MICHAL MEDVECKY

USING TERRAFORM TO MANAGE YOUR OPENSTACK INFRASTRUCTURE

OPENSTACK CEE DAY BUDAPEST, JUNE 7TH 2017

WHO AM I

- Contracting my DevOps skills to Deutsche Telekom Pan-Net s.r.o.
- Prague/CZ, Bratislava/SK
- Long-time open-source enthusiast
- michal AT medvecky DOT net
- https://www.linkedin.com/in/michalmedvecky/

WHAT DO WE DO IN DT PAN-NET

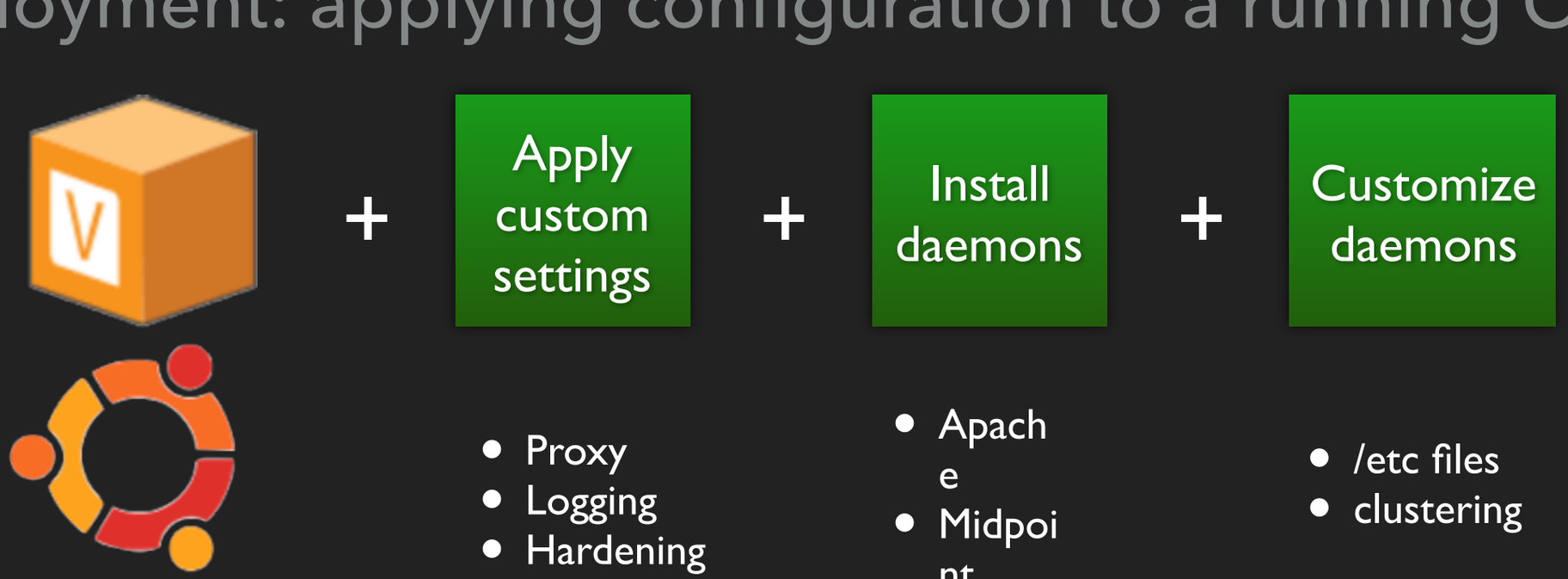
- CI/CD of security infrastructure components for Pan-Net clouds
- we get OS_AUTH_URL, OS_USERNAME, OS_PASSWORD
- we are expected to deliver the whole working infrastructure (auth services, registration services, administration services)
- one command to do the whole job
  - create machines
  - install + configure services
  - setup

(DIS)ADVANTAGES OF CI/CD APPROACH TO SYSTEM ADMINISTRATION

- We do not touch machines directly
- Number of servers/resources does not matter (at all)
- From idea to production, everything is logged
- Even project management is in code
- Automation, automation, automation
- Easy simple-to-complex development

PROVISIONING VS. DEPLOYMENT VS. SETUP

- Provisioning: process of creation of the *sshable* infrastructure
- Deployment: applying configuration to a running OS

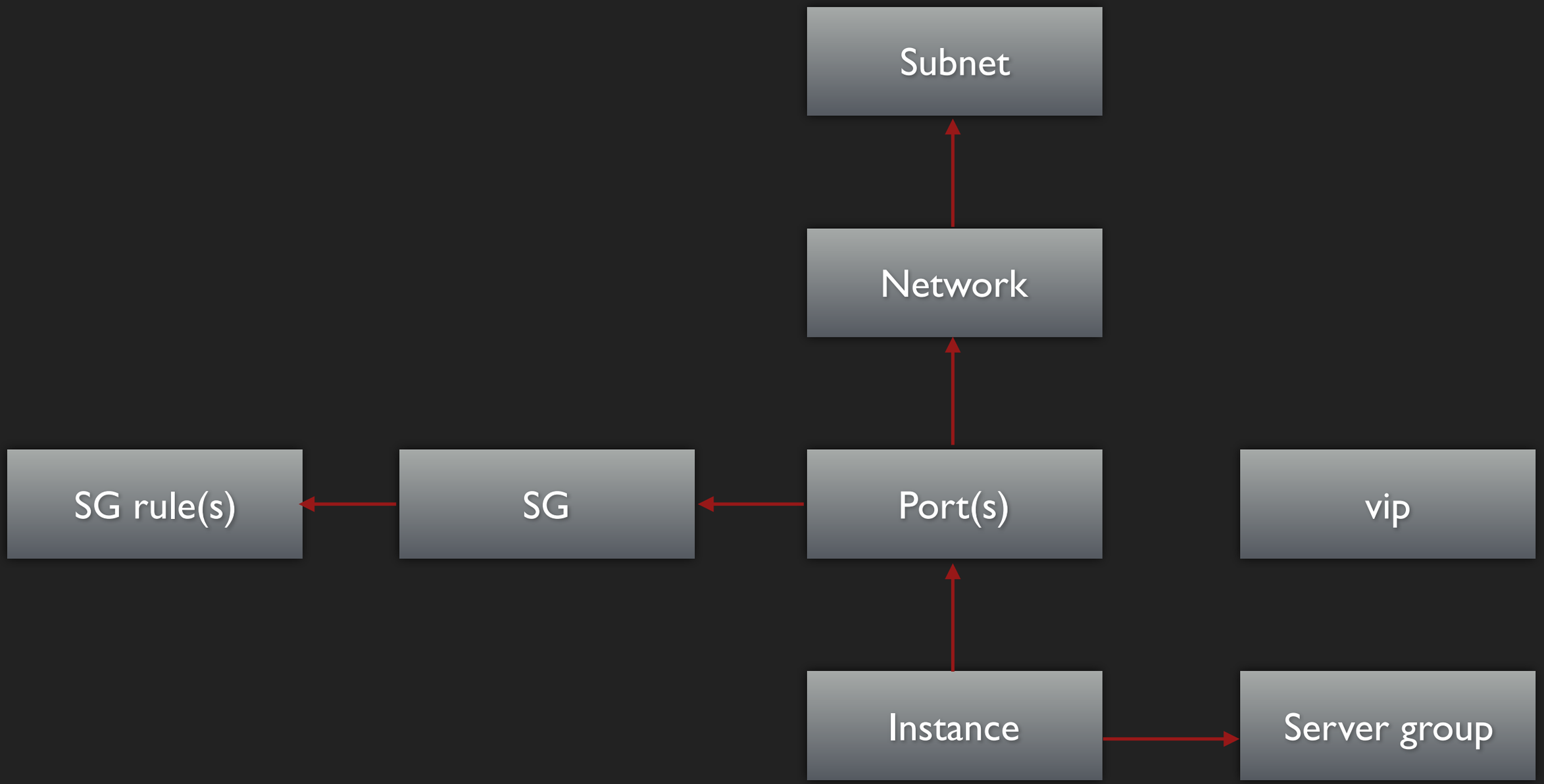


- Setup: Creating our own stuff (users, roles, interactions …)

OUR PROVISIONING OPTIONS

- Ansible (os_server module)
- Heat (you know) + SaltStack (omg)
- OpenStack CLI
- Juju
- Terraform

INFRASTRUCTURE IN OPENSTACK



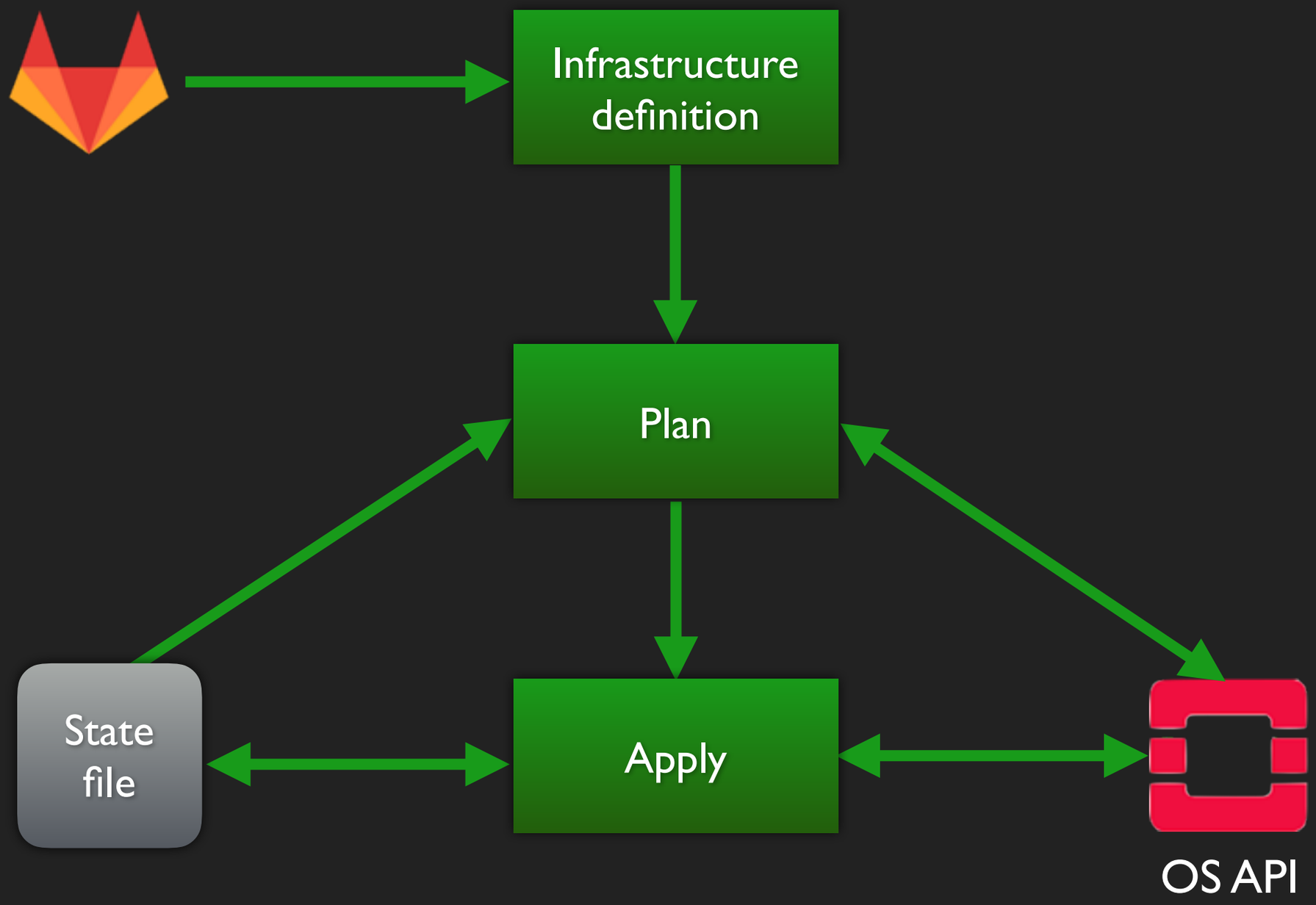
WHAT IS TERRAFORM?

- all-in-one ultimate IaaC tool under (heavy) development
- (partially) supporting multiple cloud providers
- open-source, golang, developed on GitHub
- made by Hashicorp, Inc.
- http://terraform.io/
- latest version is 0.9.6 (not even close to stable)
- binary distribution, no deployment, just wget + chmod + run

HOW TERRAFORM IS USED

- You declaratively describe (“code”) your infra components
- It communicates with cloud provider APIs and makes the infrastructure match your definition (creates/destroys/changes resources)
- Terraform keeps its own cache of infra state
- You can describe resources in multiple clouds (those that are implemented and not buggy)
- You can start from very scratch

TERRAFORM LIFECYCLE



SUPPORTED RESOURCES

- images_image
- networking_network
- blockstorage_volume_v1
- blockstorage_volume
- blockstorage_volume_attach
- compute_floatingip
- compute_floatingip_associate
- compute_instance
- compute_keypair
- compute_secgroup
- compute_servergroup
- compute_volume_attach
- dns_recordset
- dns_zone
- networking_floatingip
- networking_network
- networking_port
- networking_router_interface
- networking_router_route
- networking_router
- networking_subnet
- networking_secgroup
- networking_secgroup_rule
- lb_member_v1
- lb_monitor_v1
- lb_pool_v1
- lb_vip_v1
- lb_loadbalancer
- lbaas_listener
- lbaas_pool
- lbaas_member
- lbaas_monitor
- and more

SIMPLEST DEFINITION

```
resource “openstack_networking_secgroup_v2” "idm_backends" {

  name = "idm-backends"

  description = "IDM backend servers"

}
```

VARIABLES

```
variable "domain" {
  description = "Deployment domain name”
}
```

```
domain = “something.funny.in.pan-net.eu"
```

INSTANCE DECLARATION EXAMPLE

```
resource "openstack_compute_instance_v2" "idm" {

  count = "${lookup(var.server_counts,"idm")}"
  name = "idm0${count.index+1}.${var.domain}"
  image_id = "${var.image_id}"
  flavor_name = “m1.large"
  key_pair = "${var.keypair}"
  availability_zone = "mz"

  metadata {
    tags = "idm,lb-idm-backend,${var.deployment_name}"
  }

  network {
    port = "${element(openstack_networking_port_v2.idm_port_backend.*.id,
            count.index)}"
}

}
```

SG AND SG RULE

```
resource "openstack_networking_secgroup_v2" "idm_endpoints" {

  name = "${var.sg_prefix}idm-endpoints"
  description = "IDM Load Balancers"

}
```

```
resource "openstack_networking_secgroup_rule_v2" "allow-idm-access-to-lbs-ssl" {

  direction = "ingress"
  ethertype = "IPv4"
  protocol = "tcp"
  port_range_min = 443
  port_range_max = 443
  remote_ip_prefix = "0.0.0.0/0"
  security_group_id = "${openstack_networking_secgroup_v2.idm_endpoints.id}"

}
```

STATE FILE

- by default, saved to terraform.tfstate (local file)
- in OpenStack, you can use Swift
- but there are dragons
  - does not support locking (with Swift)
  - you need to backup the file
  - you should be able to edit it manually

SAMPLE TFSTATE CONTENT


```
"openstack_networking_secgroup_v2.idm_backends": {
    "type": "openstack_networking_secgroup_v2",
    "depends_on": [],
    "primary": {
        "id": "609f20d3-c76e-4927-a0f2-ac9df16521b5",
        "attributes": {
            "description": "IDM backend servers",
            "id": "609f20d3-c76e-4927-a0f2-ac9df16521b5",
            "name": "tf-iam-idm-backends",
            "region": "RegionOne",
            "tenant_id": "3e8f42a379284709b3abd556a6885102"
        },
        "meta": {},
        "tainted": false
    },

    "deposed": [],
    "provider": ""

},
```

IMPORTING EXISTING INFRASTRUCTURE

```
terraform import openstack_networking_secgroup_v2.idm_backends UUID
```

- not all resources support import (e.g. compute_instance)
- beware of terraform destroy
  - you might not be willing to destroy your imported resources

TERRAFORM DESTROY

- destroys all your (tf managed) infrastructure
- clears the state file
- keeps your definition (of course :)

HOW TO HAPPILY USE IT WITH OPENSTACK

- You need to understand your infrastructure well …
  - … and what Terraform does for you.
- Decouple your infrastructure to very basic objects
- Share the state file wisely
- Always check “terraform plan” before applying
- Always expect trouble (bugs, unexpected behavior)

. . . CONTINUED

- Don’t use it in non-fully-automated environments
- Expect yourself to write bug reports
- Expect yourself to compile latest commits

THANK YOU.