Эффективное управление инфраструктурой

TECH

Обо мне

- K8s
- Hasicorp stack
- Prometheus stack
- Istio
- GitlabCI
- Gitops

@Mrgreyves

Дроздецкий Владимир

DevOps Teamlead

О нас

10 k8s

clusters
(1000+ CPU cores, 2+ TB Ram)

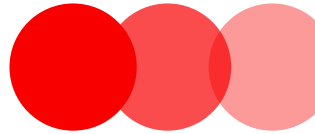
- Golang/Python
- Postgresql
- Redis
- Elasticsearch

Подробнее о нас:
https://github.com/magnit-tech/magnit-online

MJ TECH

План

- О чем IaC
- Terraform и как с ним жить
- Как готовить terraform модули
- Terragrunt – зачем он нам?
- Нам тоже нужны CI/CD пайплайны для инфраструктуры
- Секция вредных советов



IaC

Как много всего в этих трех буквах

IaC

А всегда ли он нам нужен?

IaC

Инструменты для управления инфраструктурой:

01 Terraform

02 Pulumi

03 AWS Cloudformation

04 CDK

05 Ansible, Chef, Puppet, etc.

06 Bash ^_^

Terraform

- Декларативное управление инфраструктурой (HCL)
- Множество провайдеров (если нет нужного, напиши свой)
- State + Backend
- Модули
- Командная разработка

Terraform

Блокировка

- VPN
- Локальное кэширование
- Зеркала (а оно точно безопасное?)
- Свой образ с установленными провайдера (не забывай обновлять)

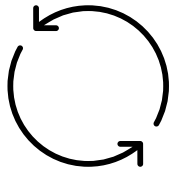
Backend

- Local
- Terraform cloud + opensource alternatives
- S3
- Gitlab

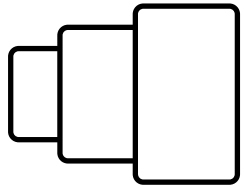
Terraform

Backend – а нужны ли нам блокировки?

Terraform modules



Переиспользование кода



Версионирование (git tag наше все)

Terraform modules

- Структура репозитория
- Типичного модуля

Terraform modules

Переменные

- Много переменных не бывает
- Значение по умолчанию
- Описание

```
variable "memory" {
  description = "VM RAM size"
  default     = 2
  type        = number
}

variable "disk" {
  description = "VM Disk size"
  default     = 10
  type        = number
}

variable "zone" {
  description = "Default zone"
  default     = "ru-central1-a"
  type        = string
}
```

Terraform modules

Переменные

- Проверка значений

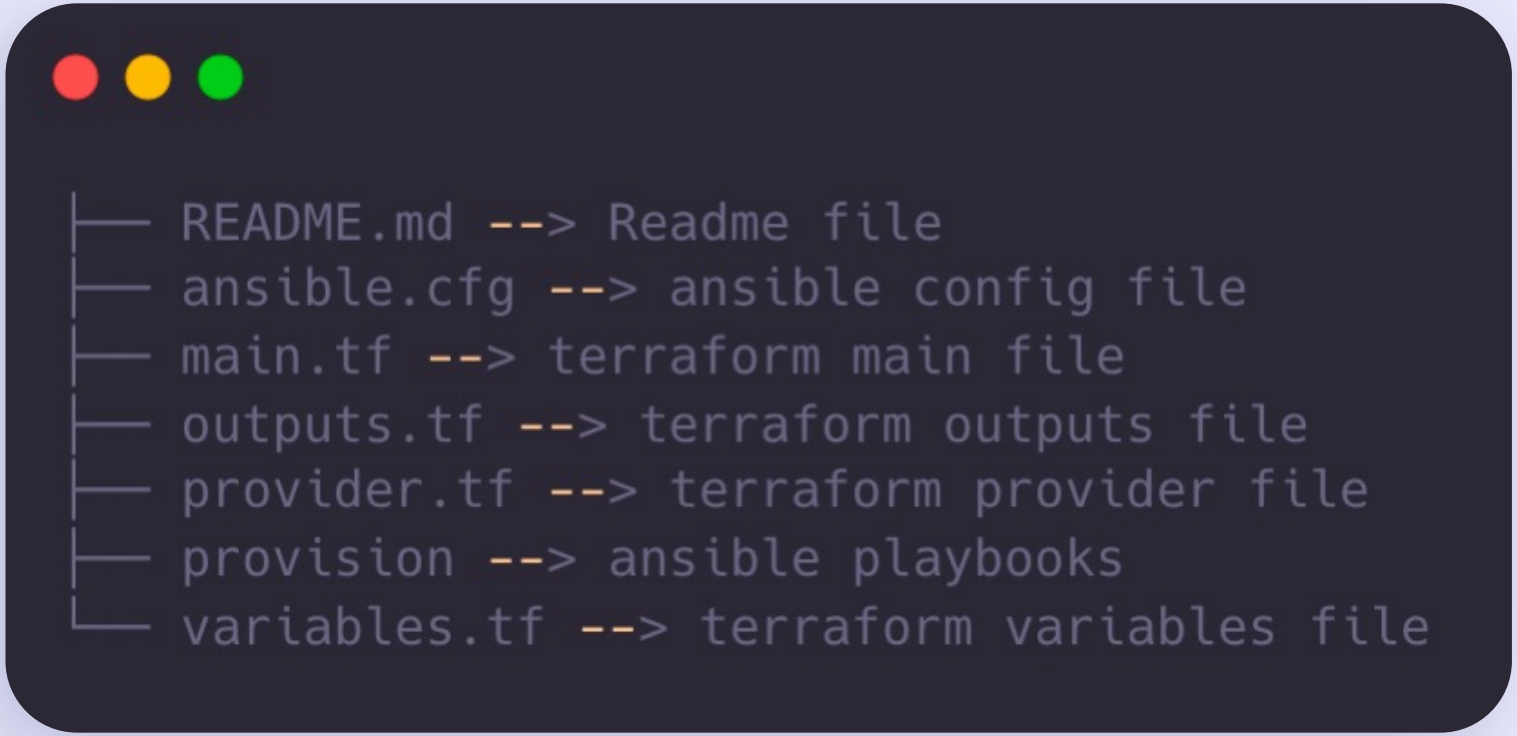
```
variable "psql_service_name" {
  description = "Managed Service for PostgreSQL name"
  type        = string
  validation {
    condition     = can(regex("^[0-9A-Za-z_]*$", var.psql_service_name))
    error_message = "The psql_service_name value can contain numbers, letters and
_."
}
```

Terraform modules

Нужен ли нам базовый модуль?

Terraform modules

Типичный generic модуль выглядит так:



Terraform modules

Main.tf

```
resource "yandex_compute_instance" "generic"
resource "null_resource" "node-exporter"
resource "null_resource" "ca-cert"
resource "null_resource" "setup-users"
resource "null_resource" "crt-issuer-consul-template"
resource "consul_node" "node"
resource "consul_service" "node"
resource "yandex_dns_recordset" "generic"
resource "yandex_dns_recordset" "generic-custom-dns"
resource "yandex_compute_snapshot_schedule"
"snapshot"
```

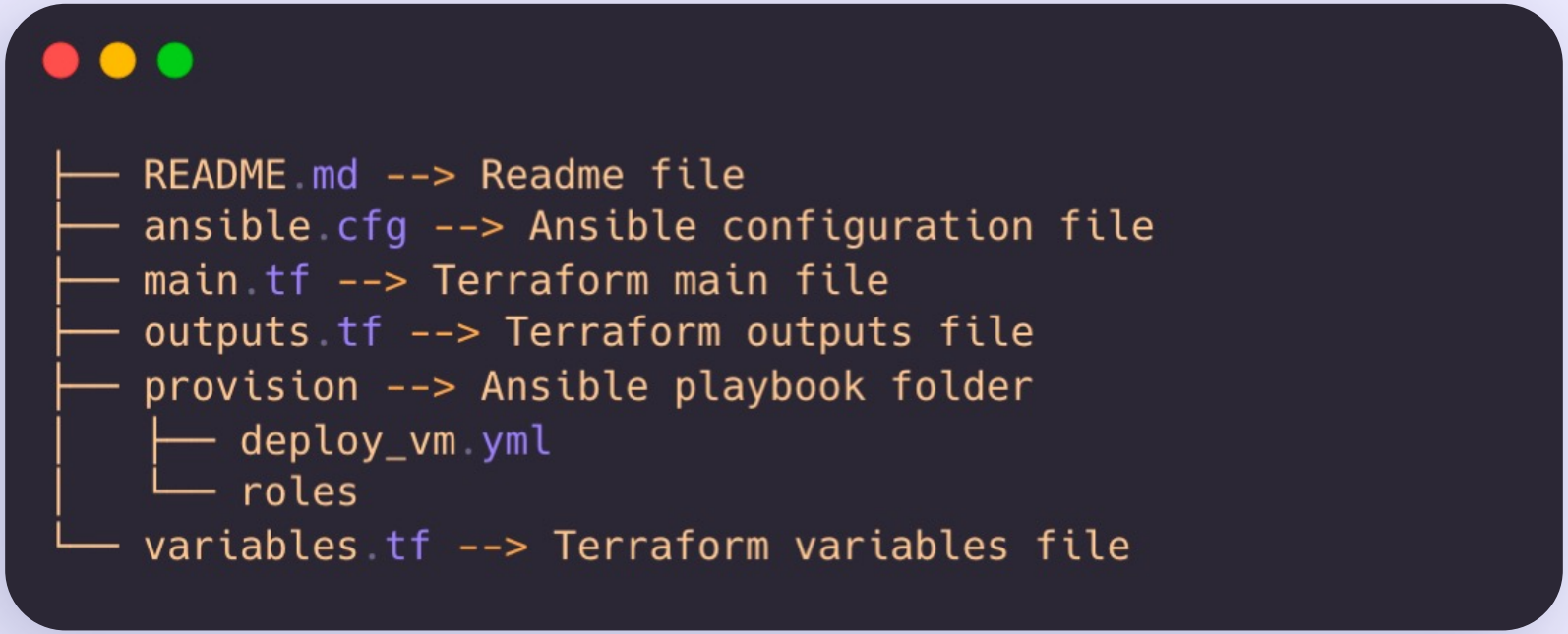
Terraform modules

Null_resource

```
resource "null_resource" "node-exporter" {
  count = var.num
  provisioner "local-exec" {
    command = format("ansible-playbook -D -i %s, -u ubuntu
%s/provision/node-exporter.yml",
      var.bootstrap &&
yandex_compute_instance.generic[count.index].network_interface[0].nat_ip_add
ress != "" ?
yandex_compute_instance.generic[count.index].network_interface[0].nat_ip_add
ress :
yandex_compute_instance.generic[count.index].network_interface[0].ip_address
,
    path.module, )
  }
  depends_on = [yandex_compute_instance.generic]
}
```

Terraform modules

- Модуль типичного сервиса на ВМ
- (Общая структура репозитория)



Terraform modules

Main.tf

типичного модуля

```
module "sentry" {
  source = "git@gitlab.com:olololo-trolololo/generic.git//.?ref=3.0.2"
}

variable "sentry_version" {
  type    = string
  default = "23.3.1"
}

resource "null_resource" "sentry_playbook" {
  depends_on = [module.sentry]
}

resource "consul_node" "sentry" {
  depends_on = [module.sentry]
}

resource "consul_service" "sentry" {
  depends_on = [
    consul_node.sentry,
  ]
}
```

Terraform modules

Окончательный вызов модуля

```
terraform {
  source = "git@gitlab.com:olololo-trololo/sentry.git//.?ref=2.0.0"
}

include {
  path = find_in_parent_folders()
}

inputs = {

  vm_name                 = "sentry"
  num                     = 1
  cpu                     = 16
  memory                  = 32

}
```

Какой то странный код, согласись?=)

Terragrunt или когда Terraform и его модулей становится недостаточно

Terragrunt

Особенности:

01

DRY подход

02

Расширенная шаблонизация

03

Дополнительные типы переменных

04

Гибкое управление зависимостями

Terragrunt

Было

```
infra-dev
ifnra-prod
infra-uat
ifra-whatever
```

Стало

```
├── README.md --> Readme
├── dev --> dev env
├── prod --> prod env
├── provider.tmpl --> Terraform provider
template
├── terragrunt.hcl --> Terragrunt
configuration
└── uat --> uat env
```

Terragrunt

Terragrunt.hcl

```
locals {
  env_vars    = read_terragrunt_config(find_in_parent_folders("env.hcl"))
  group_vars = read_terragrunt_config(find_in_parent_folders("group.hcl"))
}

generate "provider" {
  path      = "provider.tf"
  if_exists = "overwrite"
  contents = templatefile("${get_parent_terragrunt_dir()}/provider.tmpl", {
    zone           = "ru-central1-a"
    folder_id      = local.env_vars.locals.folder_id
    consul_address = local.env_vars.locals.consul_address
  })
}

inputs = merge({
  subnet_id         = local.group_vars.locals.subnet_id
  network_id        = local.env_vars.locals.network_id
  folder_id         = local.env_vars.locals.folder_id

}
```

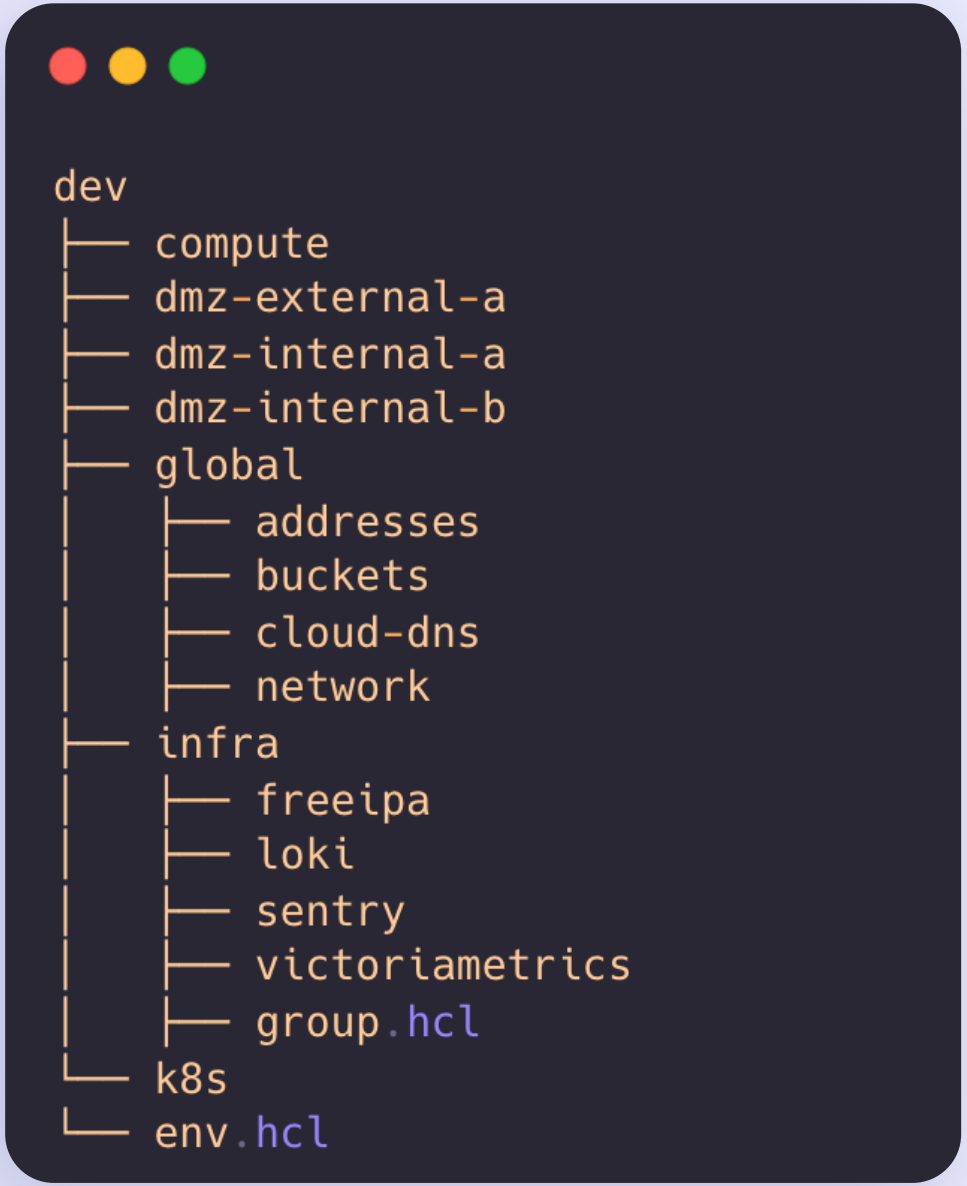
Terragrunt

Provider.tmpl

```
terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
    }
  }

  backend "http" {
    address        = "https://olololo-trololo/${replace(module_path,"/","-")}"
    lock_address   = "https://olololo-trololo/${replace(module_path,"/","-")}/lock"
    unlock_address = "https://olololo-trololo/${replace(module_path,"/","-")}/lock"
    username       = "gitlab-ci-token"
  }

}

provider "yandex" {
  cloud_id  = "super-duper-cloud-id"
  folder_id = "${folder_id}"
  zone      = "${zone}"
}
```

Terragrunt

Dev env



Terragrunt

Создание ресурсов при помощи terragrunt

```
terraform {
  source = "git@ololo-trololo/sentry.git//.?ref=2.0.0"
}

include {
  path = find_in_parent_folders()
}

inputs = {
}
```

Terragrunt

Зависимости ресурсов:

Terragrunt

Создание master node

```
terraform {
  source = "git@gitlab.com:ololo-org/devops/shared-
knowledge/tfmodules/ycloud/k8s_master.git//.?ref=1.3.0"
}

include {
  path = find_in_parent_folders()
}

dependency "sg" {
  config_path = "${get_repo_root()}/dev/global/security-groups/k8s/master"
}

inputs = {
  cluster_name              = "platform-dev"
  cilium                    = true
  version_k8s               = "1.24"
  zone                      = "ru-central1-a"
  vpc_folder_id             = "folde-id" # dev catalog
  security_group_ids = [
    dependency.sg.outputs.security_group_id
    ]

}
```

Terragrunt

Создание worker node

```
terraform {
  source = "git@gitlab.com:ololo-org/devops/shared-
knowledge/tfmodules/ycloud/k8s_node.git//.?ref=2.0.3"
}

include {
  path = find_in_parent_folders()
}

dependency "masters" {
  config_path = "../masters"
}

dependency "sg" {
  config_path = "${get_repo_root()}/dev/global/security-groups/k8s/worker"
}

inputs = {

  cluster_id  = dependency.masters.outputs.cluster_id
  pool_name   = "dev-zone-a-v2" # unique
  k8s_version = "1.24"
  nat         = false
  security_group_ids = [
    dependency.sg.outputs.security_group_id
    ]

}
```

Terragrunt

Особенности:

01

Отдельный стейт для каждого ресурса (не всем может быть удобно)

02

Terragrunt run all

03

Курица или яйцо

CD infra-pipeline

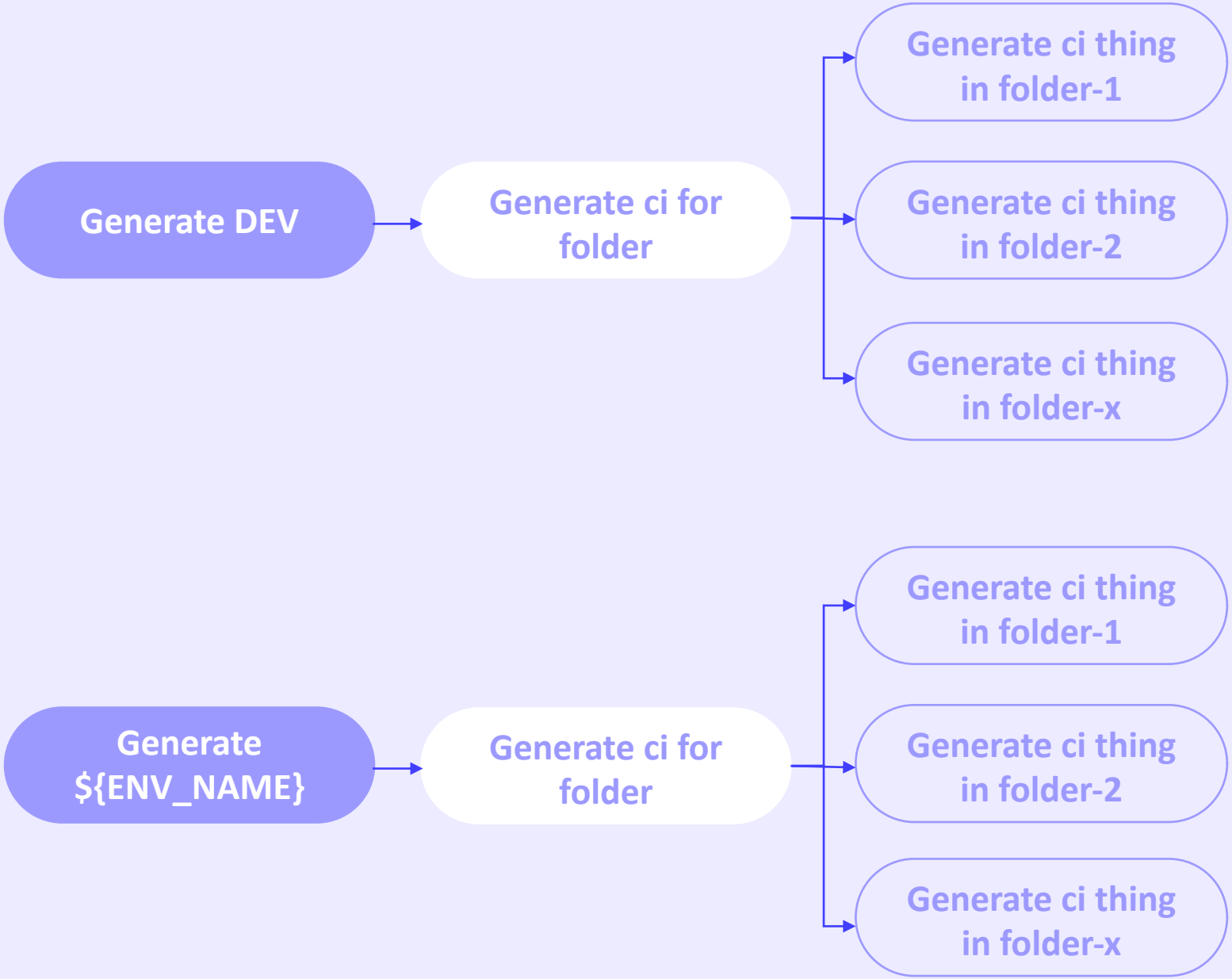
```
plan:
  stage: teplansts
  script:
    - terragrunt run-all plan

apply:
  stage: apply
  script:
    - terragrunt run-all apply -auto-approve

output:
  stage: output
  script:
    - terragrunt run-all output
```

CD infra-pipeline

Gitlab + Triggers + Downstream pipelines = 💓

CD infra-pipeline

Pipeline for everything



CD infra-pipeline

Stages for everything

```yaml
include:
  - '.gitlab-ci-terragrunt.yml'
  - '.gitlab-ci-templates.yml'
  - '.gitlab-ci-mr-jobs.yml'

generate:dev:
  stage: setup
  extends:
    - .runners:dev
  script:
    - ./generate-default-jobs.sh dev dev
  artifacts:
    paths:
    - .gitlab-ci-dev.yml
    expire_in: 1 week
  rules:
    - if: $CI_COMMIT_REF_PROTECTED == "true"

dev:
  stage: triggers
  trigger:
    include:
      - artifact: .gitlab-ci-dev.yml
        job: generate:dev
  rules:
    - if: $CI_COMMIT_REF_PROTECTED == "true"
```

CD infra-pipeline

Dev – triggered pipeline

```yaml
stages:
  - build
  - deploy

folder-name:plan:
  stage: build
  variables:
    TF_ROOT: "folder-path"
  extends:
    - .terragrunt:plan:folder-path
    - .rules:default:folder-path
    - .variables:protected

folder-name:apply:
  stage: deploy
  variables:
    TF_ROOT: "folder-path"
  extends:
    - .terragrunt:apply:folder-path
    - .variables:protected
  needs:
    - folder-name:plan
```

CD infra-pipeline

Magic script

```
#!/bin/sh
cat >.gitlab-ci-${parent_prefix}.yml <<EOF
include:
  - '.gitlab-ci-terragrunt.yml'
  - '.gitlab-ci-templates.yml'

stages:
  - build
  - deploy
  - setup
  - triggers

EOF

for root in ${has_terragrunt}
do
prefix=$(echo "$root" | tr / -)
basename=$(basename "$root")
cat >>.gitlab-ci-${parent_prefix}.yml <<EOF
${basename}:plan:
  stage: build
  variables:
    TF_ROOT: "${root}"
  extends:
    - .terragrunt:plan:${env}
    - .rules:default:${env}
    - .variables:protected

EOF
done
```

Хозяйке на заметку

- Следите за переменными
- Автоматизируйте свою работу
- Версионируйте все
- Переиспользуйте свои наработки, не повторяйтесь
- IaC - в моменте может быть не нужен

TECH

Планы на будущее

- Разработчики могут влиять на инфраструктуру
- Chat-OPS

Спасибо за внимание!

Дроздецкий Владимир

DevOps TeamLead

@Mrgreyves