Filling the gap between Terraform and Argo CD

Nathanael Liechti
Cloud Native Bern Meetup, 26th November 2025

Speaker

Nathanael Liechti
Kubernetes Platform Engineer / CS Student

hobbies: Cycling 🚴 & Biketouring 🏕️

@the-technat

Agenda

1. Why an Argo CD provider?
2. History of the provider
3. Use Case @ Swiss Post
4. Use Case : Registering Clusters to central Argo CD
5. Use Case : Creating API Tokens for service accounts
6. Use Case : Terraform Onboarding Module

Why an Argo CD Provider?

Why an Argo CD Provider?

Why an Argo CD Provider?

I thought Argo CD already allowed for 100% declarative configuration?

Wouldn't using the Kubernetes provider to handle Argo CD configuration be enough?

Benefits using the Argo CD Provider

Cluster-independent (you don’t need access to the Kubernetes API to configure Argo CD)

```
provider "argocd" {
  server_addr = "argocd.local:443"
  auth_token  = "1234..."
}

provider "argocd" {
  use_local_config = true
}

provider "argocd" {
  auth_token   = "1234..."
  port_forward = true
}

provider "argocd" {
  core = true
}
```

Benefits using the Argo CD Provider

```
▶ kubectl api-resources |grep ar
applications
applicationsets
appprojects
```

```
true          Application
true          ApplicationSet
true          AppProject
```

Atomic configuration

Sample File	Resource Name	Kind	Description
argocd-cm.yaml	argocd-cm	ConfigMap	General Argo CD configuration
argocd-repositories.yaml	my-private-repo / istio-helm-repo / private-helm-repo / private-repo	Secrets	Sample repository connection details
argocd-secret.yaml	argocd-secret	Secret	User Passwords, Certificates (deprecated), Signing Key, Dex secrets, Webhook secrets
argocd-rbac-cm.yaml	argocd-rbac-cm	ConfigMap	RBAC Configuration
argocd-tls-certs-cm.yaml	argocd-tls-certs-cm	ConfigMap	Custom TLS certificates for connecting Git repositories via HTTPS (v1.2 and later)
argocd-ssh-known-hosts-cm.yaml	argocd-ssh-known-hosts-cm	ConfigMap	SSH known hosts data for connecting Git repositories via SSH (v1.2 and later)

Argo CD - Declarative GitOps CD for Kubernetes | Declarative Setup

Benefits using the Argo CD Provider

```yaml
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: meetup
  namespace: argocd
spec:
  descrip
  destina
  - names
    serve
  roles:
  - name:
    polic
    - p, p
  sourceNa
  - argoc
  sourceRe
  - '*'
```

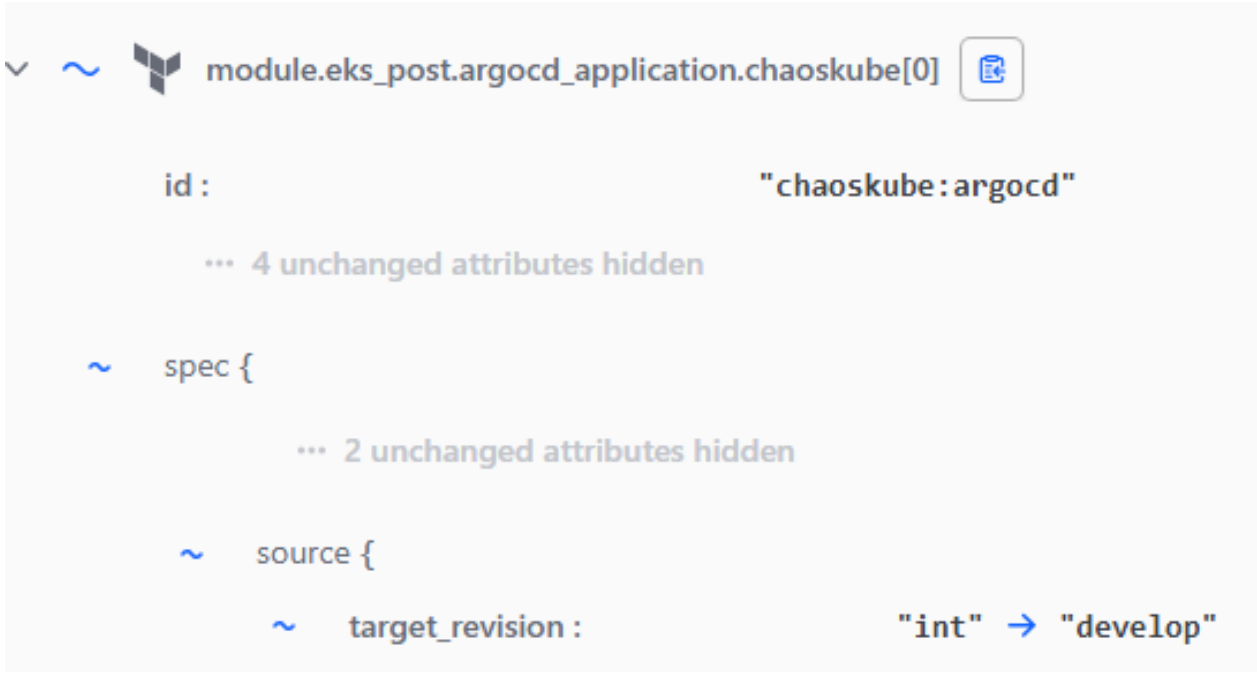
```
resource "kubernetes_manifest" "test-crd" {
  manifest = yamldecode(file("${path.module}/argocd_app.yaml"))
}
```

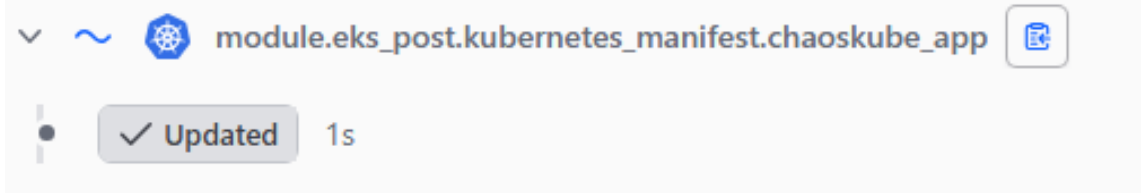
```
resource "argocd_project" "meetup" {
  metadata {
    name      = "meetup"
    namespace = "argocd"
  }

  spec {
    description  = "simple project"

          "p, proj:someotherproject:testrole, applications, sync, meetup/*, allow",
        ]
      }
    }
  }
```

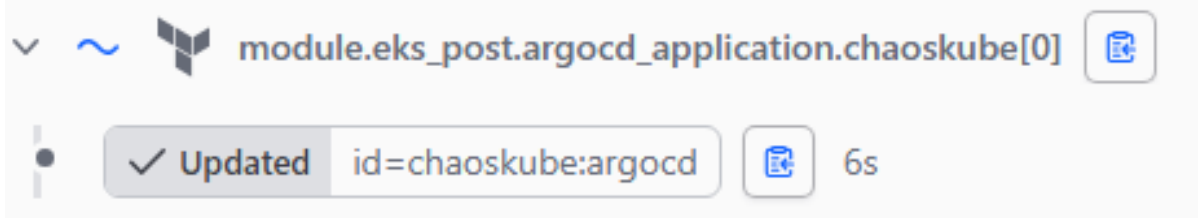
```
Plan: 1 to add, 0 to change, 0 to destroy.
kubernetes_manifest.test-crd: Creating ...
kubernetes_manifest.test-crd: Creation complete after 0s

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

```
Error: failed to expand project

  with argocd_project.meetup,
  on main.tf line 16, in resource "argocd_project" "meetup":
  16: resource "argocd_project" "meetup" {

invalid policy rule 'p, proj:someotherproject:testrole, applications, sync, meetup/*,
allow': policy subject must be: 'proj:meetup:testrole', not
'proj:someotherproject:testrole'
```

Benefits using the Argo CD Provider







History of the Provider

Beginnings of the Provider

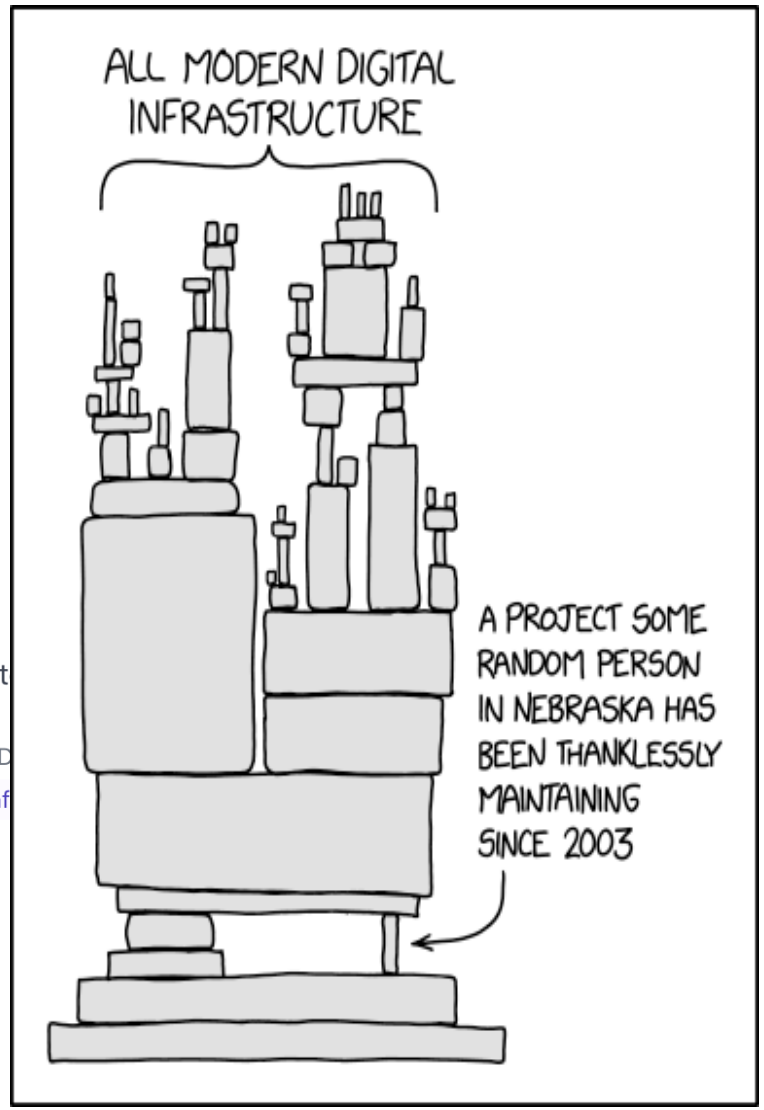
argocd

by:oboukili

Continuous Integration/Deployment (CI/CD)

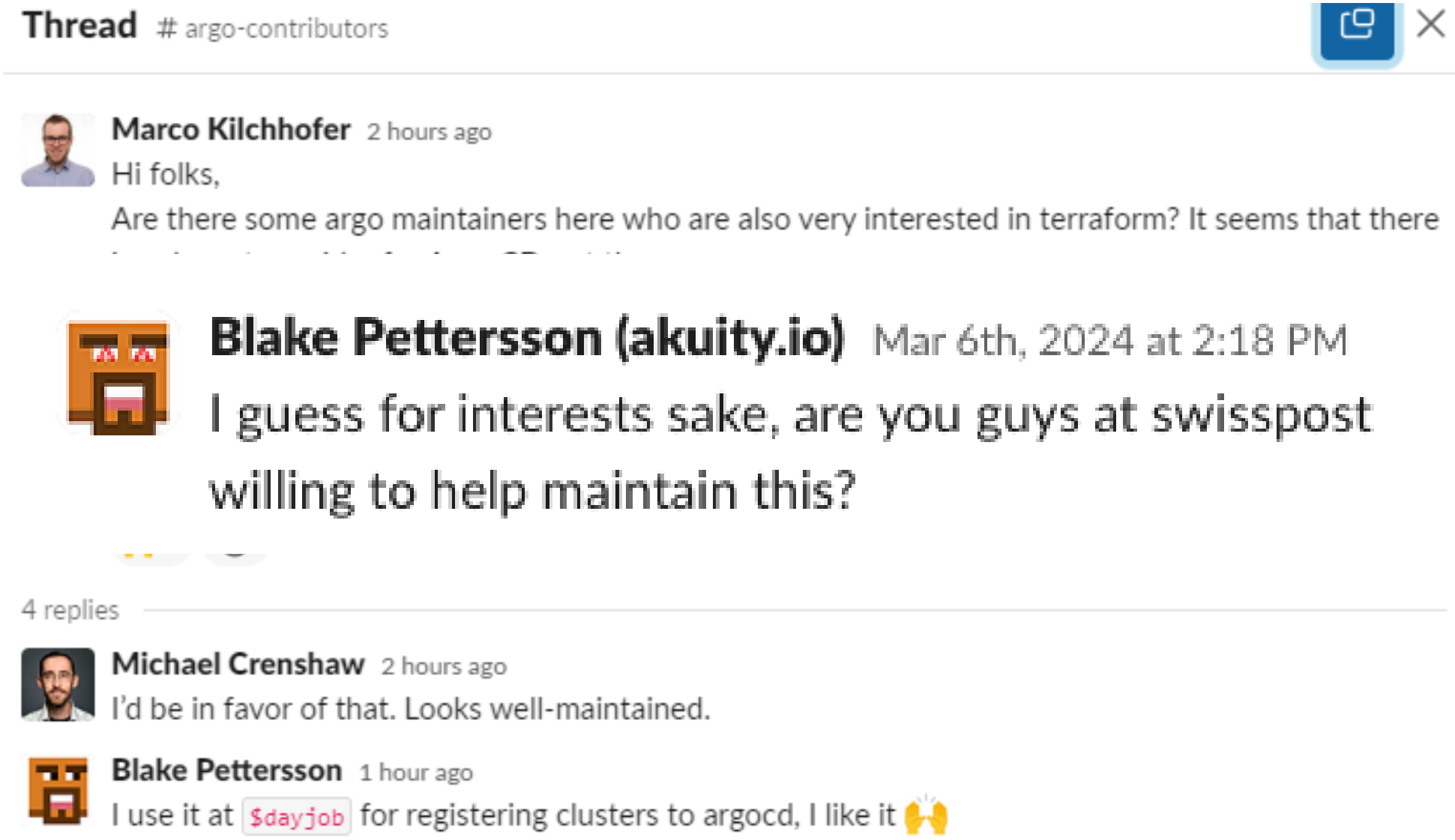
This Provider was migrated to - https://registry.t

VERSION	PUBLISHED	SOURCE COD
6.2.0	a year ago	oboukili/terraf



Provider Downloads	All versions
Downloads this week	41,551
Downloads this month	229,720
Downloads this year	3.1M
Downloads over all time	11.6M

Let’s make it official



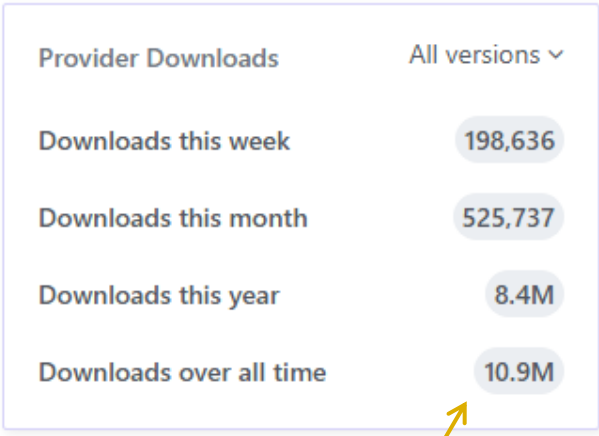
Official Argo CD Provider!

argocd

by:argoproj-labs

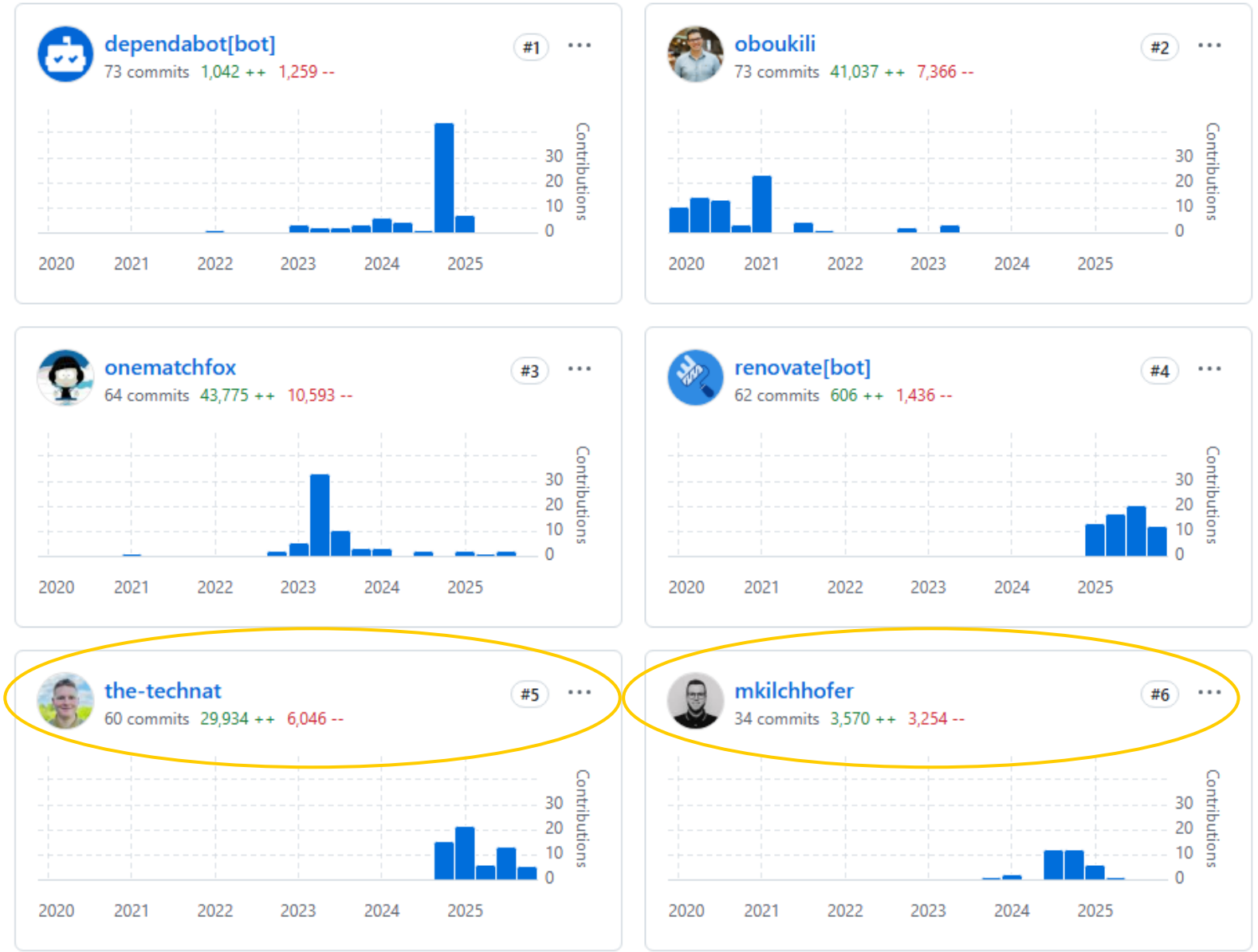
Continuous Integration/Deployment (CI/CD)

VERSION	PUBLISHED	SOURCE CODE
7.12.0	**a day ago**	argoproj-labs/terraform-provider-argocd



(since April 2024)

How does maintaining go?



Use Case @ Swiss Post

K8s@Swiss Post factoids

We strive for 100% automation

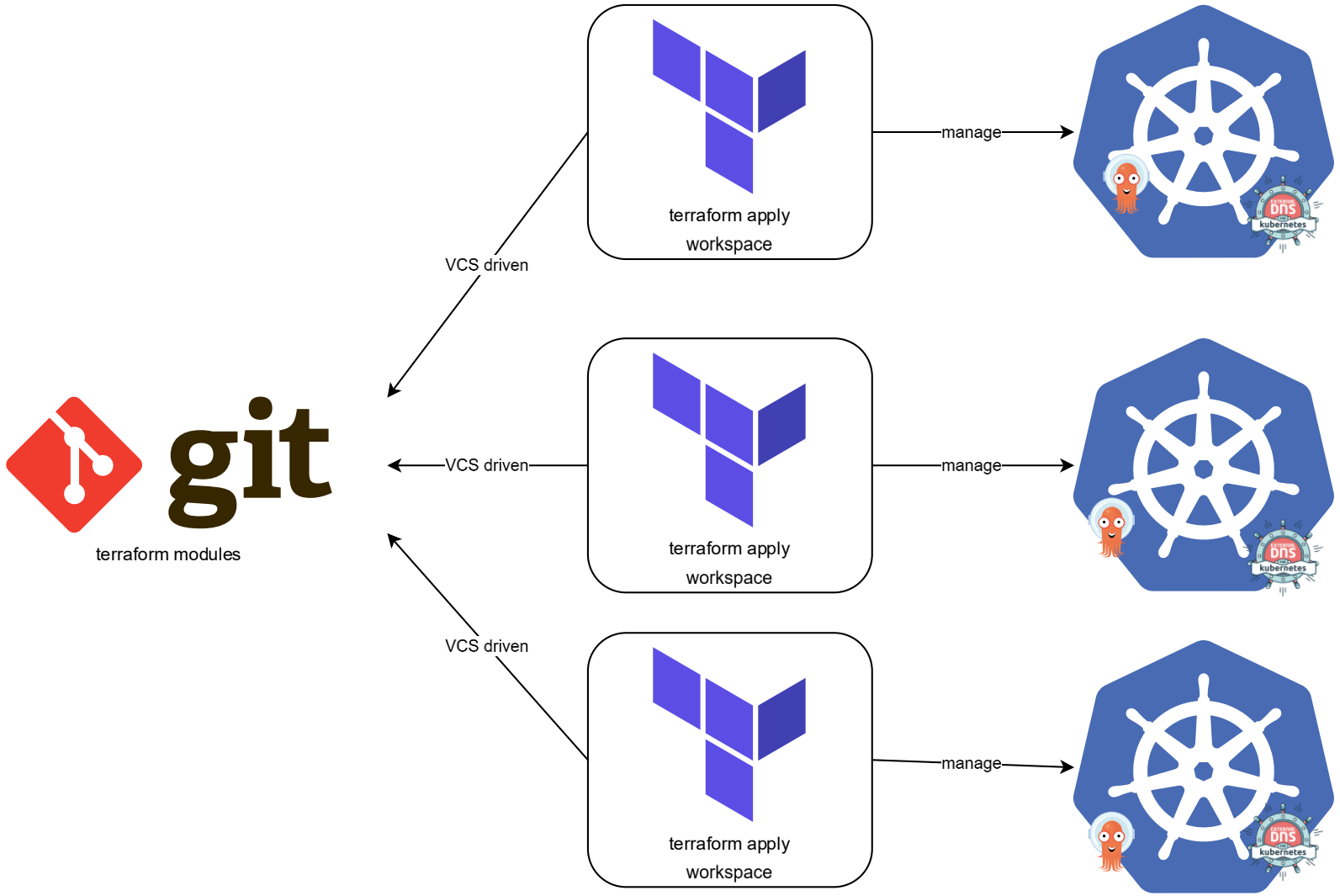
We have shared clusters per business unit

Our clusters can be managed independently

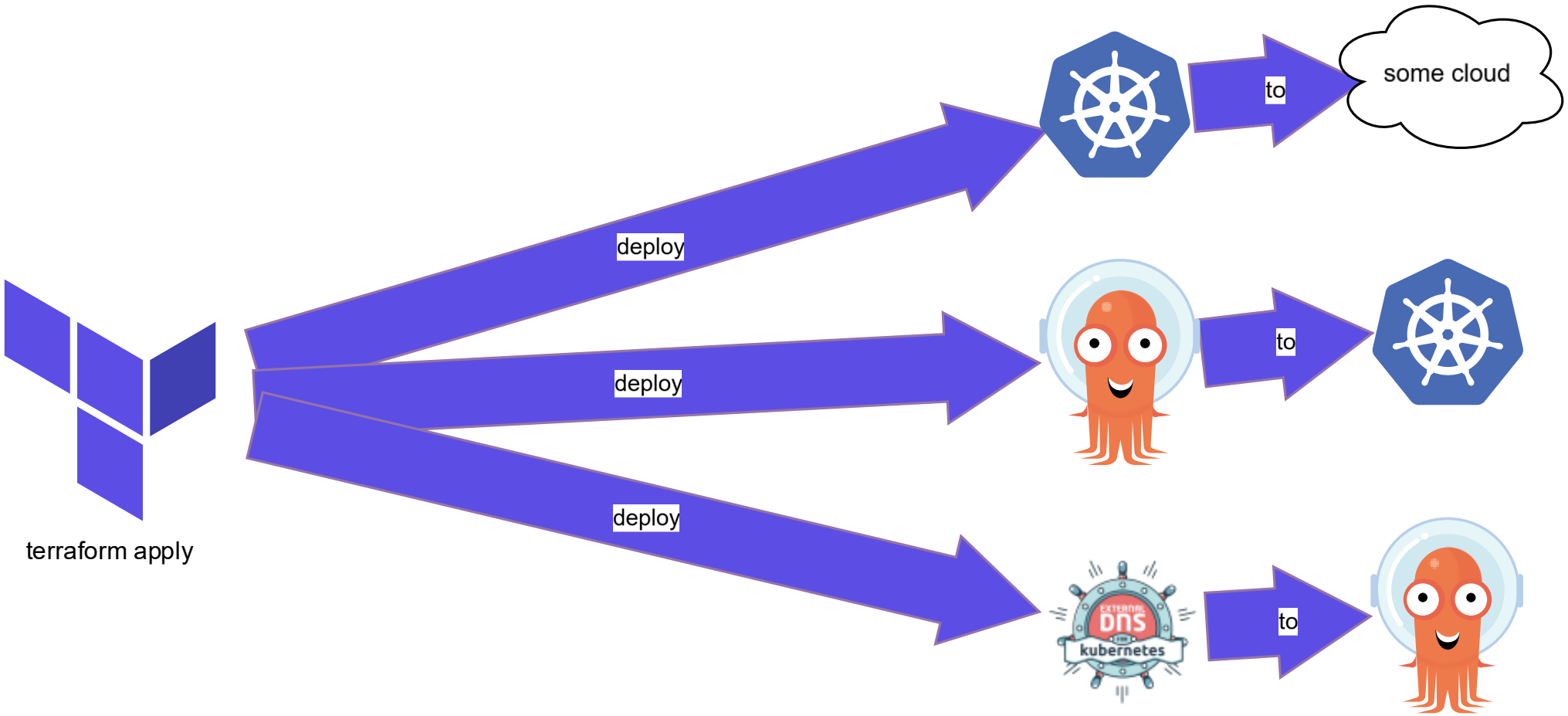
We manage everything using Terraform and deploy infrastructure addons using Argo CD

All our clusters should be configured the same with only a small degree of customization

Swiss Post's Deployment Model



Deploying a cluster in one run



Install EKS Cluster using AWS Provider

```
module “eks” {
  source  = "terraform-aws-modules/eks/aws"
  version = "~> 21.0"

  name                = “meetup_cluster"
  kubernetes_version = "1.33"

  compute_config = {
    enabled    = true
    node_pools = ["general-purpose"]
  }

  vpc_id     = "vpc-1234556abcdef"
  subnet_ids = ["subnet-abcde012", "subnet-bcde012a", "subnet-fghi345a"]
}
```

Install Terraform using Helm Provider

```
resource "helm_release" "argocd" {
  name       = "argocd"
  repository = “oci://ghcr.io/argoproj/argo-helm/"
  chart      = "argo-cd"
  version    = “9.1.3"
  namespace  = “argocd”
  wait       = true

  set {
    name  =  “configs.secret.argocdServerAdminPassword”
    value = bcrypt_hash.argocd_password.id
  }

  depends_on = [
    module.eks_cluster
  ]
}
```

Helm Provider Declaration

```hcl
provider "helm" {
  kubernetes {
    host                   = module.eks.cluster_endpoint
    cluster_ca_certificate = base64decode(module.eks.cluster_certificate_authority_data)

    exec {
      api_version = "client.authentication.k8s.io/v1beta1"
      command     = "aws"
      args = ["eks", "get-token", "--cluster-name", module.eks.cluster_name, "--output", "json"]
    }
  }
}
```

Create Argo CD Apps

```
resource "argocd_application" "external_dns" {
  metadata {
    name      = “external-dns”
    namespace = “argocd”
  }
  wait = true
  spec {
    project = "swisspost-addons"
    source {
      path            = "helm"
      repo_url        = “https://(...).git”
      target_revision = int
      helm { value_files = ["values.yaml", “EKS-values.yaml"] }
    }
    destination {
      server    = "https://kubernetes.default.svc"
      namespace = kubernetes_namespace.external_dns[0].metadata[0].name
    }
  }
  depends_on = [helm_release.argocd]
}
```

Argo CD Provider Declaration

```hcl
provider "argocd" {
  username = "admin"
  password = random_password.argocd_password.result

  port_forward_with_namespace = "argocd"

  kubernetes {
    host                   = module.eks.cluster_endpoint
    cluster_ca_certificate = base64decode(module.eks.cluster_certificate_authority_data)

    exec {
      api_version = "client.authentication.k8s.io/v1beta1"
      command     = "aws"
      args = ["eks", "get-token", "--cluster-name", module.eks.cluster_name, "--output", "json"]
    }
  }
}
```

Important Facts about this deployment model

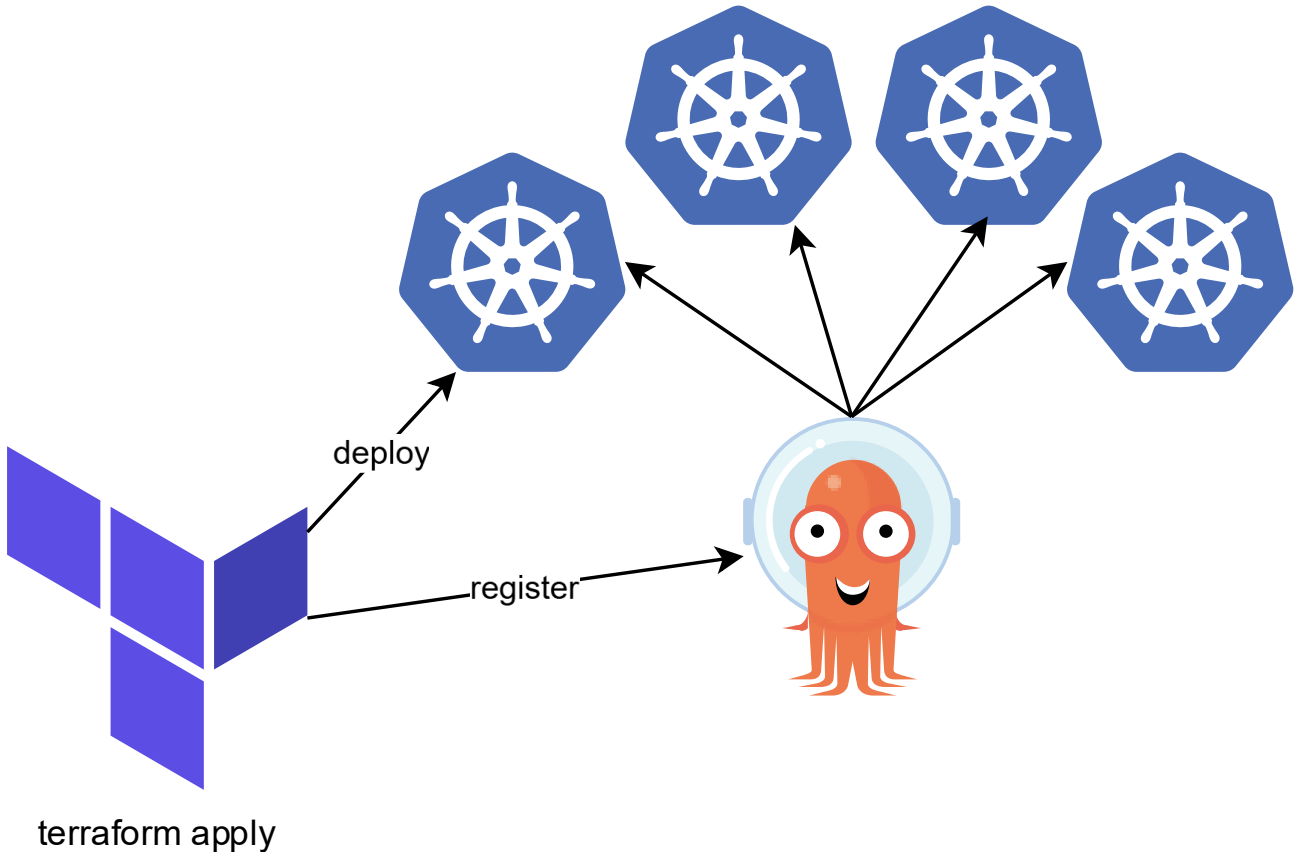
Don’t update the control-plane and helm/k8s things in the same run

Create and Delete works tough

> **WARNING**
>
> When using interpolation to pass credentials to the Kubernetes provider from other resources, these resources SHOULD NOT be created in the same Terraform module where Kubernetes provider resources are also used. This will lead to intermittent and unpredictable errors which are hard to debug and diagnose. The root issue lies with the order in which Terraform itself evaluates the provider blocks vs. actual resources. Please refer to this section of Terraform docs for further explanation.

Use Case: Registering clusters in Argo CD

Use Case: Registering Clusters in Argo CD

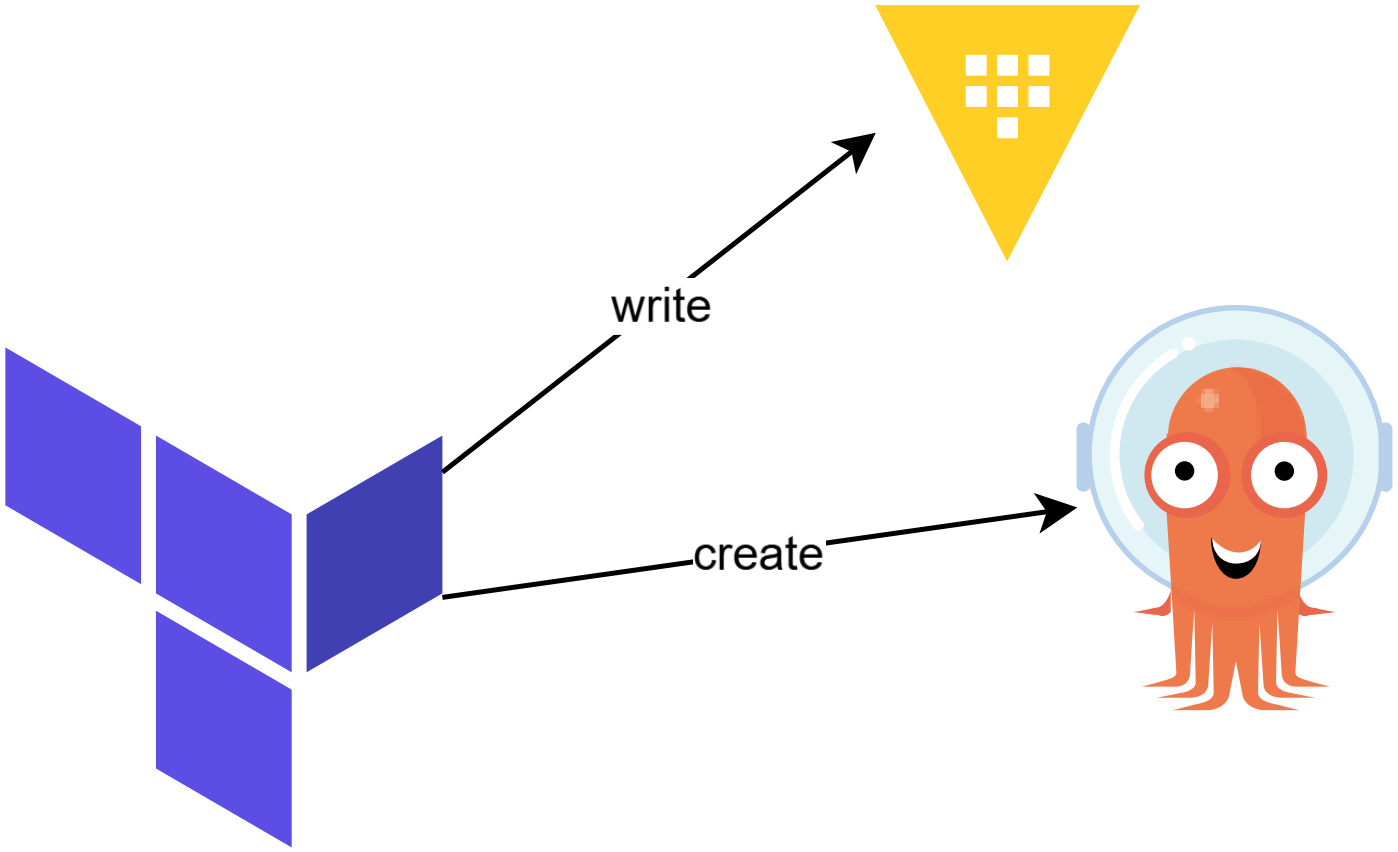


Use Case: Registering Clusters in Argo CD

```
resource "argocd_cluster" "eks" {
  server     = format("https://%s", module.eks.cluster_endpoint)
  name       = “meetup_cluster"
  namespaces = ["default", "optional"]

  config {
    aws_auth_config {
      cluster_name = “meetup_cluster"
      role_arn     = "arn:aws:iam::<123456789012>:role/<role-name>"
    }
    tls_client_config {
      ca_data = base64decode(module.eks.cluster_certificate_authority_data)
    }
  }
}
```

Use Case: Generate project/account tokens

Use Case: Generating tokens

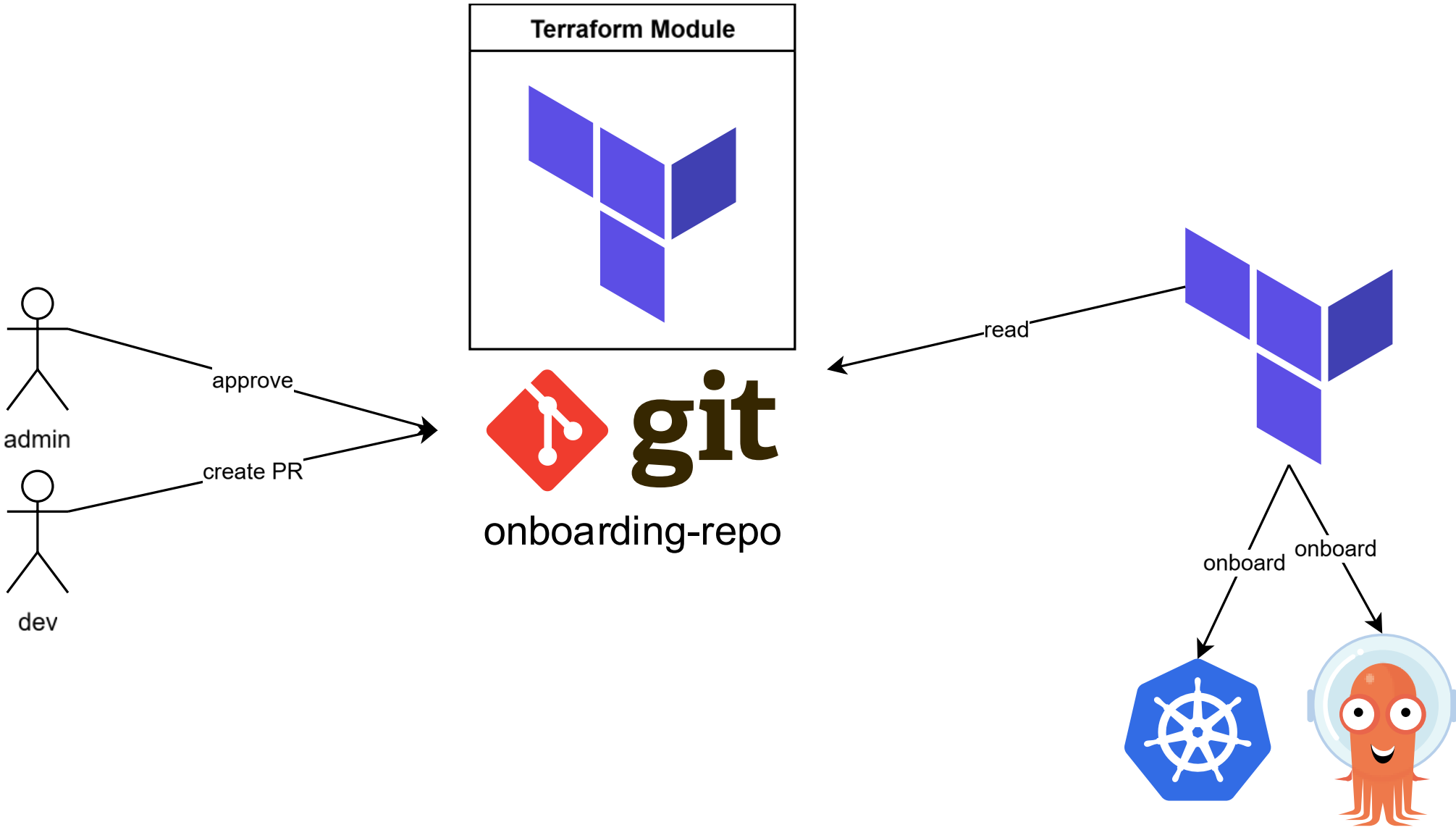


Use Case: Generating tokens

```
resource "argocd_project_token" "secret" {
  project      = "someproject"
  role         = "foobar"
  description  = "short lived token"
  expires_in   = "1h"
  renew_before = "30m"
}

resource "vault_kv_secret_v2" "example" {
  mount     = vault_mount.kvv2.path
  name      = “argocd_token"
  data_json = jsonencode({ “token” = argocd_project_token.secret.jwt })
}

```

Use Case: Terraform Onboarding

Use Case: Terraform Onboarding Module



Talk Recommendation: Terraform Onboarding Module

https://www.youtube.com/watch?v=2oJAZpLAPtg

Summary

The UX is better using the Argo CD Terraform Provider

Use Cases are very specific and narrow

OSS is hard

Danke, Merci,
Grazie, Thank you