Introduction to Terraform

AiDO
DevOps and more

Ansible in DevOps

MeetUp #6; Bydgoszcz 2020-06-10

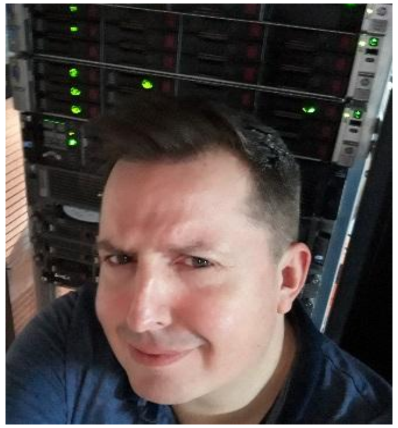
Waldemar Bałdowski

<waldemar.baldowski@gmail.com>

Agenda

- About me
- Terraform overview – architecture, IaC, commands
- From simple code to modules – online demo
- Reference materials
- Q&A

About me



Professional profile

- Now: Technical Architect in Atos
- Past: PSI, TE, Sygnity, Red Hat
- Focused on:
  - Tooling and Automation
  - Privite and Public Cloud
  - Infrastructure as a Code
  - DevOps and Agile methodologies
  - In love with GNU/Linux ;)

Email: waldemar.baldowski@gmail.com

Private profile

- Lives in Bydgoszcz
- Married, have got son
- Interested in:
  - MeetUp like our about Ansible ;)
  - Music production, DAW
  - Guitar & Accordion Player
  - Member of "Half the Way" band
  - Likes spend time on sea side

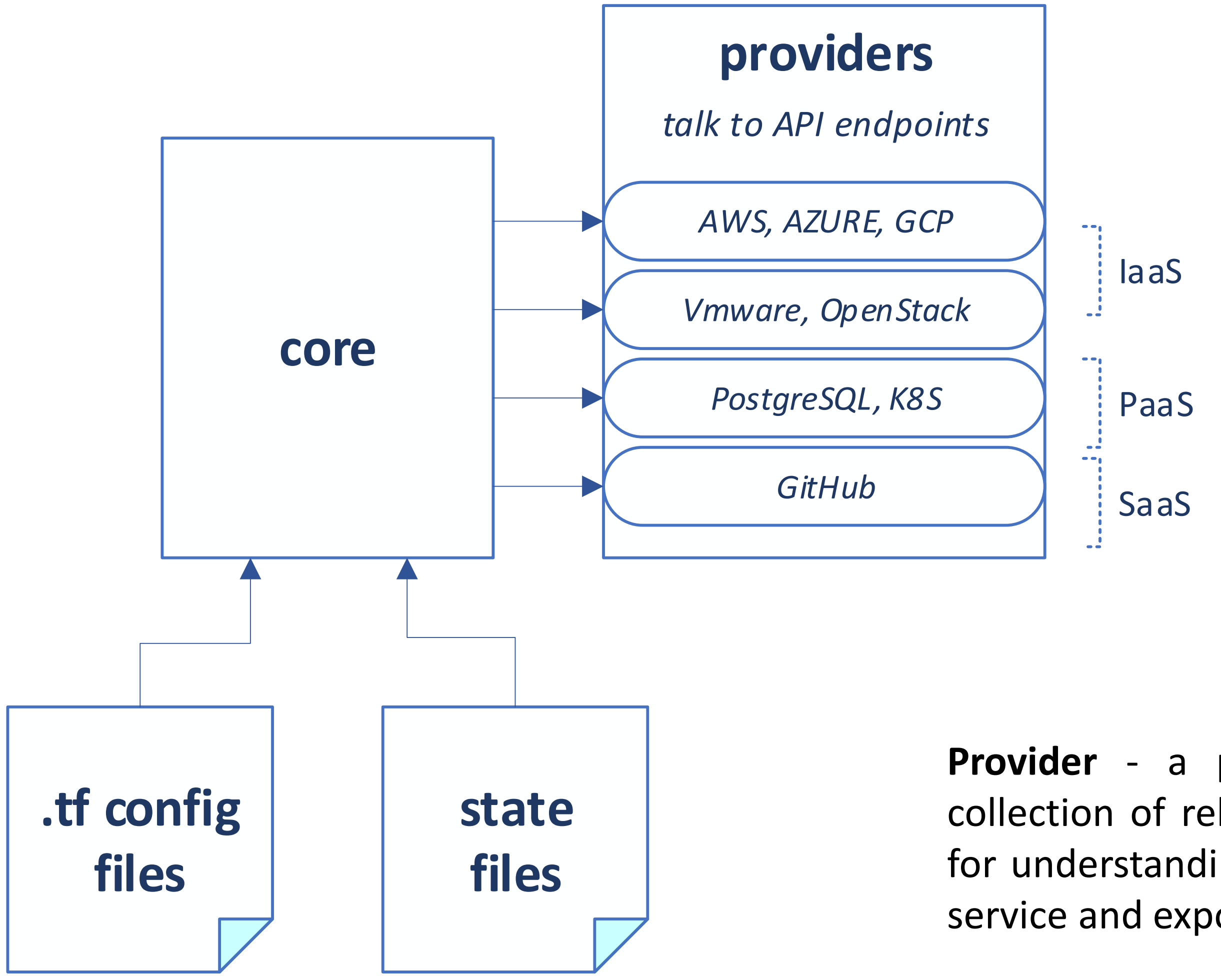
What is terraform?

- **Terraform** – A tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions. In another word it is infrastructure provisioning tool, like CloudFormation.

- **Terraform Cloud** - An application that helps teams use Terraform together. Terraform Cloud provides a consistent environment for Terraform runs, as well as hosting Terraform state and other shared information. Terraform Cloud is available as a hosted service at https://app.terraform.io. Free 30-days account for full features testing is available.
- It is also available in an on-premises distribution called **Terraform Enterprise**.

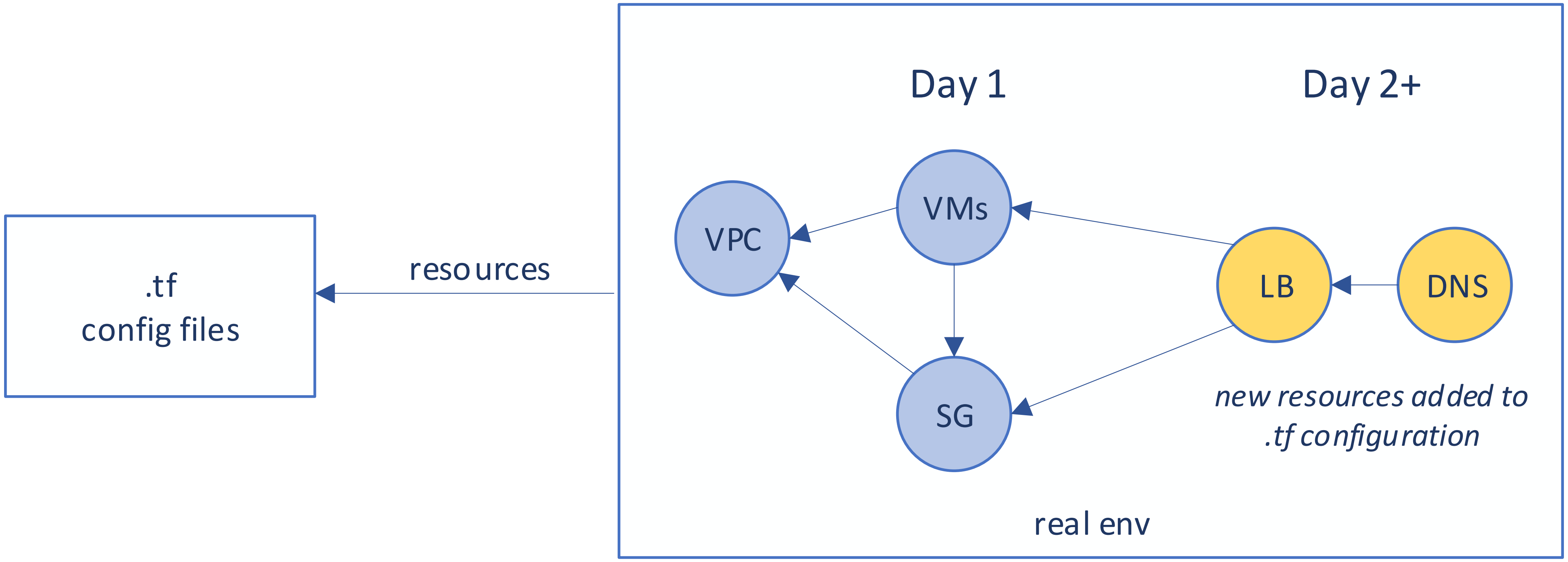
- Full glossary from Terraform.

Architecture overview

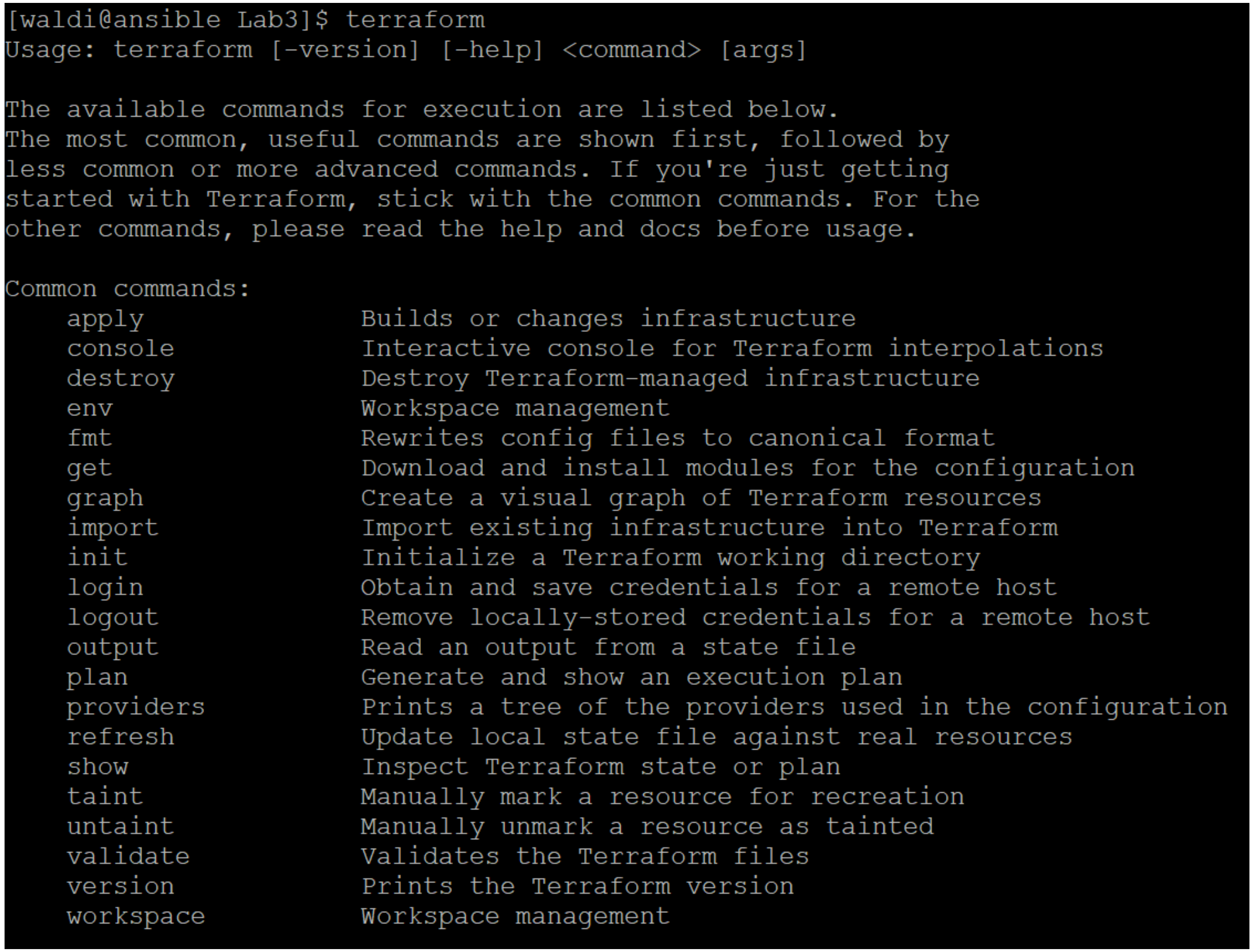


Provider - a plugin for Terraform that makes a collection of related resources available. Responsible for understanding [API](#) interactions with some kind of service and exposing [resources](#) based on that API.

Infrastructure as a code approach



Basic commands #1

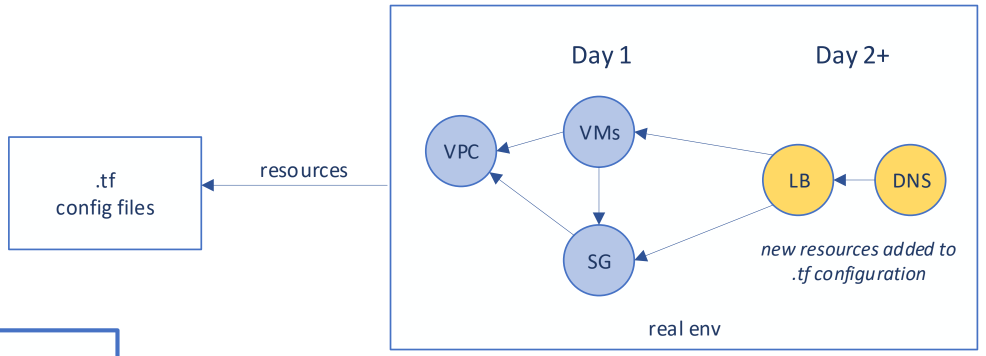


```
[waldi@ansible Lab3]$ terraform
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Workspace management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a Terraform working directory
    login              Obtain and save credentials for a remote host
    logout             Remove locally-stored credentials for a remote host
    output             Read an output from a state file
    plan               Generate and show an execution plan
    providers          Prints a tree of the providers used in the configuration
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version
    workspace          Workspace management
```

Basic commands #2



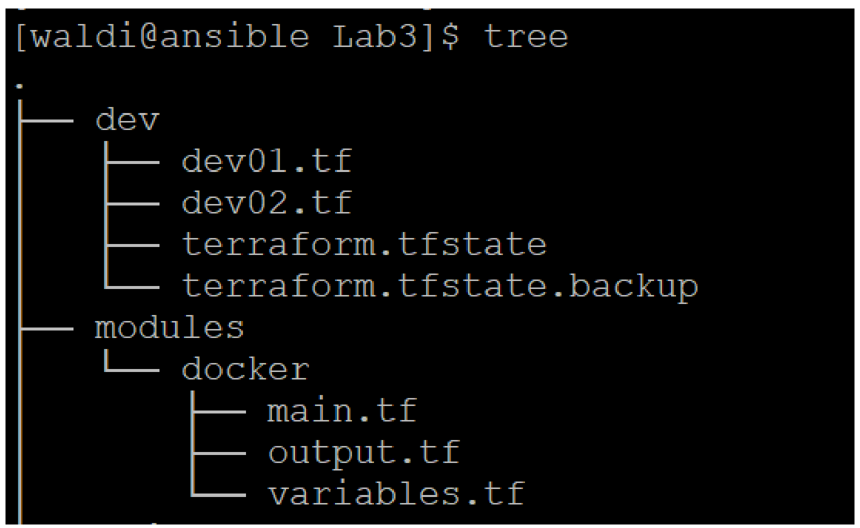
Commands	Actions
terraform init (one time)	initialize providers
terraform refresh	tf view <– real env
terraform plan	real env –> desired state
terraform apply	plan –> real env
terraform destroy	plan -> destroy real env

Symbols in terraform plan output:

+ create
- destroy
-/+ replace (destroy and then create)
~ update in-place
<= read (data sources)

From simple config to module

- Online demo



```
[waldi@ansible Lab3]$ tree
.
├── dev
│   ├── dev01.tf
│   ├── dev02.tf
│   ├── terraform.tfstate
│   └── terraform.tfstate.backup
├── modules
│   └── docker
│       ├── main.tf
│       ├── output.tf
│       └── variables.tf
```

- Example review from public registry

  https://registry.terraform.io/modules/terraform-aws-modules/rds/aws/2.14.0

Reference materials

- Terraform CLI:
  - https://www.terraform.io/docs/cli-index.html
- Installation packages:
  - https://www.terraform.io/downloads.html
- Providers:
  - https://www.terraform.io/docs/providers/index.html
- Backends for Team working (storage for state file):
  - https://www.terraform.io/docs/backends/index.html
- Public registry:
  - https://registry.terraform.io/ https://github.com/terraform-aws-modules/
- Terraform Enterprise/Cloud - Sentinel:
  - https://www.terraform.io/docs/cloud/sentinel/index.html
- Modules:
  - https://www.terraform.io/docs/modules/index.html

Thank you