Managing the Selenoid cluster with the help of Terraform

Leonid Rudenko

About me

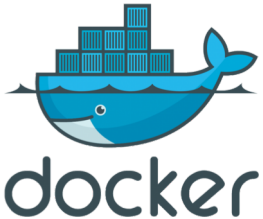
5.5 years @ Yandex

3 years @ JetBrains

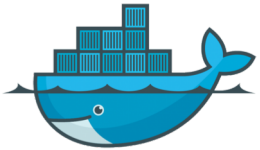
♥ Kotlin

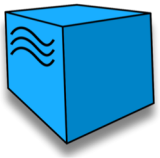
What’s going on here

What’s going on here



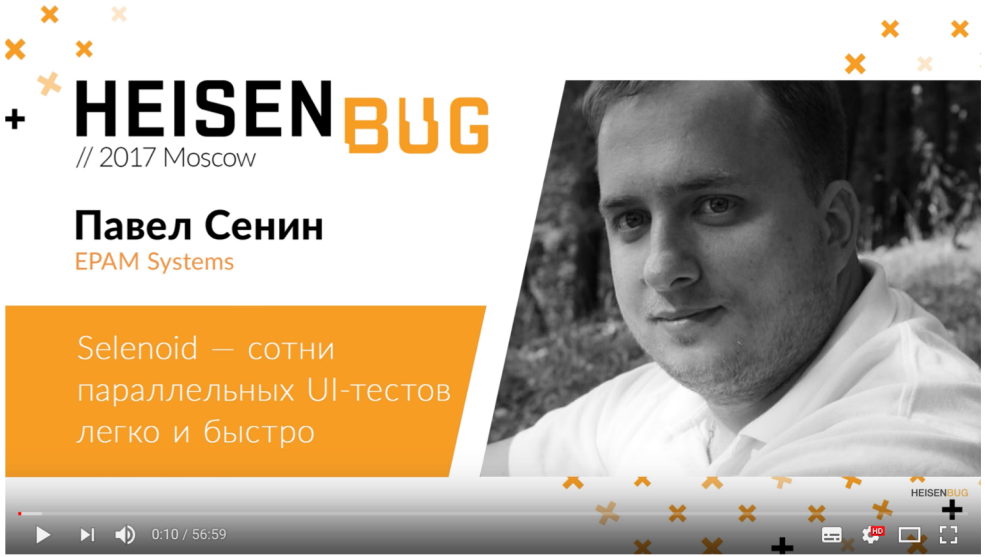
What’s going on here





Selenoid

https://youtu.be/YqcQ5Mg44DU



Павел Сенин — Selenoid — сотни параллельных UI-тестов легко и быстро

Selenoid

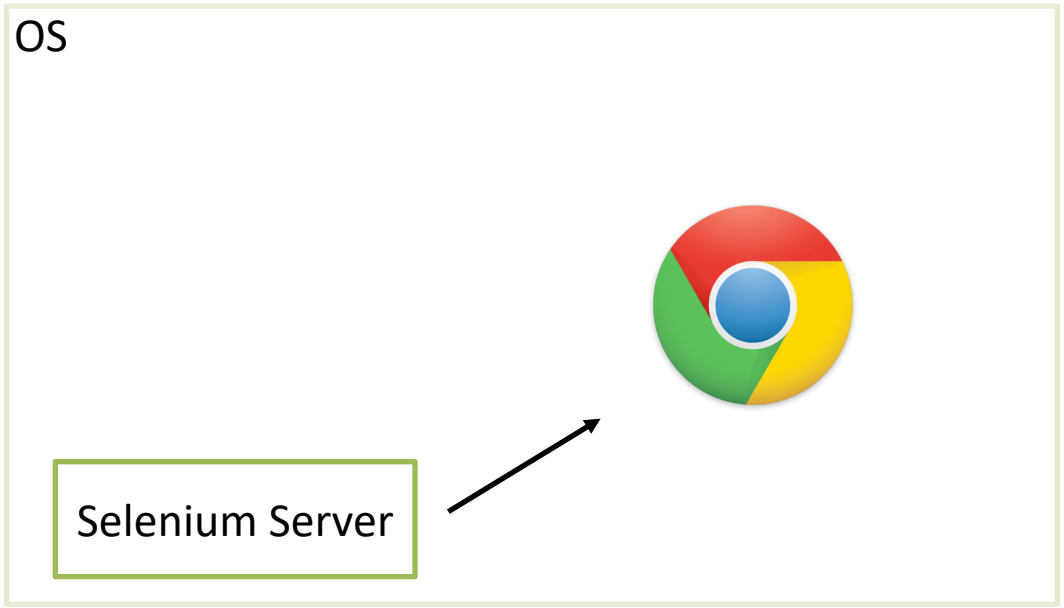
https://youtu.be/YqcQ5Mg44DU

https://youtu.be/wAKcBinMn6o

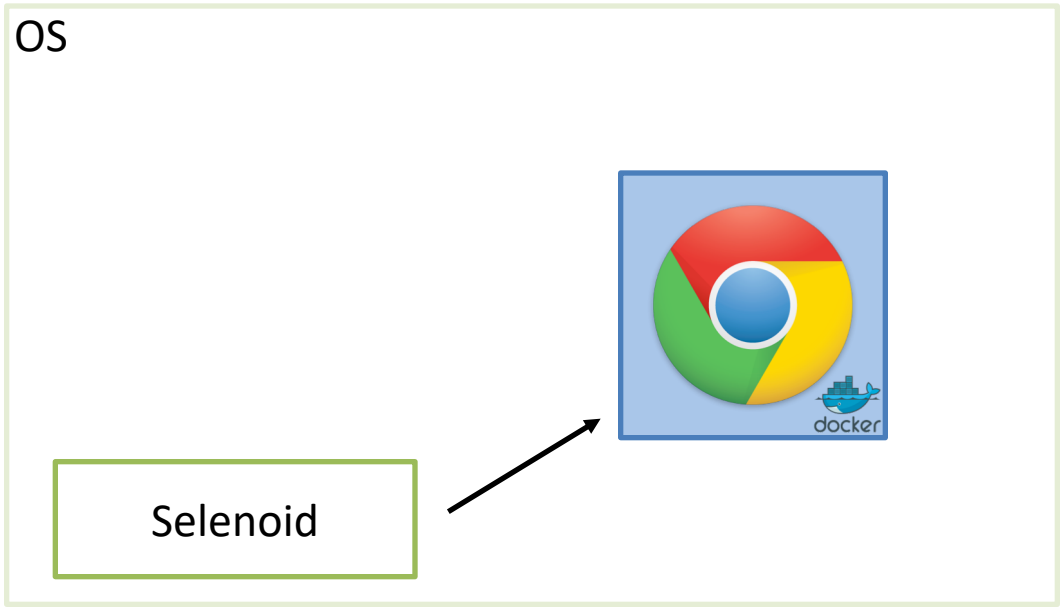
https://youtu.be/4ZHQheFc4-8

https://youtu.be/G-TrW9SZRNg

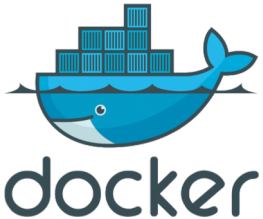
Selenoid

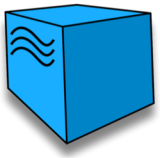


Selenoid



What’s going on here





What’s going on here

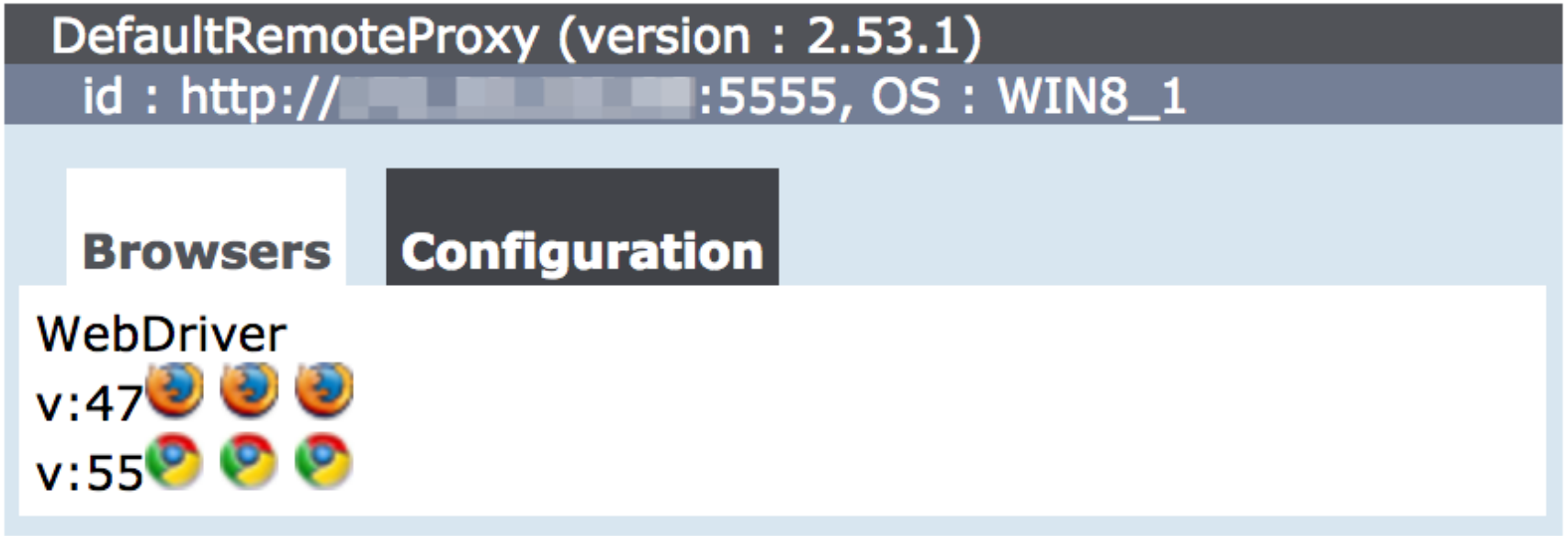
1. Motivation
2. Terraform 101
3. Tips & Tricks for Selenoid cluster
4. Cluster monitoring

What’s going on here

1. Motivation
2. Terraform 101
3. Tips & Tricks for Selenoid cluster
4. Cluster monitoring

A long time ago in a galaxy far, far away....

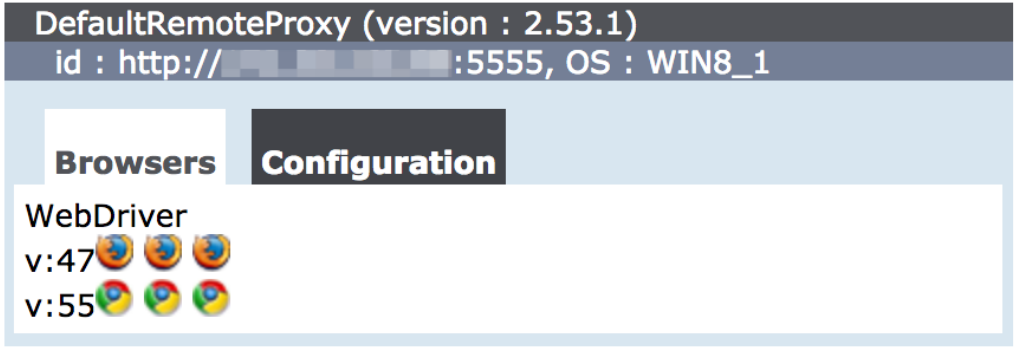
Traditional Selenium Grid



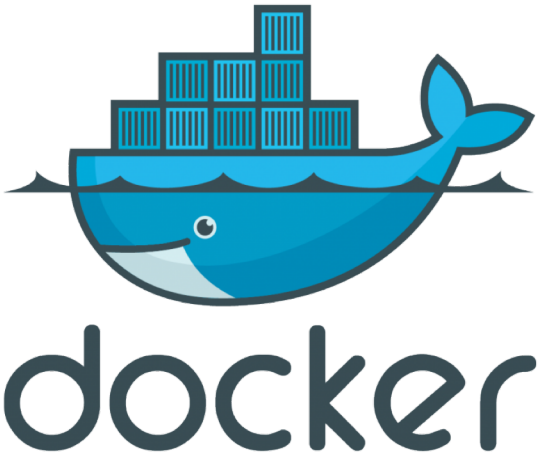
Traditional Selenium Grid

Browser isolation

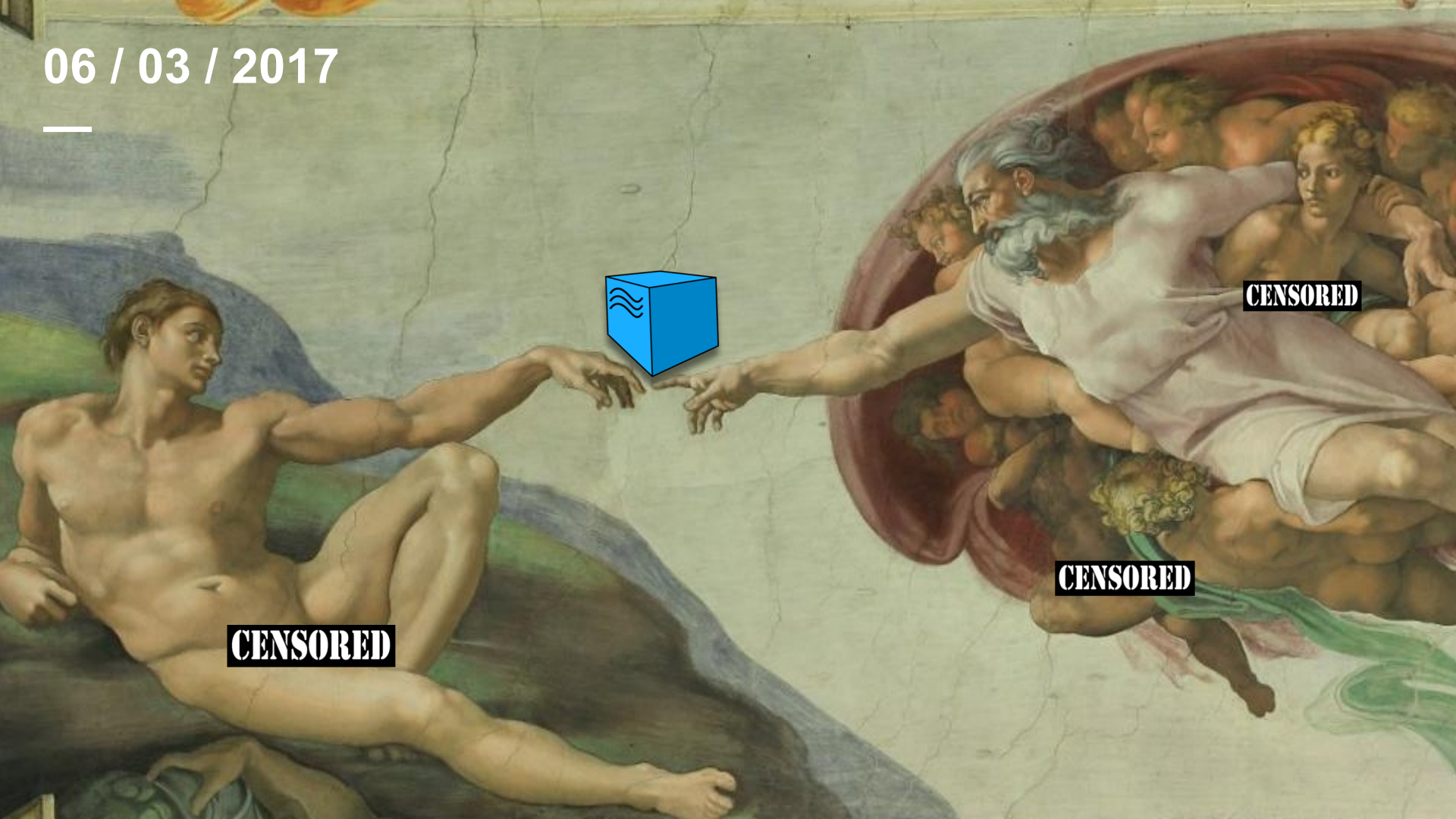
– chromedriver temp files



Docker All The Things



06 / 03 / 2017



Selenoid Quick Start


```
docker pull selenoid/firefox:latest
docker pull selenoid/chrome:latest
docker pull aandryashin/selenoid:1.0.0
```

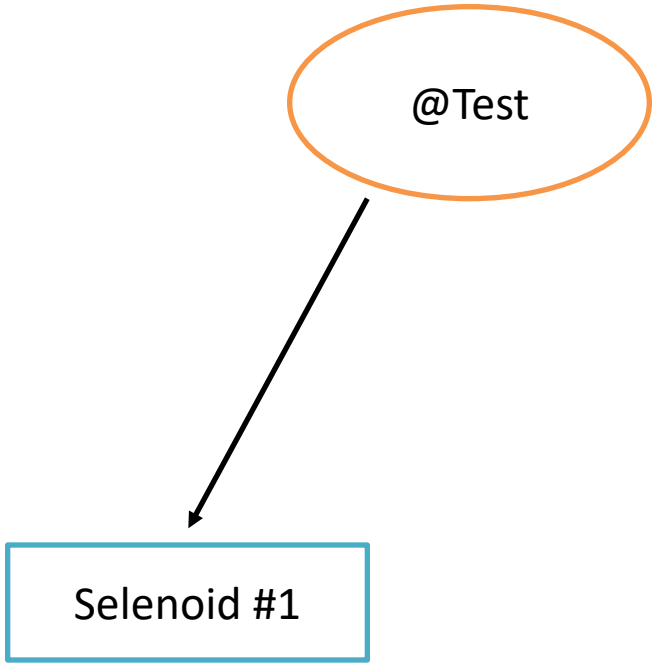
Selenoid Quick Start


```
docker pull selenoid/firefox:latest
docker pull selenoid/chrome:latest
docker pull aandryashin/selenoid:1.0.0

nano /etc/selenoid/browsers.json
```

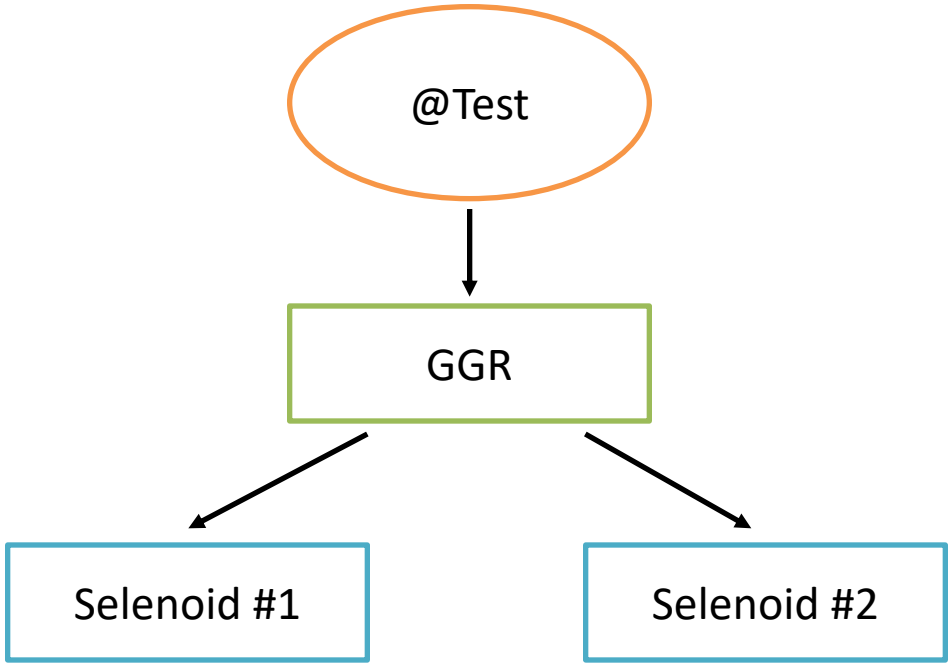
Selenoid Quick Start


```
docker pull selenoid/firefox:latest
docker pull selenoid/chrome:latest
docker pull aandryashin/selenoid:1.0.0

nano /etc/selenoid/browsers.json

docker run -d -p 4444:4444 \
    -v /etc/selenoid:/etc/selenoid:ro \
    -v /var/run/docker.sock:/var/run/docker.sock \
    aandryashin/selenoid:1.0.0
```

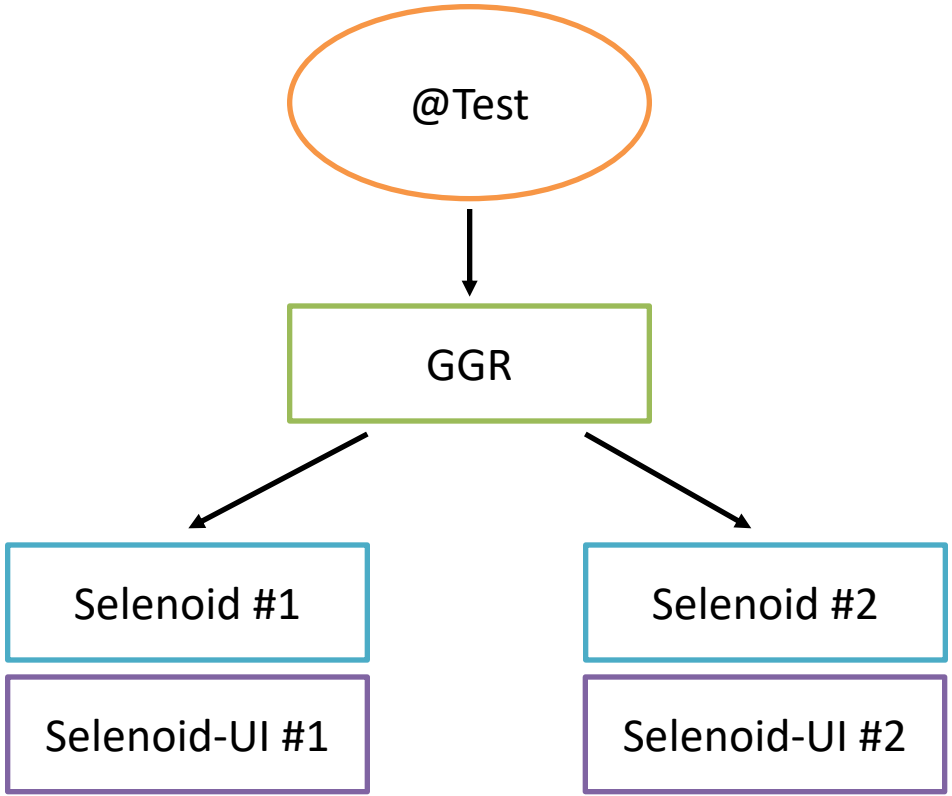
Why Terraform?



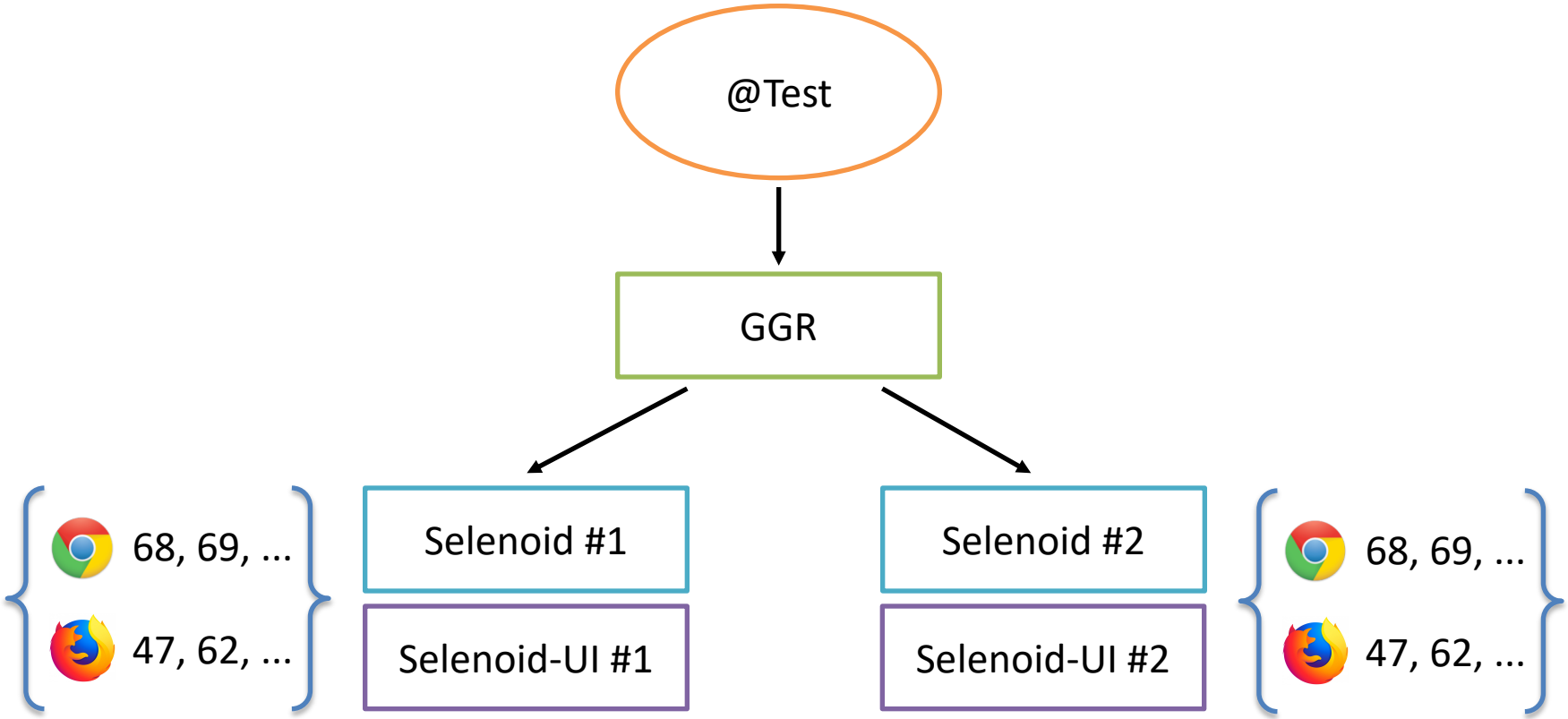
Why Terraform?



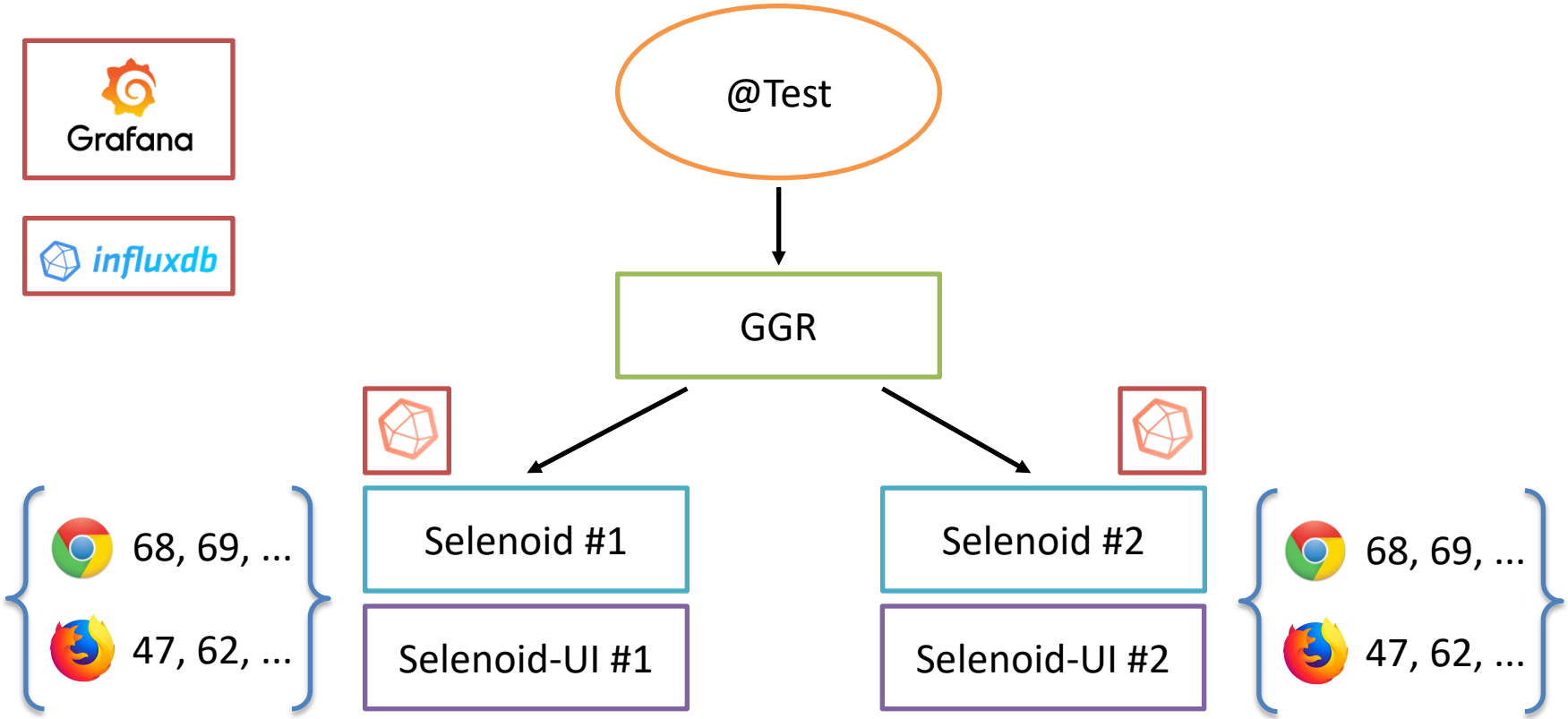
Why Terraform?



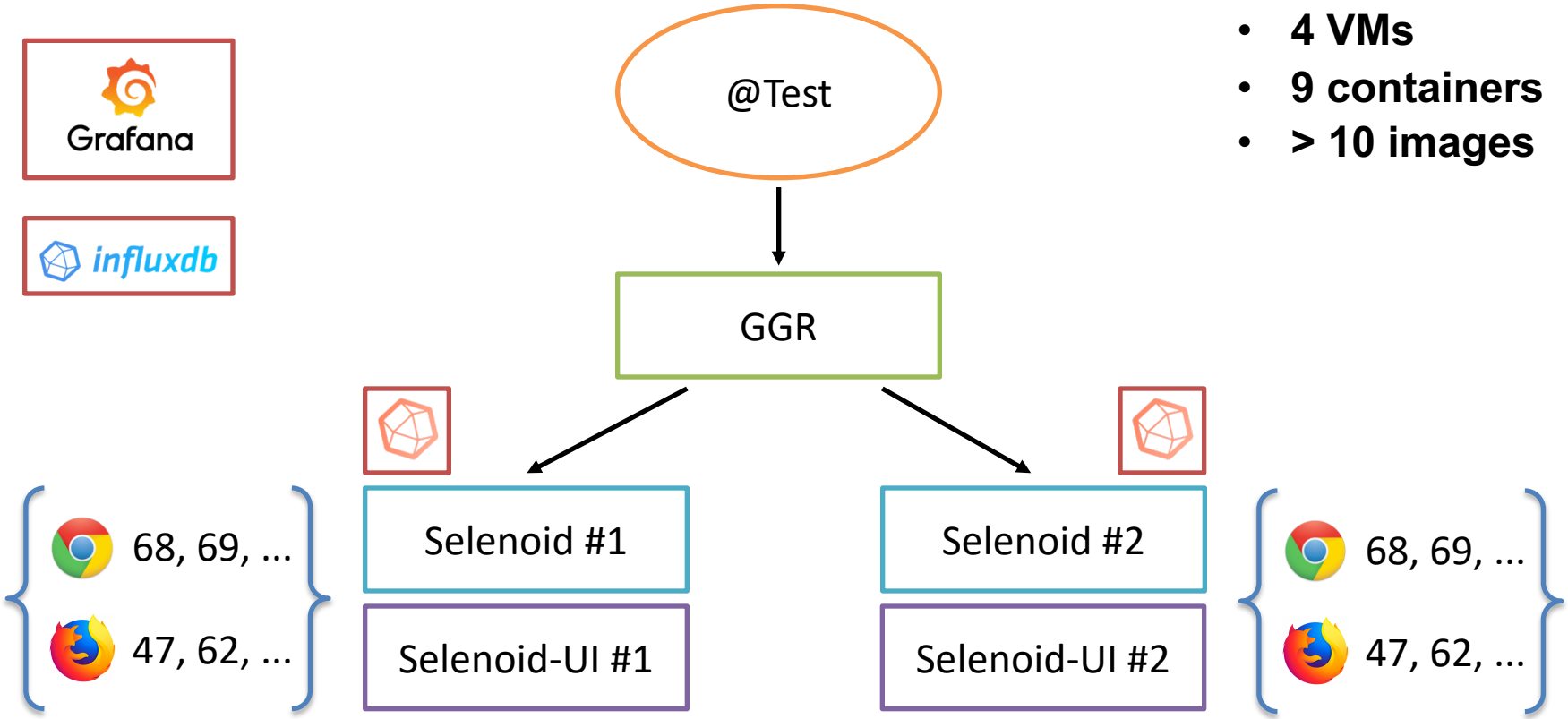
Why Terraform?



Why Terraform?



Why Terraform?



- **4 VMs**
- **9 containers**
- **> 10 images**

Terraform

HashiCorp

Terraform

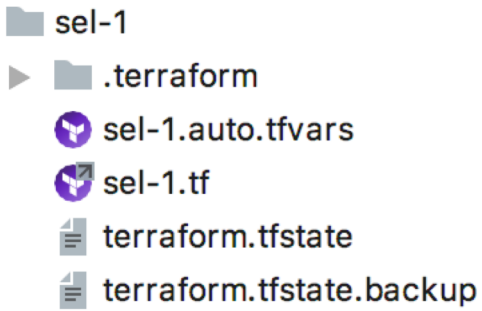
Write, Plan, and Create Infrastructure as Code

What’s going on here

1. Motivation
2. Terraform 101
3. Tips & Tricks for Selenoid cluster
4. Cluster monitoring

Terminology

- **Configuration**



Terminology

- Configuration
- **Provider**

```
provider "docker" {
  host = "tcp://virtual-machine.company.net:2375"
}
```

Terminology

- Configuration
- **Provider**

```
provider "vsphere" {
  user                 = "admin"
  password             = "pa$$w0rd"
  vsphere_server       = "vcenter-srv.company.net"
  allow_unverified_ssl = true
}
```

Terminology

- Configuration
- Provider
- **Resource**

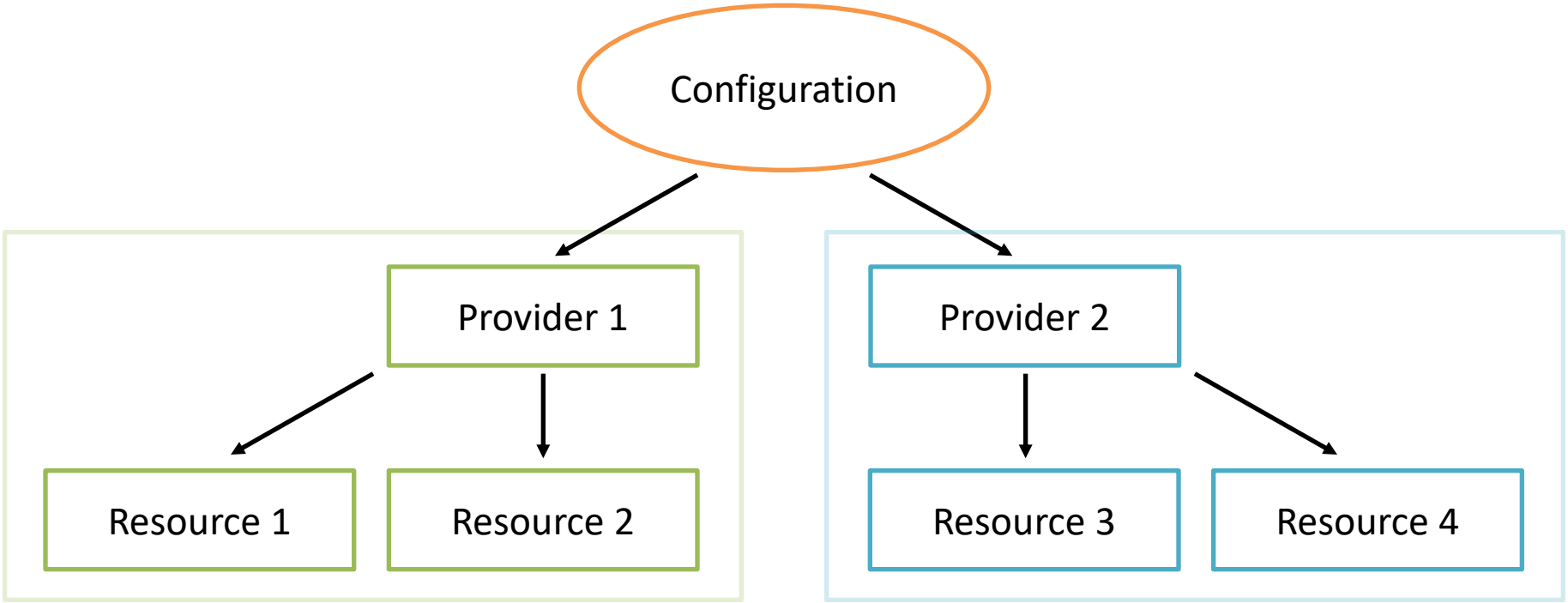
```
resource "docker_container" "selenoid" {
  image   = "aerokube/selenoid:1.7.2"
  command = ["-limit", "8", "-timeout", "2m0s"]
  ...
}
```

Terminology

- Configuration
- Provider
- **Resource**


```
resource "docker_container" "selenoid" {
  image   = "aerokube/selenoid:1.7.2"
  command = ["-limit", "8", "-timeout", "2m0s"]
  ...
}
```

Terminology



Installation

- https://www.terraform.io/downloads.html

```
$ terraform version
Terraform v0.11.10

$
```

Task #1

- **Selenoid is running on http://localhost:4444 (Linux and Mac)**

```
$ docker images | grep selenoid
$
$ docker ps –a
$
```

- **https://git.io/tf-task1**

Provider


```
provider "docker" {
  host = "unix:///var/run/docker.sock"
  // host = "tcp://virtual-machine.company.net:2375"
}
```

Resource: image


```
resource "docker_image" "selenoid" {
  name = "aerokube/selenoid:1.8.2"
}
```

Resource: container


```
resource "docker_container" "selenoid" {
  name    = "selenoid"
  image   = "${docker_image.selenoid.latest}"
  command = ["-limit", "4", "-timeout", "2m0s"]
  ports {
    internal = 4444
    external = 4444
  }
  volumes {
    host_path      = "/var/run/docker.sock"
    container_path = "/var/run/docker.sock"
  }
  upload {
    content = "{}"
    file    = "/etc/selenoid/browsers.json"
  }
}
```

Resource: container


```
resource "docker_container" "selenoid" {
  name    = "selenoid"
  image   = "${docker_image.selenoid.latest}"
  command = ["-limit", "4", "-timeout", "2m0s"]
  ports {
    internal = 4444
    external = 4444
  }
  volumes {
    host_path      = "/var/run/docker.sock"
    container_path = "/var/run/docker.sock"
  }
  upload {
    content = "{}"
    file    = "/etc/selenoid/browsers.json"
  }
}
```

Resource: container


```
resource "docker_container" "selenoid" {
  name    = "selenoid"
  image   = "${docker_image.selenoid.latest}"
  command = ["-limit", "4", "-timeout", "2m0s"]
  ports {
    internal = 4444
    external = 4444
  }
  volumes {
    host_path      = "/var/run/docker.sock"
    container_path = "/var/run/docker.sock"
  }
  upload {
    content = "{}"
    file    = "/etc/selenoid/browsers.json"
  }
}
```

Resource: container


```
resource "docker_container" "selenoid" {
  name    = "selenoid"
  image   = "${docker_image.selenoid.latest}"
  command = ["-limit", "4", "-timeout", "2m0s"]
  ports {
    internal = 4444
    external = 4444
  }
  volumes {
    host_path      = "/var/run/docker.sock"
    container_path = "/var/run/docker.sock"
  }
  upload {
    content = "{}"
    file    = "/etc/selenoid/browsers.json"
  }
}
```

Resource: container


```
resource "docker_container" "selenoid" {
  name    = "selenoid"
  image   = "${docker_image.selenoid.latest}"
  command = ["-limit", "4", "-timeout", "2m0s"]
  ports {
    internal = 4444
    external = 4444
  }
  volumes {
    host_path      = "/var/run/docker.sock"
    container_path = "/var/run/docker.sock"
  }
  upload {
    content = "{}"
    file    = "/etc/selenoid/browsers.json"
  }
}
```

Terraform init


```
$ terraform init
```

Terraform init


```
$ terraform init

Initializing provider plugins...
- Checking for available provider plugins on
https://releases.hashicorp.com...
- Downloading plugin for provider "docker" (1.0.1)...

Terraform has been successfully initialized!
$
```

Terraform apply


```
$ terraform apply
```

Terraform apply


```
$ terraform apply
...

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only ‘yes’ will be accepted to approve.

  Enter a value:
```

Terraform apply


```
$ terraform apply

docker_image.selenoid: Creating...
docker_image.selenoid: Creation complete after 4s
docker_container.selenoid: Creating...
docker_container.selenoid: Creation complete after 1s

Apply complete! Resources: 2 added, 0 changed, 0
destroyed.
$
```

Is Selenoid running?

- http://localhost:4444/status

Terraform state file


```
$ ls
selenoid.tf       terraform.tfstate
$
```

Terraform destroy


```
$ terraform destroy
```

Terraform destroy


```
$ terraform destroy

docker_container.selenoid: Destroying...
docker_container.selenoid: Destruction complete after 1s
docker_image.selenoid: Destroying...
docker_image.selenoid: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.

$
```

Is Selenoid running?

- http://localhost:4444/status

Task #2

- **Selenoid runs containers with Chrome and Firefox**

- https://git.io/tf-task2

Variable


```
variable "browsers" {
  type = "list"
  default = [
    "selenoid/chrome:69.0",
    "selenoid/firefox:62.0"
  ]
}
```

Variable usage


```
resource "docker_image" "browser_images" {
  count        = "${length(var.browsers)}"
  name         = "${element(var.browsers, count.index)}"
  keep_locally = true
}
```

Variable usage


```
resource "docker_image" "browser_images" {
  count        = "${length(var.browsers)}"
  name         = "${element(var.browsers, count.index)}"
  keep_locally = true
}
```

Variable usage


```
resource "docker_image" "browser_images" {
  count         = "${length(var.browsers)}"
  name          = "${element(var.browsers, count.index)}"
  keep_locally = true
}
```

Variable usage


```
resource "docker_image" "browser_images" {
  count        = "${length(var.browsers)}"
  name         = "${element(var.browsers, count.index)}"
  keep_locally = true
}
```

browsers.json


```
{
  "chrome": {
    "default": "69.0",
    "versions": {
      "69.0": {
        "image": "selenoid/chrome:69.0",
        "port": "4444"
      }
    }
  },
  "firefox": {
    "default": "62.0",
    "versions": {
      "62.0": {
        "image": "selenoid/firefox:62.0",
        "path": "/wd/hub",
        "port": "4444"
      }
    }
  }
}
```

Upload browsers.json


```
upload {
  content = "${file("browsers.json")}"
  file    = "/etc/selenoid/browsers.json"
}
```

Terraform apply


```
$ terraform apply
```

Is Selenoid running?

- http://localhost:4444/status

Task #3

- **Add Opera browser with no Selenoid downtime**

- **https://git.io/tf-task3**

Terraform apply


```
$ terraform apply

Terraform will perform the following actions:

-/+ docker_container.selenoid (new resource required)

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

$
```

Provisioner

Resource #1: Selenoid container with {} config

Resource #2: file browsers.json + ???

Mount config directory


```
upload {
  content = "{}"
  file    = "/etc/selenoid/browsers.json"
}

volumes {
  host_path      = "/Users/leonid.rudenko/selenoid/"
  container_path = "/etc/selenoid/"
}
```

Mount config directory


```
upload {
  content = "{}"
  file    = "/etc/selenoid/browsers.json"
}

volumes {
  host_path      = "/Users/leonid.rudenko/selenoid/"
  container_path = "/etc/selenoid/"
}
```

Mount config directory


```
upload {
  content = "{}"
  file    = "/etc/selenoid/browsers.json"
}
```

```
volumes {
  host_path      = "/Users/leonid.rudenko/selenoid/"
  container_path = "/etc/selenoid/"
}
```

Local file resource and provisioner


```
resource "local_file" "browsers_json" {
  content  = "${file("browsers.json")}"
  filename = "/Users/leonid.rudenko/selenoid/"

  provisioner "local_exec" {
    command =
      "docker kill -s HUP ${docker_container.selenoid.id}"
  }
}
```

Local file resource and provisioner


```
resource "local_file" "browsers_json" {
  content  = "${file("browsers.json")}"
  filename = "/Users/leonid.rudenko/selenoid/"

  provisioner "local_exec" {
    command =
      "docker kill -s HUP ${docker_container.selenoid.id}"
  }
}
```

Local file resource and provisioner


```
resource "local_file" "browsers_json" {
  content  = "${file("browsers.json")}"
  filename = "/Users/leonid.rudenko/selenoid/"

  provisioner "local_exec" {
    command =
      "docker kill -s HUP ${docker_container.selenoid.id}"
  }
}
```

Local file resource and provisioner


```
resource "local_file" "browsers_json" {
  content  = "${file("browsers.json")}"
  filename = "/Users/leonid.rudenko/selenoid/"

  provisioner "local_exec" {
    command =
      "docker kill -s HUP ${docker_container.selenoid.id}"
  }
}
```

Is Selenoid running?

- http://localhost:4444/status

```
total:        4
used:         0
queued:       0
pending:      0
▼ browsers:
  ▼ chrome:
      69.0:   {}
  ▼ firefox:
      62.0:   {}
```

Terraform apply


```
$ terraform apply

Terraform will perform the following actions:
  + docker_image.browser_images[2]
-/+ local_file.browsers_json (new resource required)

Plan: 2 to add, 0 to change, 1 to destroy.

$
```

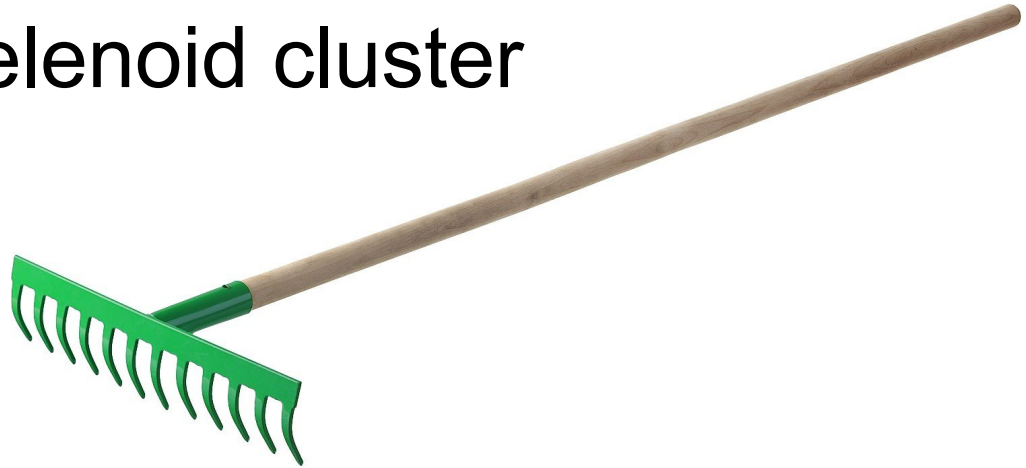
Is Selenoid running?

- http://localhost:4444/status

```
total:      4
used:       0
queued:     0
pending:    0
▼ browsers:
  ▼ chrome:
      69.0:   {}
  ▼ firefox:
      62.0:   {}
  ▼ opera:
      55.0:   {}
```

What’s going on here

1. Motivation
2. Terraform 101
3. Tips & Tricks for Selenoid cluster
4. Cluster monitoring



1. Container parameters


```
resource "docker_container" "selenoid" {
  ...

  restart = "always"

  ...
}
```

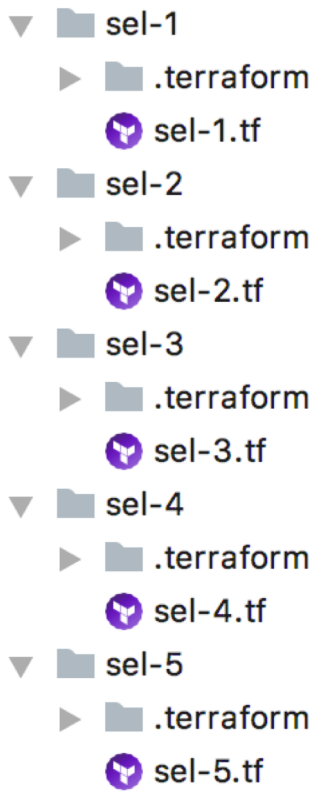
1. Container parameters


```
resource "docker_container" "selenoid" {
  ...

  log_opts = {
    max-size = "50m"
  }

  ...
}
```

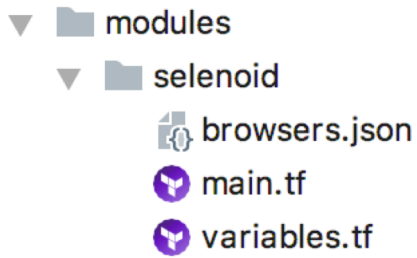
1. Container parameters


```
resource "docker_container" "selenoid" {
  ...

  env = [
    "TZ=Europe/Moscow"
  ]

  ...
}
```

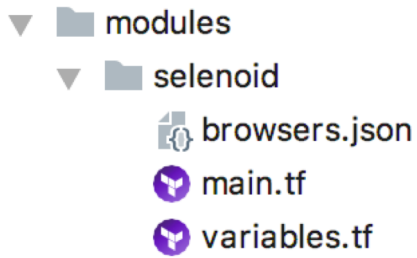
2. Multiple Selenoids



2. Multiple Selenoids: create module

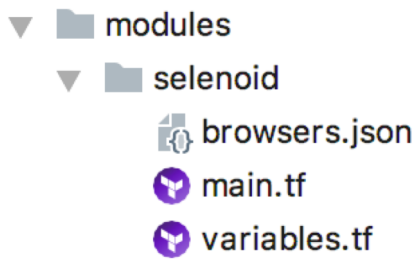


2. Multiple Selenoids: create module



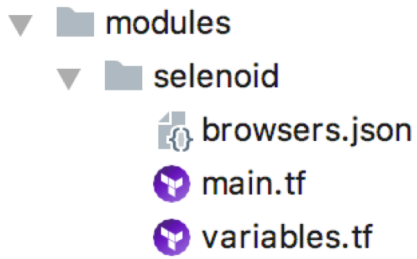
- **variables.tf**

2. Multiple Selenoids: create module



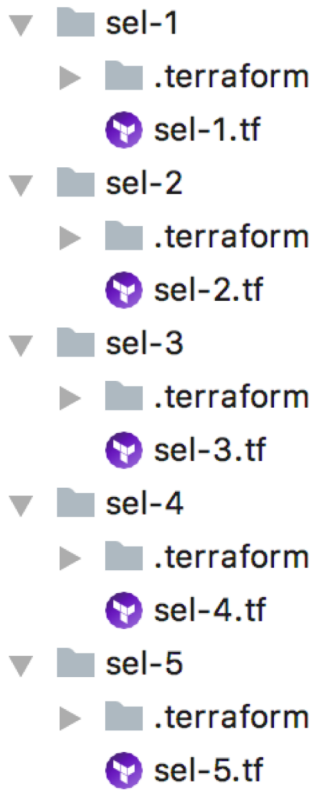
- variables.tf
- **main.tf**

2. Multiple Selenoids: create module



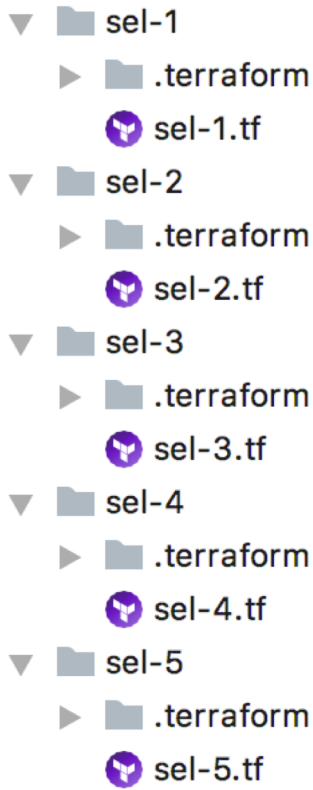
- variables.tf
- main.tf
- **browsers.json** `"${file("${path.module}/browsers.json")}"`

2. Multiple Selenoids: use module




```
module "selenoid" {
  source = "../modules/selenoid"
  hostname = "sel-1.***.net"
  selenoid_container_name = "sel1"
}
```

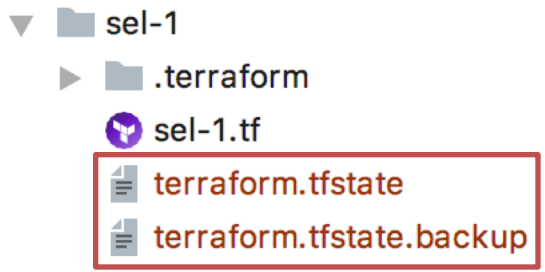
2. Multiple Selenoids: use module



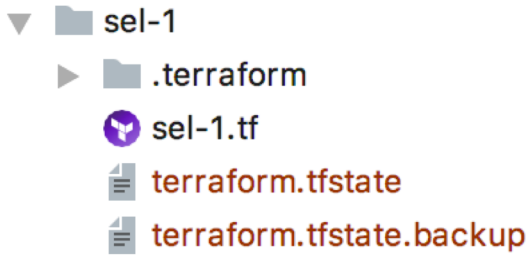

```
module "selenoid" {
  source = "../modules/selenoid"
  hostname = "sel-1.***.net"
  selenoid_container_name = "sel1"
}
```

```
$ terraform init
- module.selenoid
  Getting source "../modules/selenoid"
```

3. State files

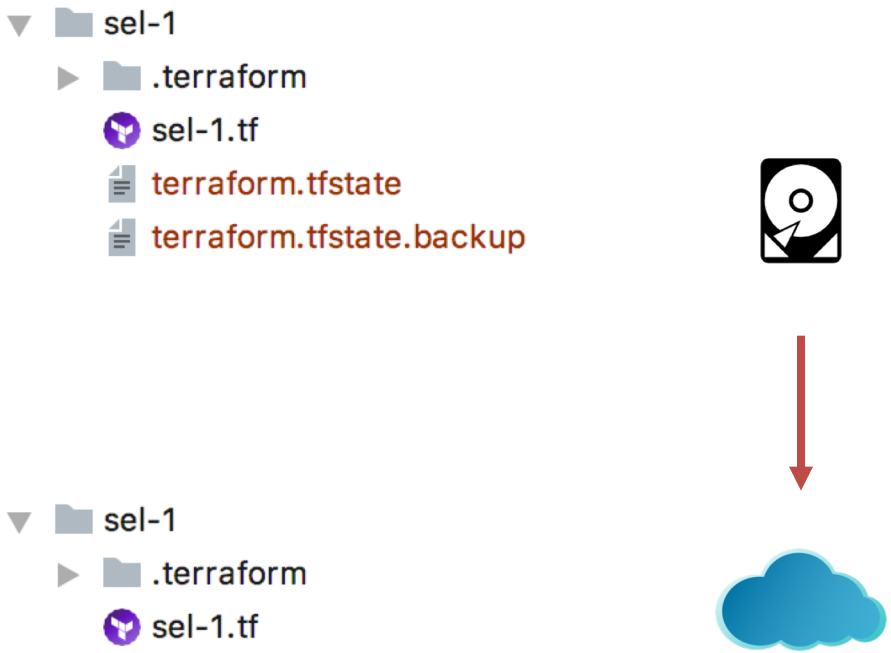


3. State files: local backend

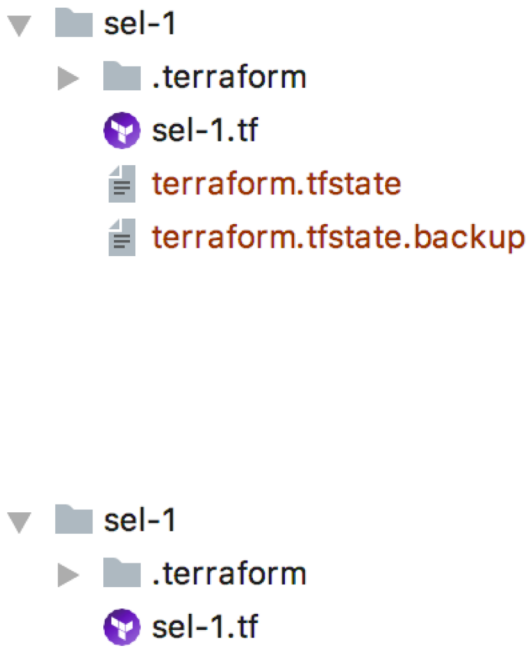


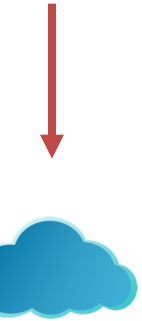
```
env.1 = PASSWORD=qwerty
```

3. State files: remote backend



3. State files: remote backend





3. State files: Consul backend

- https://www.consul.io

```
$ export CONSUL_HTTP_TOKEN=392bb2f7-44a4-9c4a-74e0-9bdaaf097e6f
```

3. State files: Consul backend


```
terraform {
  backend "consul" {
    address = "consul.***.net:8501"
    scheme  = "https"
    ca_file = "../crt/cert.pem"
    lock    = true
    path    = "projects/selenoid/sel-1.tfstate"
  }
}
```

3. State files: Consul backend


```
terraform {
  backend "consul" {
    address = "consul.***.net:8501"
    scheme  = "https"
    ca_file = "../crt/cert.pem"
    lock    = true
    path    = "projects/selenoid/sel-1.tfstate"
  }
}
```

3. State files: Consul backend


```
$ terraform init

Initializing the backend...


Successfully configured the backend "consul"! Terraform
will automatically use this backend unless the backend
configuration changes.
```

3. Remote backend: advantages

- **No sensitive data on disk/git**

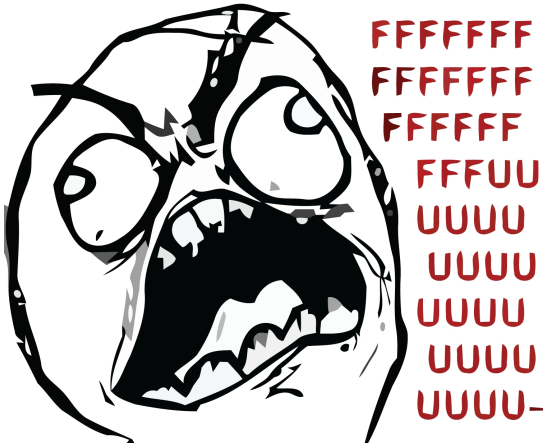
3. Remote backend: advantages

- No sensitive data on disk/git
- **State locking**

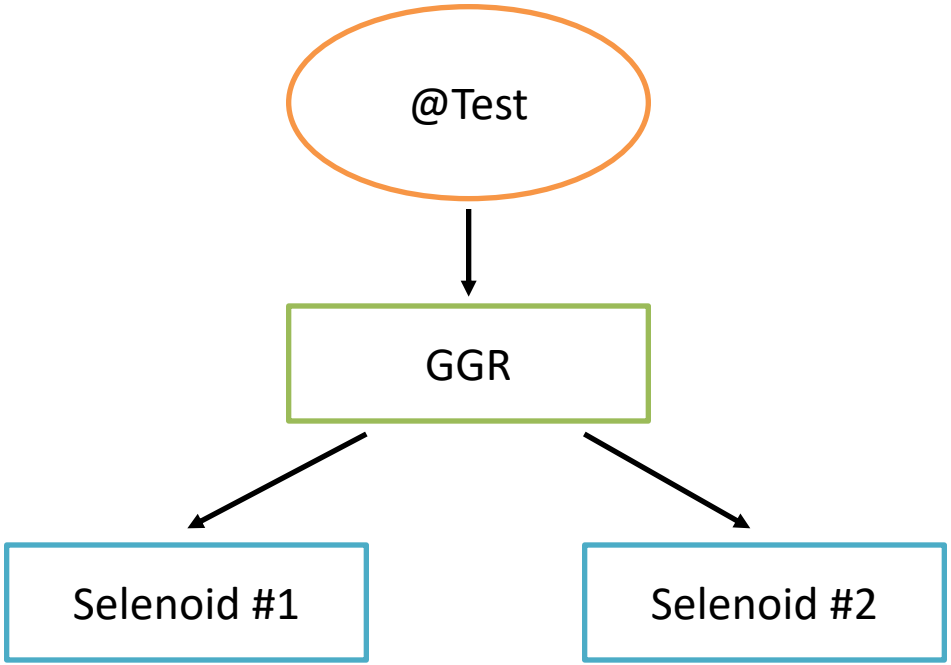
```
$ terraform apply
Acquiring state lock. This may take a few moments...

Error: Error locking state: Error acquiring the state
lock: Lock info:
  ID:   48e5ee9...
  Path: projects/selenoid/sel-1.tfstate
```

3. Remote backend: disadvantages

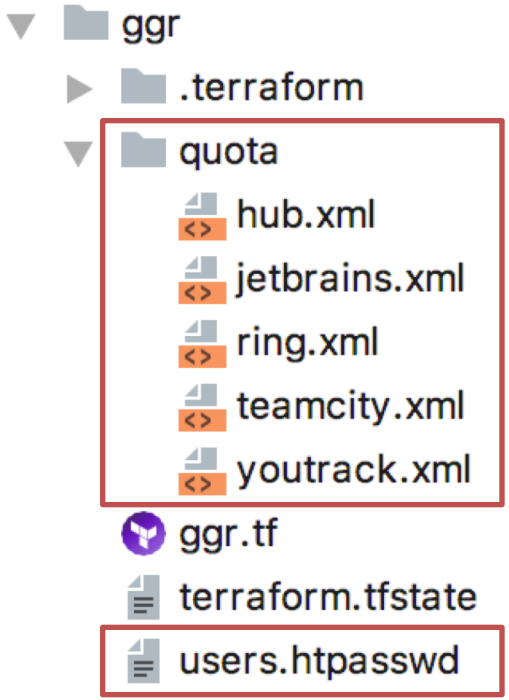
- **No way to put the backend configuration in a module**



4. Grid Router



4. Grid Router



4. Grid Router


```
resource "null_resource" "passwords" {
  triggers {
    passwords = "${sha1(file("./users.htpasswd"))}"
  }
  connection {
    host     = "${var.hostname}"
    type     = "ssh"
    user     = "***"
    password = "***"
  }
  provisioner "file" {
    source      = "./users.htpasswd"
    destination = "/home/jetbrains/users.htpasswd"
  }
}
```

4. Grid Router


```
resource "null_resource" "passwords" {
  triggers {
    passwords = "${sha1(file("./users.htpasswd"))}"
  }
  connection {
    host     = "${var.hostname}"
    type     = "ssh"
    user     = "***"
    password = "***"
  }
  provisioner "file" {
    source      = "./users.htpasswd"
    destination = "/home/jetbrains/users.htpasswd"
  }
}
```

4. Grid Router


```
resource "null_resource" "passwords" {
  triggers {
    passwords = "${sha1(file("./users.htpasswd"))}"
  }
  connection {
    host     = "${var.hostname}"
    type     = "ssh"
    user     = "***"
    password = "***"
  }
  provisioner "file" {
    source      = "./users.htpasswd"
    destination = "/home/jetbrains/users.htpasswd"
  }
}
```

4. Grid Router


```
resource "null_resource" "passwords" {
  triggers {
    passwords = "${sha1(file("./users.htpasswd"))}"
  }
  connection {
    host     = "${var.hostname}"
    type     = "ssh"
    user     = "***"
    password = "***"
  }
  provisioner "file" {
    source      = "./users.htpasswd"
    destination = "/home/jetbrains/users.htpasswd"
  }
}
```

4. Grid Router


```
resource "null_resource" "quota" {
  triggers {
    teamcity = "${sha1(file("./quota/teamcity.xml"))}"
    youtrack = "${sha1(file("./quota/youtrack.xml"))}"
    ...
  }
  connection { ... }
  provisioner "file" {
    source      = "./quota"
    destination = "/home/jetbrains/"
  }
  provisioner "remote-exec" {
    inline      = ["docker kill -s HUP ${docker_container.ggr.id}"]
  }
}
```

5. Virtual machine parameters

5. VMs in vSphere: non-official provider


```
variable "vsphere_user" {}
variable "vsphere_password" {}

provider "vmware" {
  vcenter_server       = "vcenter-srv.***.net"
  user                 = "${var.vsphere_user}"
  password             = "${var.vsphere_password}"
  insecure_connection = true
}
```

5. VMs in vSphere


```
resource "vmware_virtual_machine" "sel-1" {
  image = "Shared-images/ubuntu-16.04-testing-39.32"

  ...

  cpus = 4
  memory = 6144
}
```

5. VMs in vSphere


```
resource "vmware_virtual_machine" "sel-1" {
  image = "Shared-images/ubuntu-16.04-testing-39.32"

  ...

  cpus = 4
  memory = 6144
}
```

5. Virtual machine parameters

—

- **10 containers, 4 CPU, 8GB RAM**

5. Virtual machine parameters

- 10 containers, 4 CPU, 8GB RAM
- **8 containers, 4 CPU, 8GB RAM**

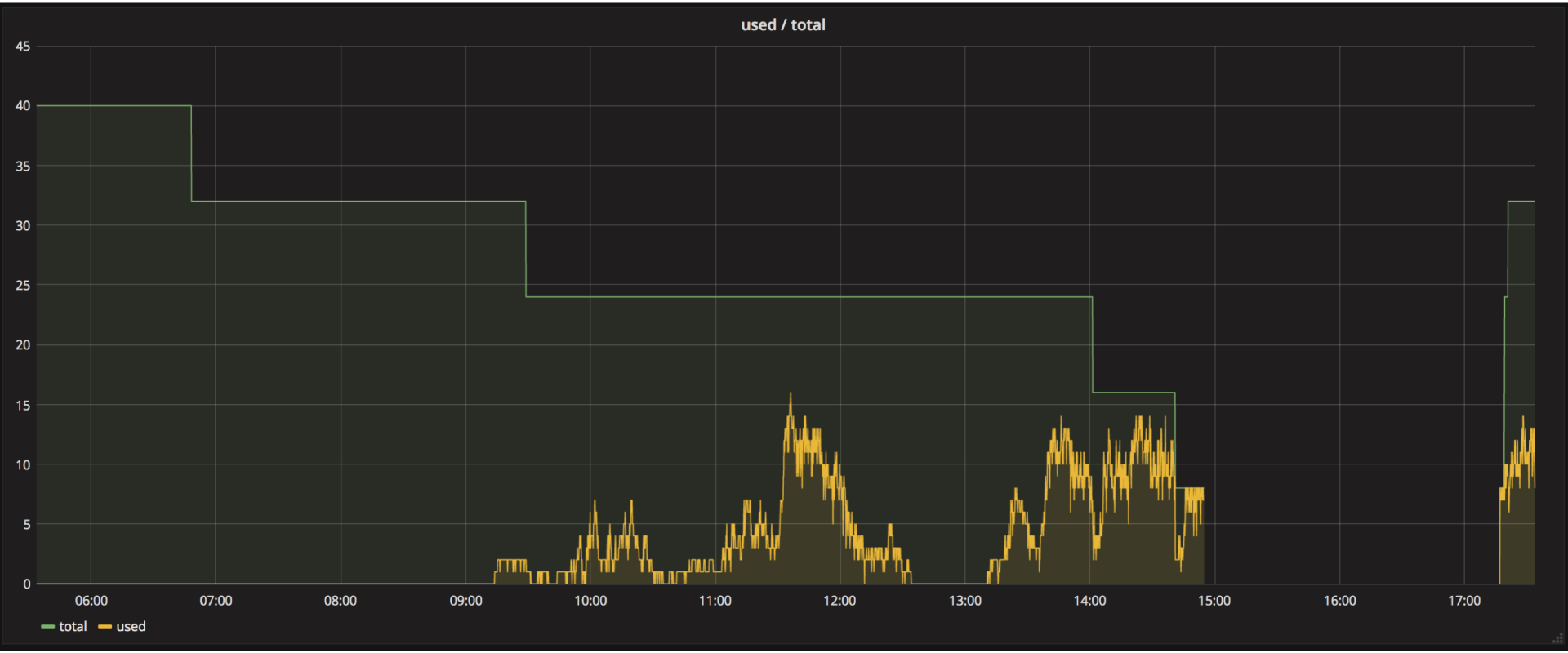
5. Virtual machine parameters

- 10 containers, 4 CPU, 8GB RAM
- 8 containers, 4 CPU, 8GB RAM
- **8 containers, 4 CPU, 6GB RAM**

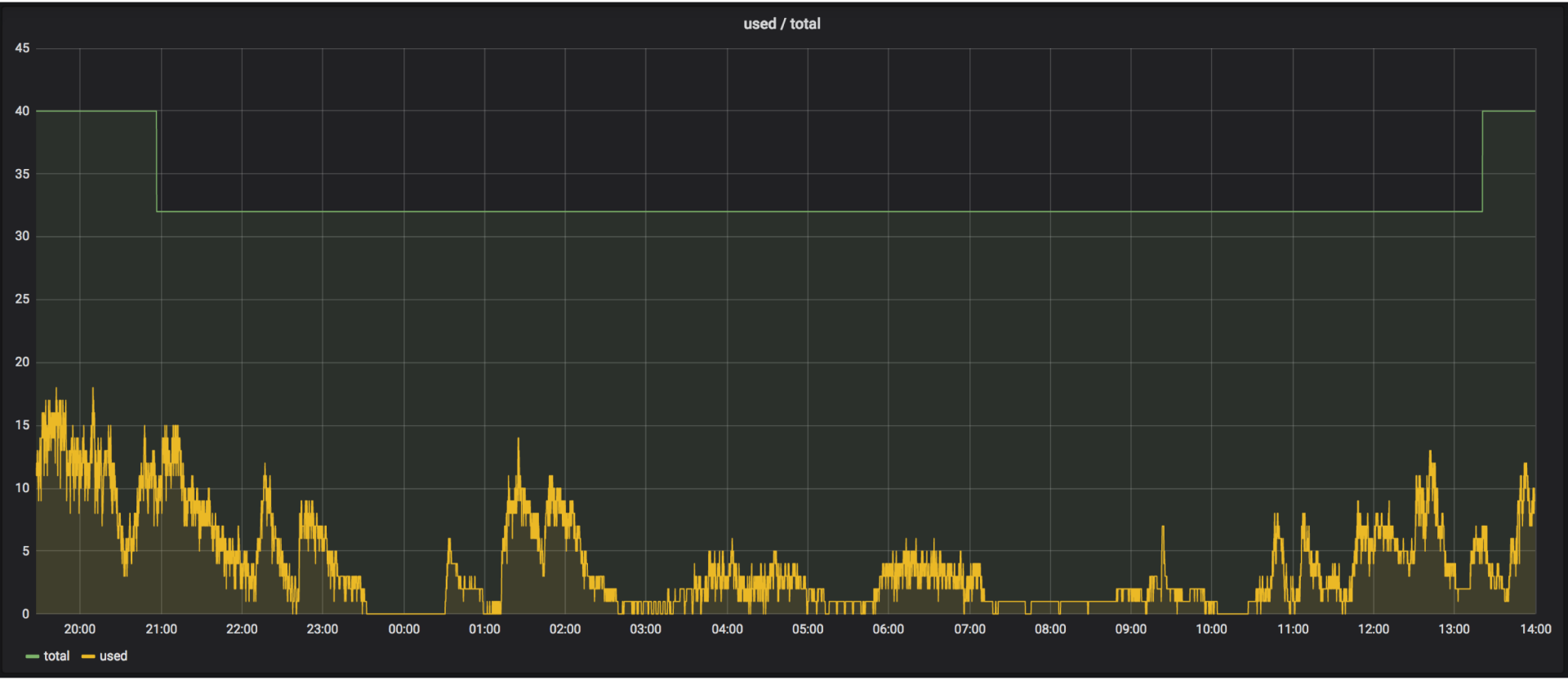
What’s going on here

1. Motivation
2. Terraform 101
3. Tips & Tricks for Selenoid cluster
4. Cluster monitoring

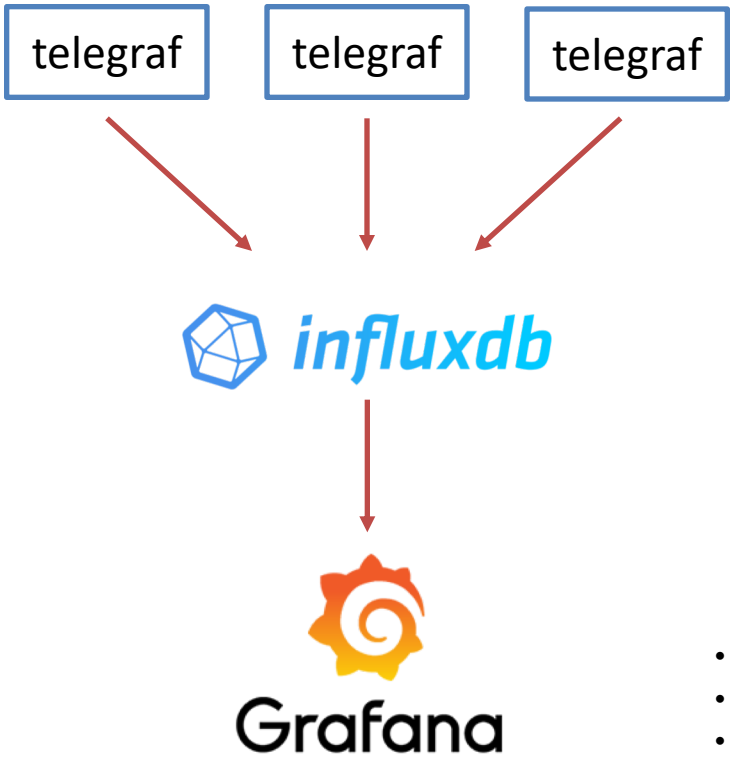
Monitoring


used / total
45
40
35
30
25
20
15
10
5
0
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
total
used

Monitoring


used / total
45
40
35
30
25
20
15
10
5
0
20:00
21:00
22:00
23:00
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
total
used

Monitoring



- https://docs.influxdata.com/telegraf
- https://docs.influxdata.com/influxdb
- http://docs.grafana.org/

1. Telegraf container


```
resource "docker_container" "telegraf" {
  image    = "${docker_image.telegraf.latest}"
  name     = "telegraf"
  hostname = "${var.selenoid_container_name}-telegraf"
  restart = "always"
  env = ["TZ=Europe/Moscow"]
  network_mode = "host"
  upload {
    content = "${file("${path.module}/telegraf.conf")}"
    file    = "/etc/telegraf/telegraf.conf"
  }
}
```

1. Telegraf container

```
resource "docker_container" "telegraf" {
  image    = "${docker_image.telegraf.latest}"
  name     = "telegraf"
  hostname = "${var.selenoid_container_name}-telegraf"
  restart = "always"
  env = ["TZ=Europe/Moscow"]
  network_mode = "host"
  upload {
    content = "${file("${path.module}/telegraf.conf")}"
    file    = "/etc/telegraf/telegraf.conf"
  }
}
```

1. Telegraf container


```
resource "docker_container" "telegraf" {
  image   = "${docker_image.telegraf.latest}"
  name    = "telegraf"
  hostname = "${var.selenoid_container_name}-telegraf"
  restart = "always"
  env = ["TZ=Europe/Moscow"]
  network_mode = "host"
  upload {
    content = "${file("${path.module}/telegraf.conf")}"
    file    = "/etc/telegraf/telegraf.conf"
  }
}
```

1. Telegraf config

```
[agent]
  interval = "10s" // collect data each 10 seconds
  round_interval = true // collect on :00, :10, :20, etc
  metric_batch_size = 1000
  metric_buffer_limit = 10000
  collection_jitter = "0s"
  flush_interval = "10s"
  flush_jitter = "0s"
  precision = ""
  debug = false
  quiet = false
  logfile = "" // to stderr
  hostname = "" // same as container
  omit_hostname = false // need to distinguish different hosts
```

1. Telegraf config: input plugins


```
[[inputs.cpu]] // cpu
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = false
[[inputs.disk]]
  mount_points = ["/"]
  ignore_fs = ["tmpfs", "devtmpfs", "devfs"]
[[inputs.mem]] // total / user / free / etc
[[inputs.system]] // load average 1 / 5 / 15
[[inputs.httpjson]] // selenoid
  servers = ["http://localhost:4444/status"]
  response_timeout = "5s"
  method = "GET"
```

1. Telegraf httpjson


```
{
  "total": 8,
  "used": 3,
  "queued": 0,
  "pending": 0,
  "browsers": {
    "chrome": {
      "63.0": {
        "teamcity": {
          "count": 1,
          "sessions": [ ... ]
        },
        "youtrack": {
          "count": 2,
          "sessions": [ ... ]
        }
      }
    }
  }
}
```

```
{
  "total": 8,
  "used": 3,
  "queued": 0,
  "pending": 0,
  "browsers_chrome_63.0_teamcity_count": 1,
  ...
  "browsers_chrome_63.0_youtrack_count": 2,
  ...
}
```

1. Telegraf config: output plugins


```
[[outputs.influxdb]]
  urls = ["http://selenium-dashboard:8086"]
  database = "telegraf" // will be created if needed
  retention_policy = ""
  write_consistency = "any"
  timeout = "5s"
  username = "telegraf"
  password = "metrics"
  content_encoding = "gzip"
```

2. InfluxDB container


```
resource "docker_container" "influxdb" {
  image   = "${docker_image.influxdb.latest}"
  name    = "influxdb"

  restart = "always"
  env = ["TZ=Europe/Moscow"]

  upload {
    content = "${file("./influxdb.conf")}"
    file = "/etc/influxdb/influxdb.conf"
  }
  ...
}
```

2. InfluxDB config


```
// just use default config, it is fine

[http]
  enabled = true
  bind-address = ":8086"
```

2. InfluxDB container

```
resource "docker_container" "influxdb" {
  ...
  ports {
    internal = 8086
    external = 8086
  }

  volumes {
    host_path      = "${var.home}influxdb"
    container_path = "/var/lib/influxdb"
  }
}
```

3. Grafana container


```
resource "docker_container" "grafana" {
  image   = "${docker_image.grafana.latest}"
  name    = "grafana"

  restart = "always"
  env = ["TZ=Europe/Moscow"]

  upload {
    content = "${file("./grafana.ini")}"
    file = "/etc/grafana/grafana.ini"
  }
  ...
}
```

3. Grafana config


```
[analytics]
reporting_enabled = false

[security]
admin_user = admin
admin_password = admin

[users]
allow_sign_up = false
allow_org_create = false

[auth.anonymous]
enabled = true
org_name = JetBrains
org_role = Viewer

[grafana_com]
url =

[smtp]
enabled = true
host = mail.***.net:2525
skip_verify = true
```

3. Grafana container

```
resource "docker_container" "grafana" {
  ...
  ports {
    internal = 3000
    external = 80
  }

  volumes {
    host_path      = "${var.home}grafana"
    container_path = "/var/lib/grafana"
  }
}
```

Data Sources / influxdb

Type: InfluxDB

Settings

Name	influxdb	Default	☑
Type	InfluxDB		

HTTP

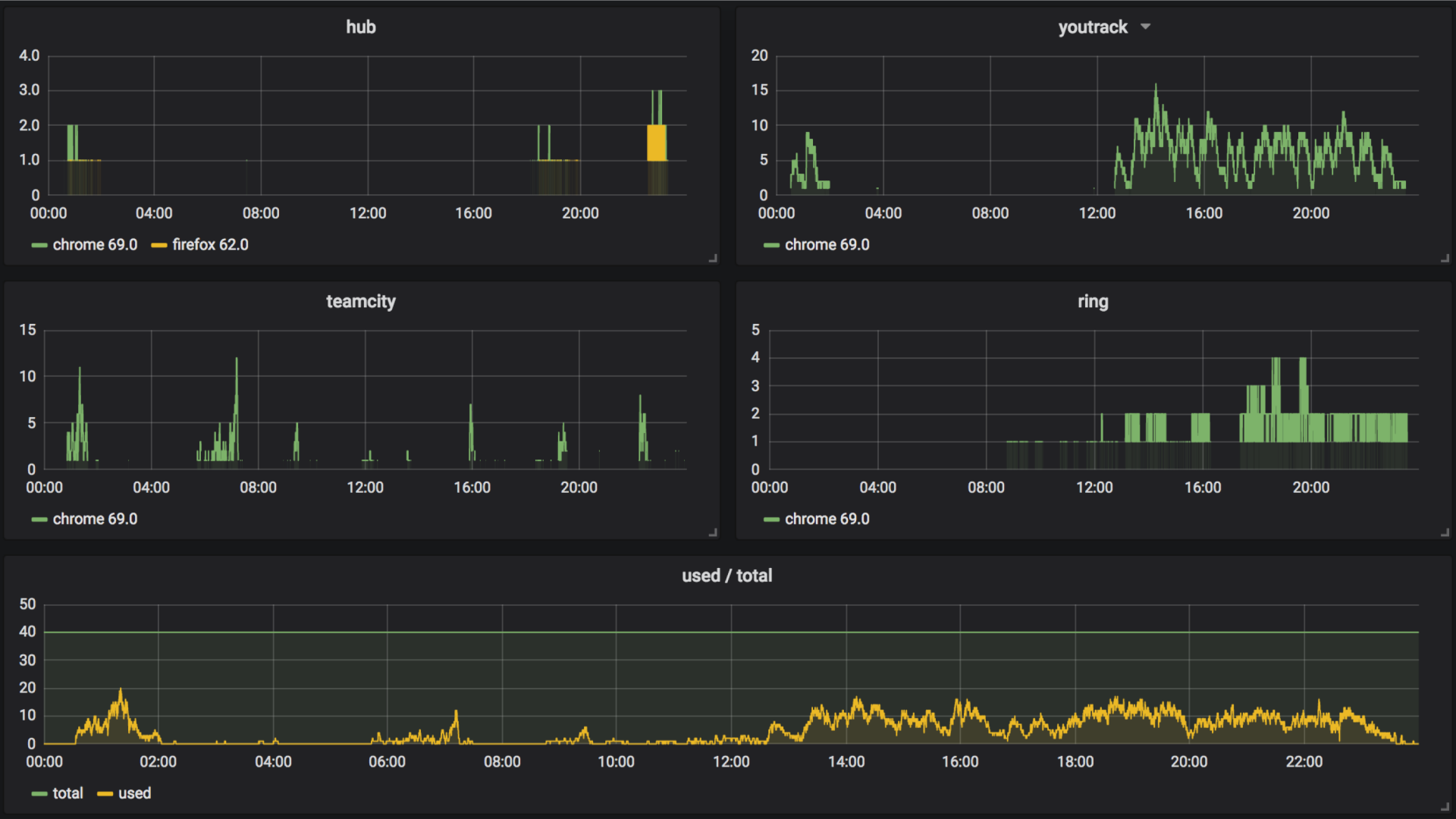
URL	http://selenium-dashboard:8086
Access	direct

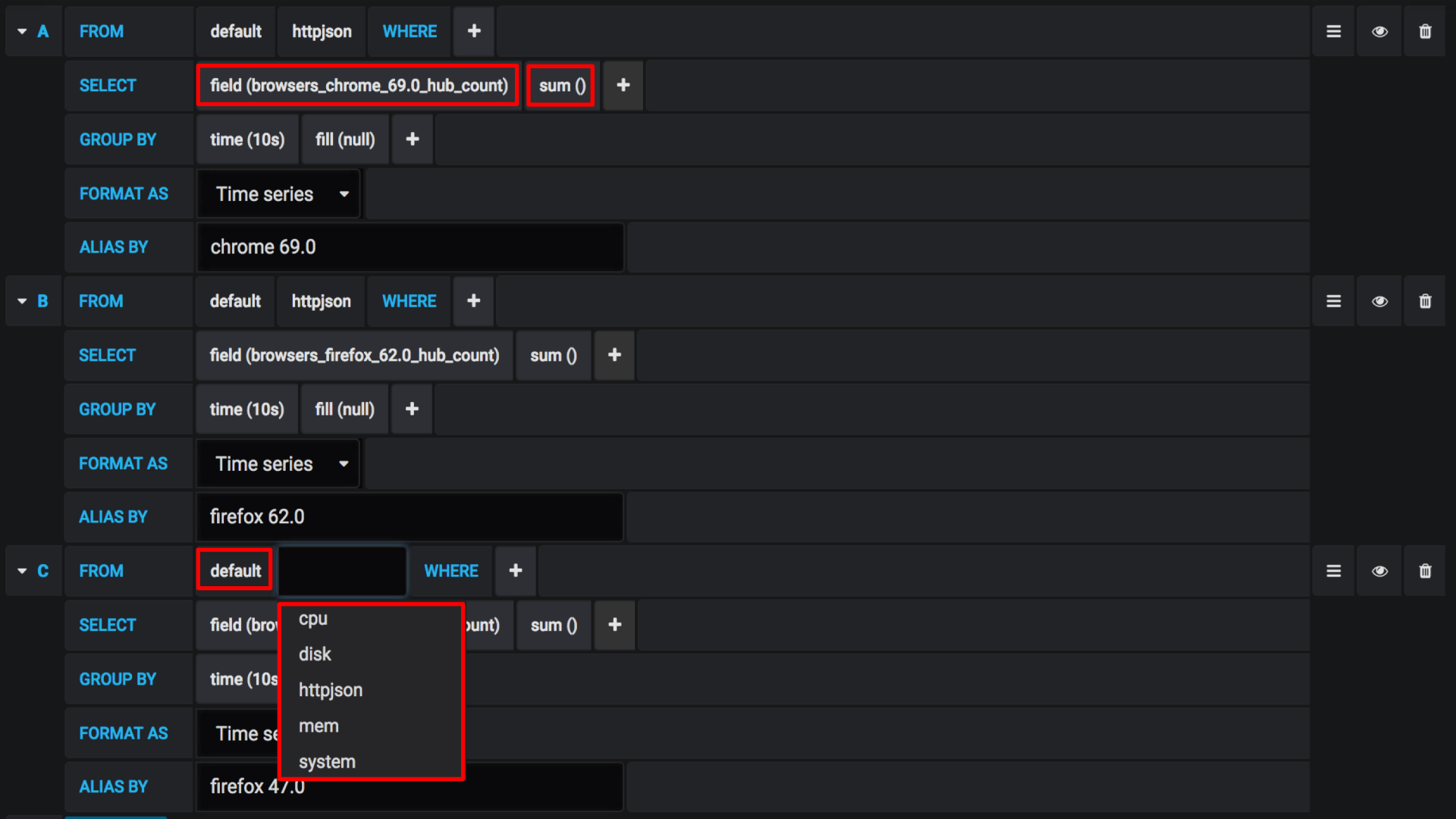
Auth

Basic Auth	☐	With Credentials	☐

InfluxDB Details

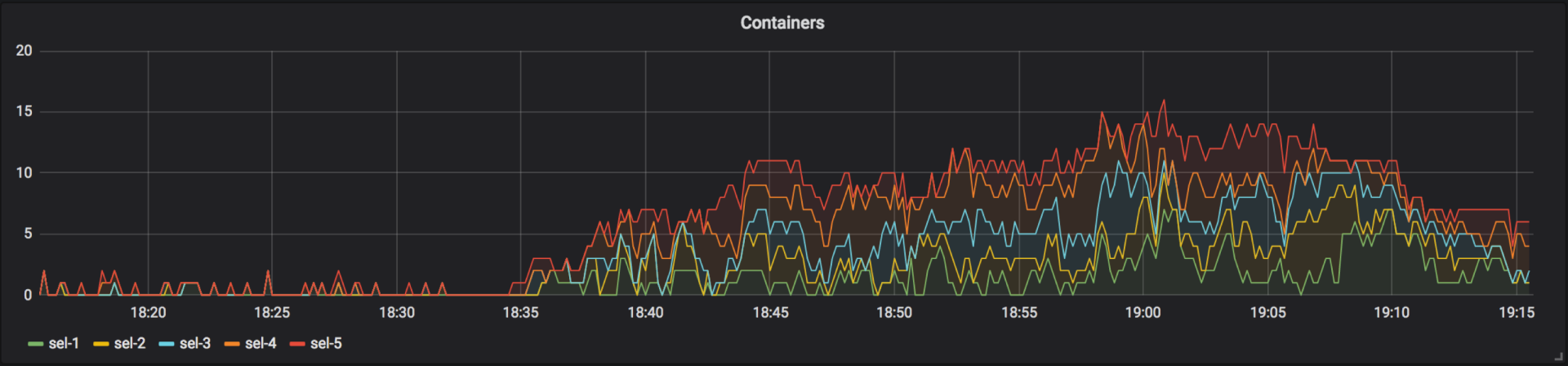
Database	telegraf		
User	telegraf	Password	•••••••••••••••••

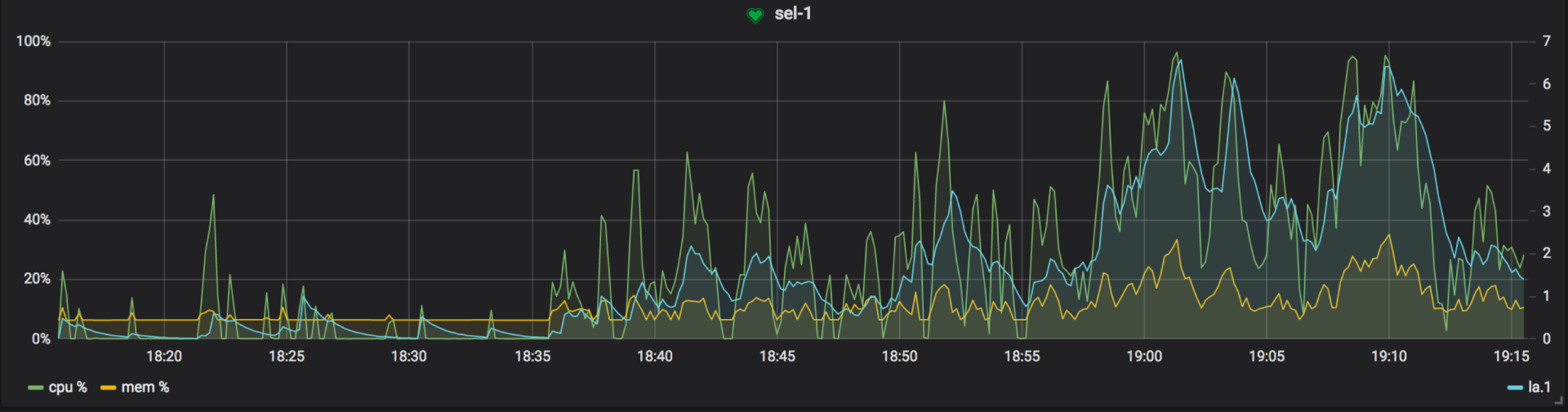

hub
4.0
3.0
2.0
1.0
0
00:00
04:00
08:00
12:00
16:00
20:00
chrome 69.0
firefox 62.0
youtrack
20
15
10
5
0
00:00
04:00
08:00
12:00
16:00
20:00
chrome 69.0
teamcity
15
10
5
0
00:00
04:00
08:00
12:00
16:00
20:00
chrome 69.0
ring
5
4
3
2
1
0
00:00
04:00
08:00
12:00
16:00
20:00
chrome 69.0
used / total
50
40
30
20
10
0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
total
used

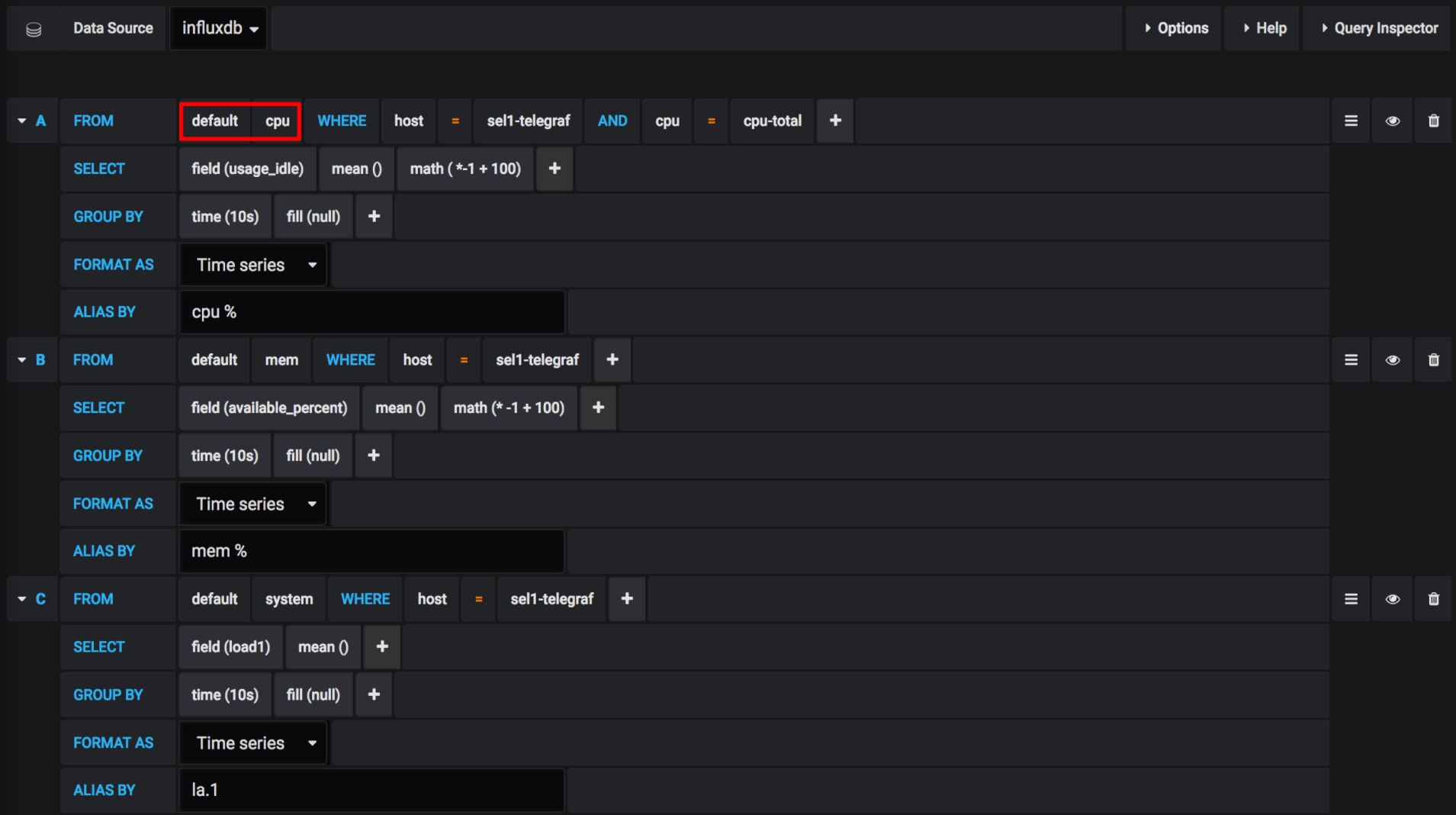

A
FROM
default
httpjson
WHERE
SELECT
field (browsers_chrome_69.0_hub_count)
sum ()
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
chrome 69.0
B
FROM
default
httpjson
WHERE
SELECT
field (browsers_firefox_62.0_hub_count)
sum ()
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
firefox 62.0
C
FROM
default
WHERE
SELECT
sum ()
GROUP BY
FORMAT AS
ALIAS BY
cpu
disk
httpjson
mem
system

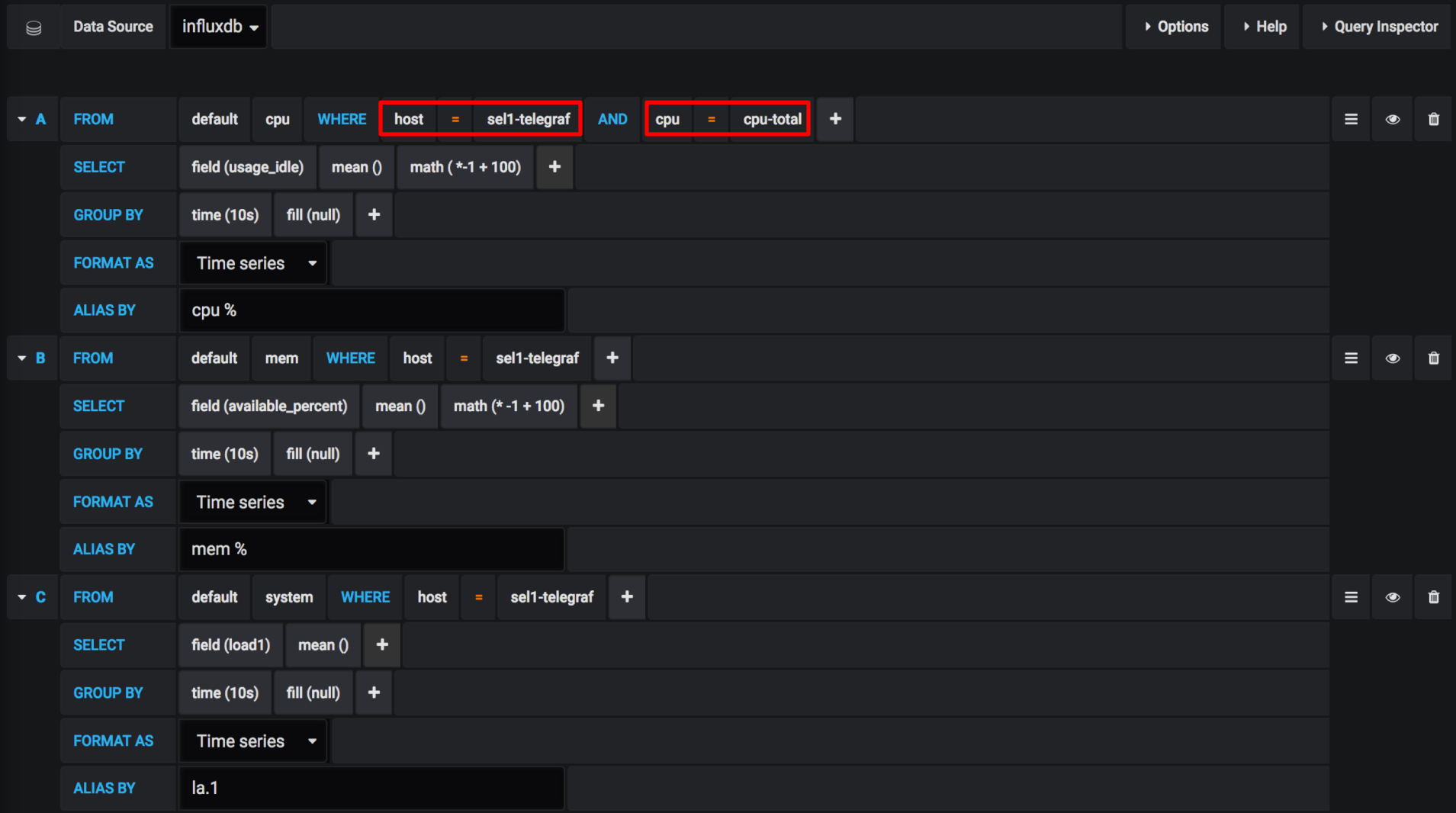
VMs

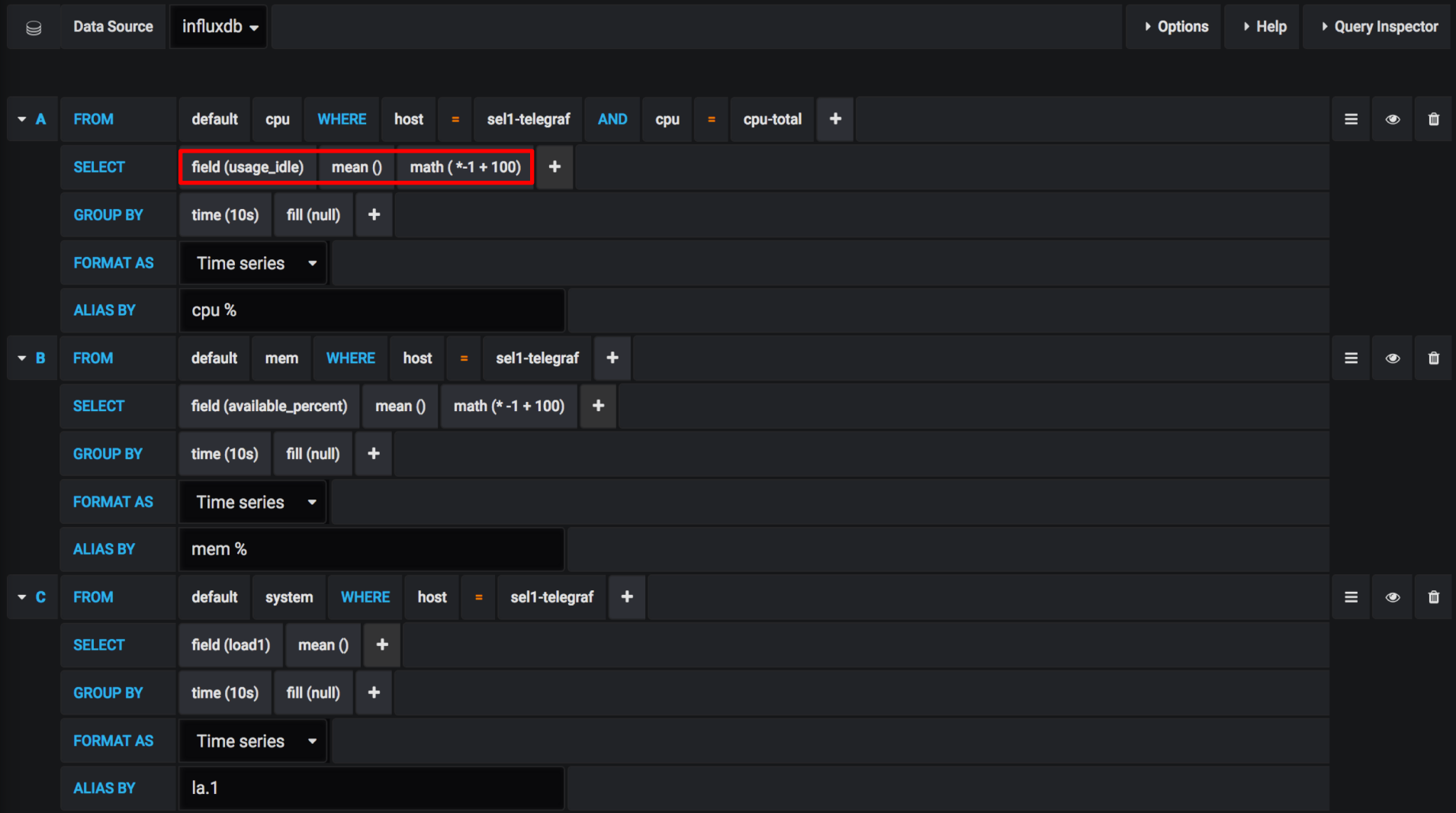
Last 1 hour Refresh every 10s

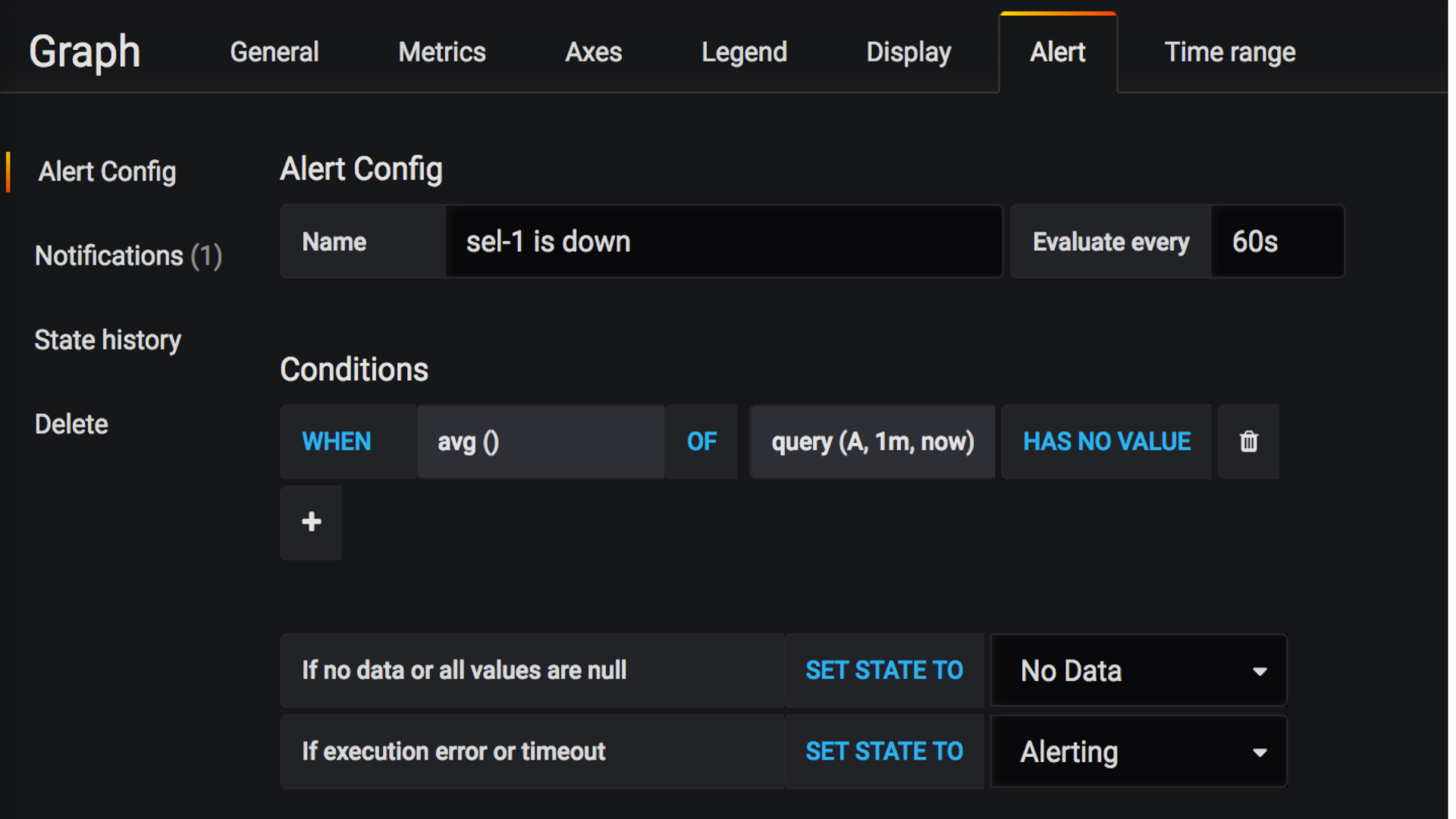





Data Source
influxdb
Options
Help
Query Inspector
A
FROM
default
cpu
WHERE
host
=
sel1-telegraf
AND
cpu
=
cpu-total
SELECT
field (usage_idle)
mean ()
math (*-1 + 100)
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
cpu %
B
FROM
default
mem
WHERE
host
=
sel1-telegraf
SELECT
field (available_percent)
mean ()
math (* -1 + 100)
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
mem %
C
FROM
default
system
WHERE
host
=
sel1-telegraf
SELECT
field (load1)
mean ()
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
la.1


Data Source
influxdb
Options
Help
Query Inspector
A
FROM
default
cpu
WHERE
host
=
sel1-telegraf
AND
cpu
=
cpu-total
SELECT
field (usage_idle)
mean ()
math (*-1 + 100)
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
cpu %
B
FROM
default
mem
WHERE
host
=
sel1-telegraf
SELECT
field (available_percent)
mean ()
math (* -1 + 100)
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
mem %
C
FROM
default
system
WHERE
host
=
sel1-telegraf
SELECT
field (load1)
mean ()
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
la.1

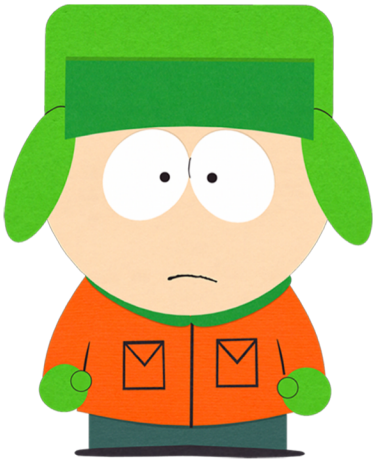

Data Source
influxdb
Options
Help
Query Inspector
A
FROM
default
cpu
WHERE
host
=
sel1-telegraf
AND
cpu
=
cpu-total
SELECT
field (usage_idle)
mean ()
math (*-1 + 100)
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
cpu %
B
FROM
default
mem
WHERE
host
=
sel1-telegraf
SELECT
field (available_percent)
mean ()
math (* -1 + 100)
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
mem %
C
FROM
default
system
WHERE
host
=
sel1-telegraf
SELECT
field (load1)
mean ()
GROUP BY
time (10s)
fill (null)
FORMAT AS
Time series
ALIAS BY
la.1


Graph
General
Metrics
Axes
Legend
Display
Alert
Time range
Alert Config
Notifications (1)
State history
Delete
Alert Config
Name
sel-1 is down
Evaluate every
60s
Conditions
WHEN
avg ()
OF
query (A, 1m, now)
HAS NO VALUE
+
If no data or all values are null
SET STATE TO
No Data
If execution error or timeout
SET STATE TO
Alerting

"You see, I learned something today..."

—

- Terraform basics
- Real-life Selenoid cluster example
- Take a look at your infrastructure



Thank you
for your attention
—

leonid.a.rudenko@gmail.com
@leonsabr