THE SCALE FACTORY

Community **events**_

DEVOPSDAYS EDINBURGH

DEVOPSDAYS
LONDON 2019

https://devopsdays.org/

DEV OPS

DevOpsDays Ghent

Cloud bending using latest **Terraform**_

Marko Bevc

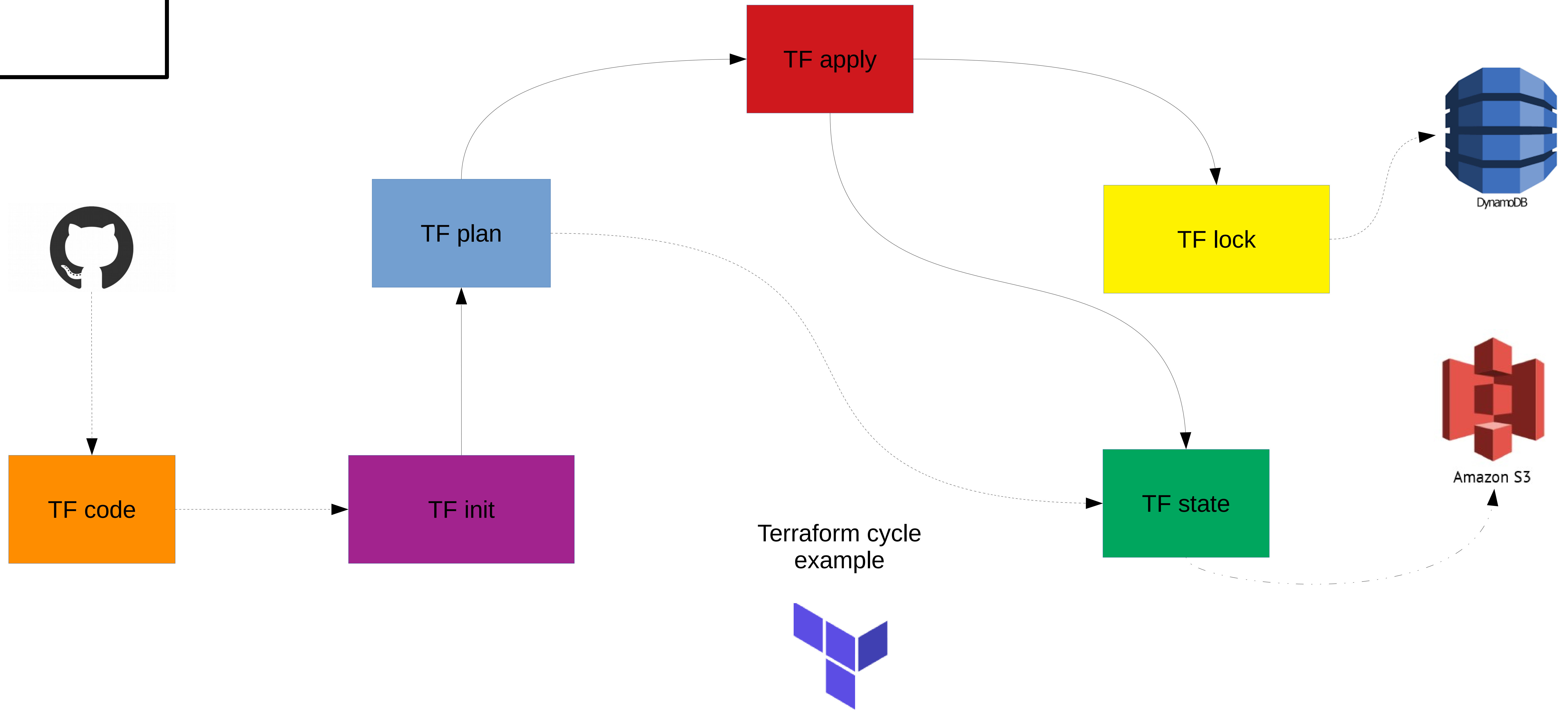
Today's **agenda**_

- TF overview / recap on adoption
- Changes with latest release
- Collaboration and teamwork
- Automation and CI integration
- Demos
- Conclusions

Terraform basics_

- Released in 2014 by Hashicorp
- Open source tool (Golang) for building, changing and versioning reproducible infrastructure
- Core: discovery, loading, RPC calls to plugins
- Plugins (providers & provisioners):
  - Expose implementation
  - Providers / services: AWS, GCP, ...
  - Provisioners: bash, archive_file

Terraform workflow_



Terraform **core_**

- IaaS: reading and interpolating configuration files and modules
- Communication with plugins over RPC
- Construction of the resource Graph
- Plan and execution
- Resource state management

```
ec2.tf:
--------
resource "aws_instance" "node" {
  count                       = var.node_count
  ami                         = var.ami_id
  instance_type               = var.instance_type
  vpc_security_group_ids      = [ aws_security_group.node_security_group.id ]
  subnet_id                   = var.vpc_subnet_i
  associate_public_ip_address = true
  monitoring                  = false
  key_name                    = var.ec2_keypair
  user_data                   = base64encode(data.template_file.node_userdata.rendered)

  lifecycle {
    create_before_destroy = true

    ignore_changes = [
      "ami",
      "key_name",
      "user_data",
    ]
  }

  tags = {
    Name          = "Node${count.index}"
    Puppet_Server = var.server_hostname
  }

  depends_on = ["null_resource.dep_arn"]
}
```

Terraform **plugins_**

- **terraform init**
- *~/.terraform.d/plugins* or *%APPDATA%\terraform.d\plugins*
- Version constraints – latest installed/match
- Provider:
  - Initialization of any included libraries used to make API calls
  - Authentication with the Infrastructure Provider
  - Define Resources that map to specific Services
- Provisioner:
  - Executing commands or scripts on the designated Resource after creation, or on destruction

```go
package main

import (
    "github.com/hashicorp/terraform/plugin"
)

func main() {
    plugin.Serve(new(MyPlugin))
}

---

func Provider() *schema.Provider {
    return &schema.Provider{
        ...
    }
}

func TestProvider(t *testing.T) {
    if err := Provider().(*schema.Provider).InternalValidate(); err != nil {
        t.Fatalf("err: %s", err)
    }
}

func resourceComputeAddress() *schema.Resource {
    return &schema.Resource {
        ...
    }
}
```

Latest Terraform 0.12 **improvements**_

- First-class expression syntax – less interpolation & list(“ “)
- Generalized type system – pass resource/module and rich value types
- Iteration constructs – for(_each), dynamic nested blocks
- Structural rendering of plans – no need for TFJSON!
- Context-rich error messages for debugging

```
module "network" {
  source = "./modules/network"
}

variable "subnet_numbers" {
  description = "List of 8-bit numbers of subnets of base_cidr_block that should be granted access."
  default = [1, 2, 3]
}

resource "aws_security_group" "example" {
  name        = "friendly_subnets"
  description = "Allows access from friendly subnets"
  vpc_id      = module.network.vpc.id

  ingress {
    from_port = 0
    to_port   = 0
    protocol  = -1

    cidr_blocks = [
      for num in var.subnet_numbers:
      cidrsubnet(data.aws_vpc.example.cidr_block, 8, num)
    ]
  }
}

output "instance_private_ip_addresses" {
  #  {"i-1234" = "192.168.1.2", "i-5678" = "192.168.1.5"}
  value = {
    for instance in aws_instance.example:
    instance.id => instance.private_ip
  }
}

output "instance_public_ip_addresses" {
  value = {
    for instance in aws_instance.example:
    instance.id => instance.public
    if instance.associate_public_ip_address
  }
}
```

```
locals {
  standard_tags = {
    Component   = "user-service"
    Environment = "production"
  }
}

resource "aws_autoscaling_group" "example" {
  # ...

  tag {
    key                 = "Name"
    value               = "example-asg-name"
    propagate_at_launch = false
  }

  dynamic "tag" {
    for_each = local.standard_tags

    content {
      key                 = tag.key
      value               = tag.value
      propagate_at_launch = true
    }
  }
}
```

```
$ terraform show -json terraform.tfplan |jq
{
  "format_version": "0.1",
  "terraform_version": "0.12.8",
  "variables": {
    "image": {
      "value": "mbevc1/demo-web:latest"
    },
    "name": {
      "value": "my-app"
    },
    "region": {
      "value": "eu-west-1"
    }
  },
  "planned_values": {
    "root_module": {
      "child_modules": [
        {
          "resources": [
            {
              "address": "module.alb.aws_alb.main",
              "mode": "managed",
              "type": "aws_alb",
              "name": "main",
              "provider_name": "aws",
              "schema_version": 0,
              "values": {
                "access_logs": [],
                "enable_cross_zone_load_balancing": null,
                "enable_deletion_protection": false,
                "enable_http2": true,
                "idle_timeout": 60,
                "load_balancer_type": "application",
                "name": "my-app-alb",
...
```

```
$ terraform plan -out terraform.tfplan

Error: Reference to undeclared output value

  on main.tf line 35, in module "alb":
  35:   security_group = module.security.lb_sgX

An output value with the name "lb_sgX" has not been declared in
module.security. Did you mean "lb_sg"?
```

Collaboration and **teamwork**_

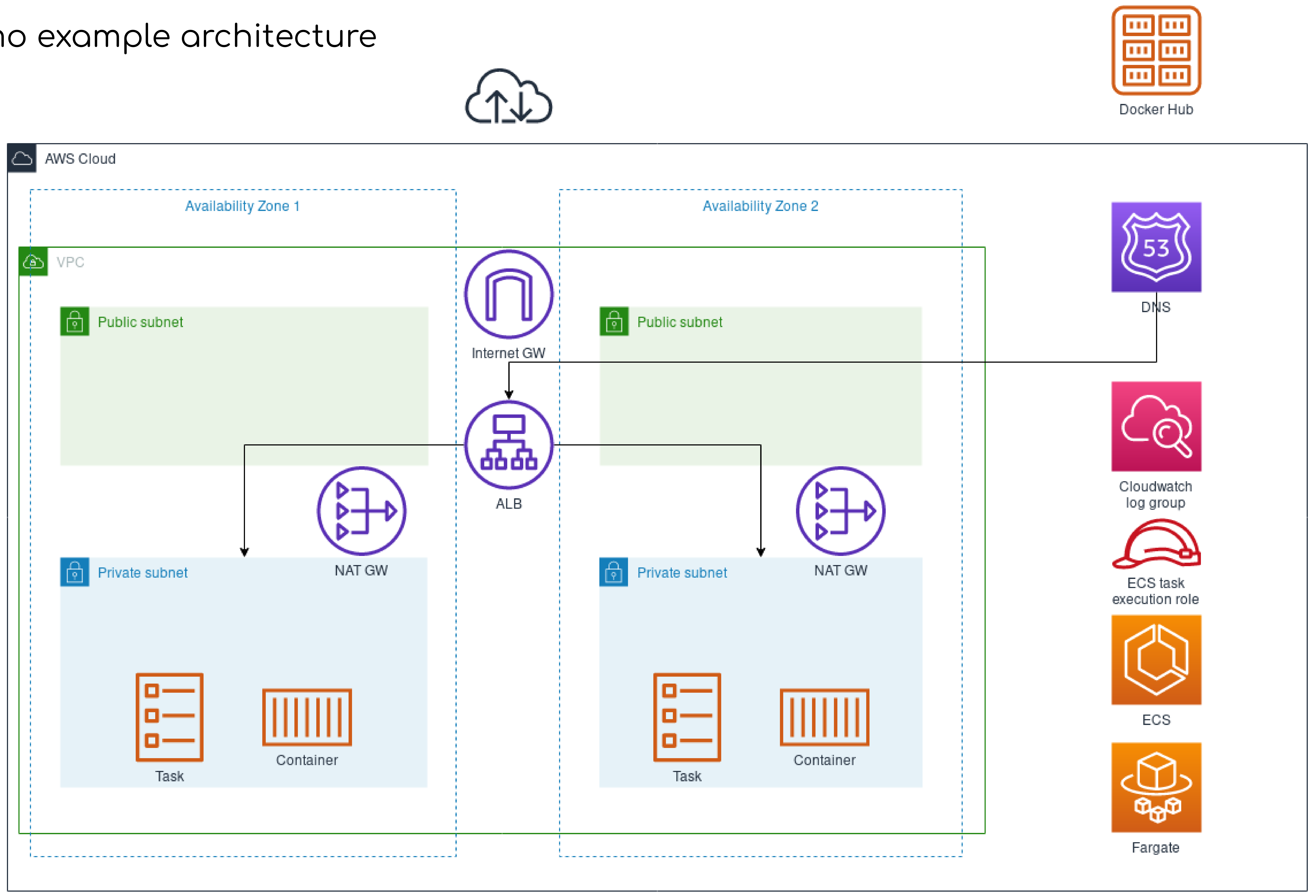
- Modules and registry
- Open source & plugins
- Workspaces – sharing configuration and backend – weak decomposition
- Remote state(JSON) backends and locking
- Importing resources
- Terraform Cloud: Full App UI, state management, VCS connections, notifications, webhooks, team collaboration and remote execution

Automation & integration_

- Validation, formatting (fmt), linting (tflint) and versions (tfenv), Kitchen Test suite, ...
- Suppressing output & reproducible plans
- Machine readable plans / states
- Wrappers and 3rd party tools - Terragrunt
- Programmatic code generation
- Secrets and state management; AWS secrets manager, Vault, encrypted storage
- CI integration – Atlantis, Terraform Cloud, ...

Time for
DEMO!_

Demo example architecture



Alternative **solutions_**

- 3rd party or manual tooling for API calls
- Native cloud tooling (CFN, GC deployment manager, …)
- CDKs: AWS, Google, Pulumi, …
- Serverless framework (serverless.com)

Conclusions_

& challenges

- Going multi-[cloud, service, platform] & platform shift
- Efficient way of describing reproducible infrastructure – IaaC
- Extensible (modules, registry, providers, ...)
- Change automation (JSON output, reproducible, CI integration)
- Upcoming features: *depends_on* for modules, module count, module and resource *for_each*
- Terraform Cloud – UI, remote state/execution, notifications, webhooks, VCS/CI
- Small gap comparing to native cloud tooling – API exposure
- SDK/CDKs – testable code
- Upgrade and 0.12 changes
- Terraform still ticks most of the boxes

- GitHub / GitLab: @mbevc1
- LinkedIn: https://www.linkedin.com/in/marko-bevc-245271118/

QUESTIONS?_

marko@scalefactory.com