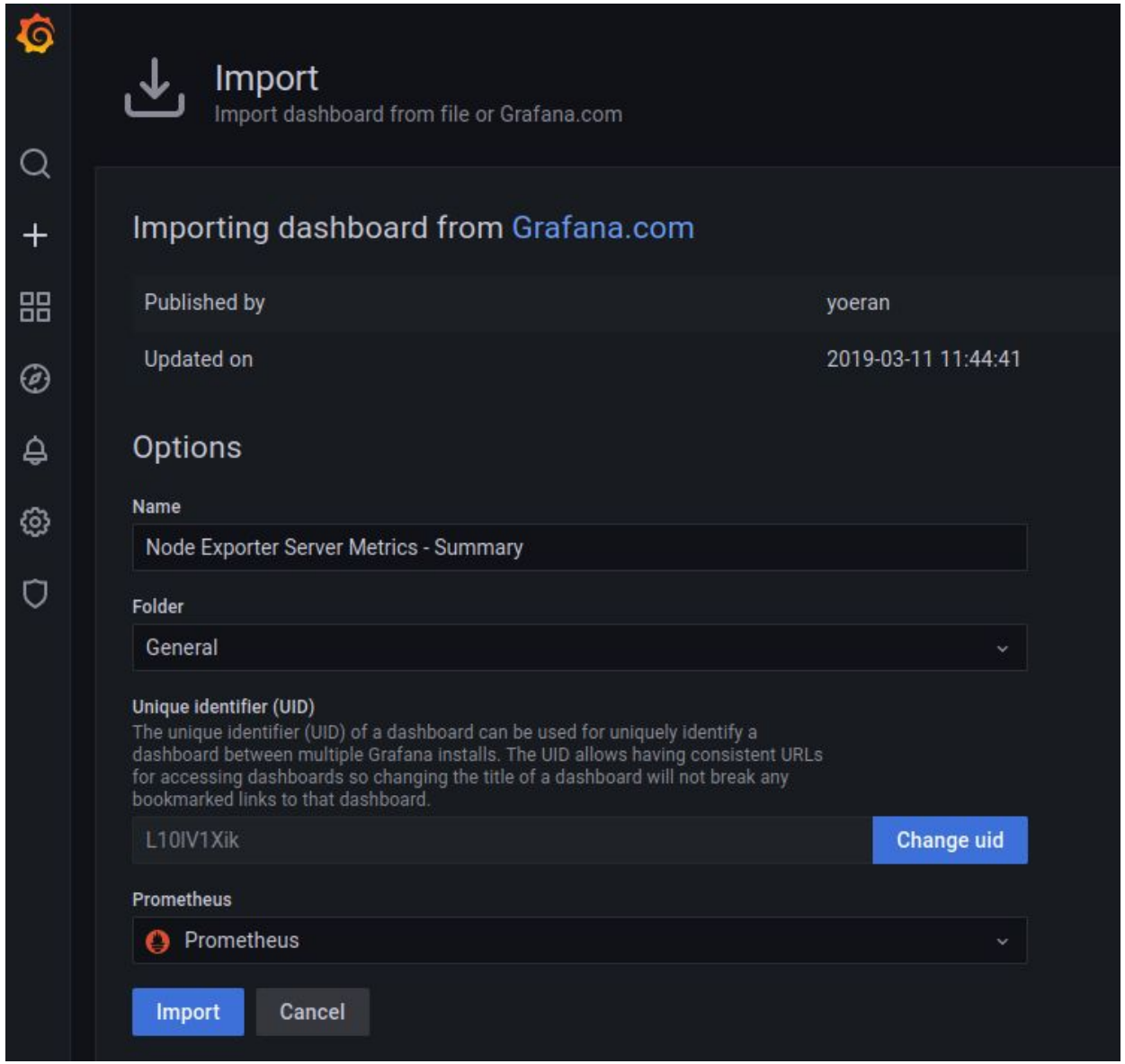
The first step is to set up a data source.





Ivan Danyliuk

Grafana configuration



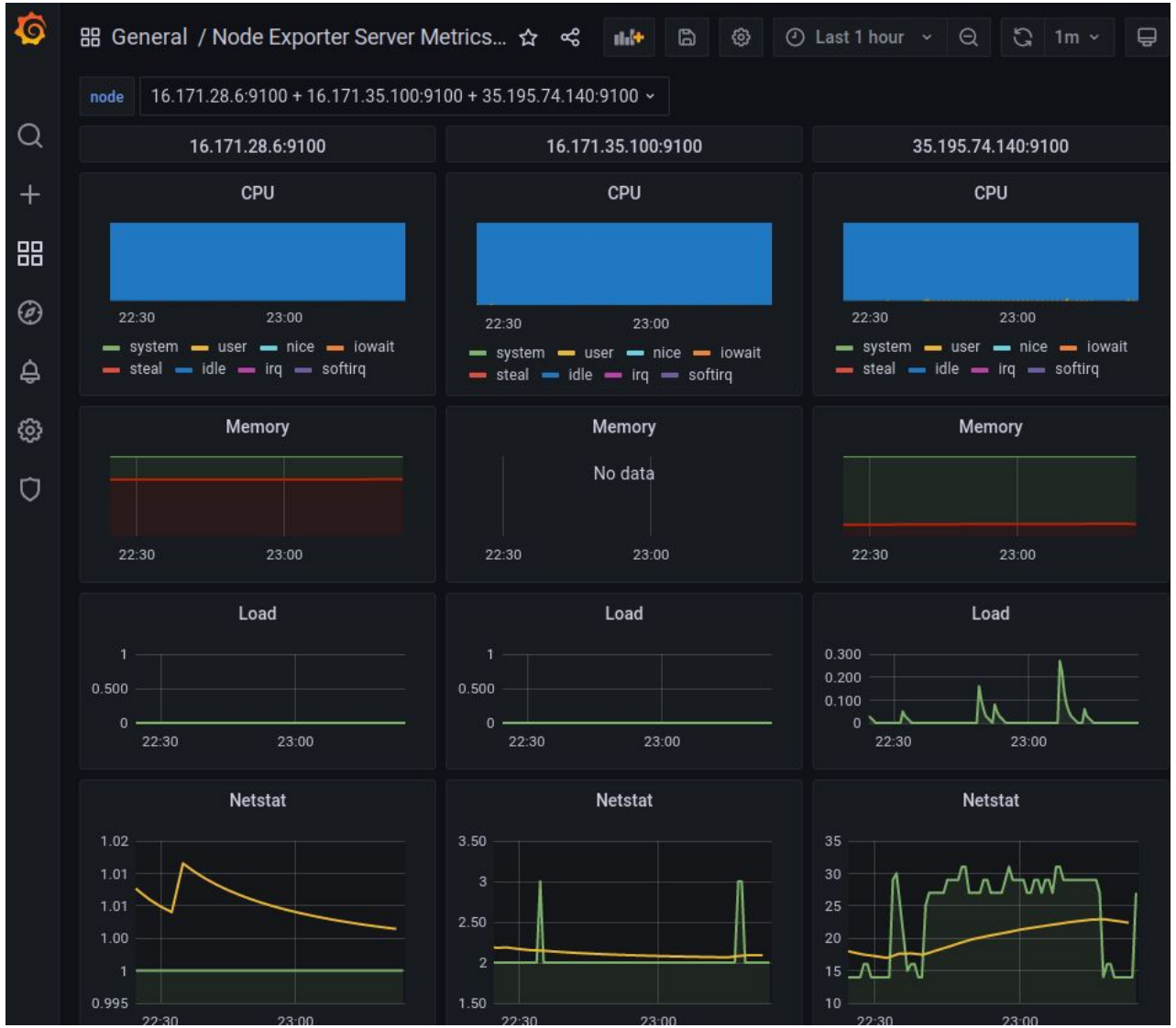
After setting up the Datasource, we can create a dashboard.

As an example, I use dashboard located at the following link:

https://grafana.com/grafana/dashboards/9901

Ivan Danyliuk

Grafana dashboard



Ivan Danyliuk

Ivan Danyliuk