

So You Want to Build a Platform?

You probably already have one!

Juho Syrjänen | ŌURA



$ whoami

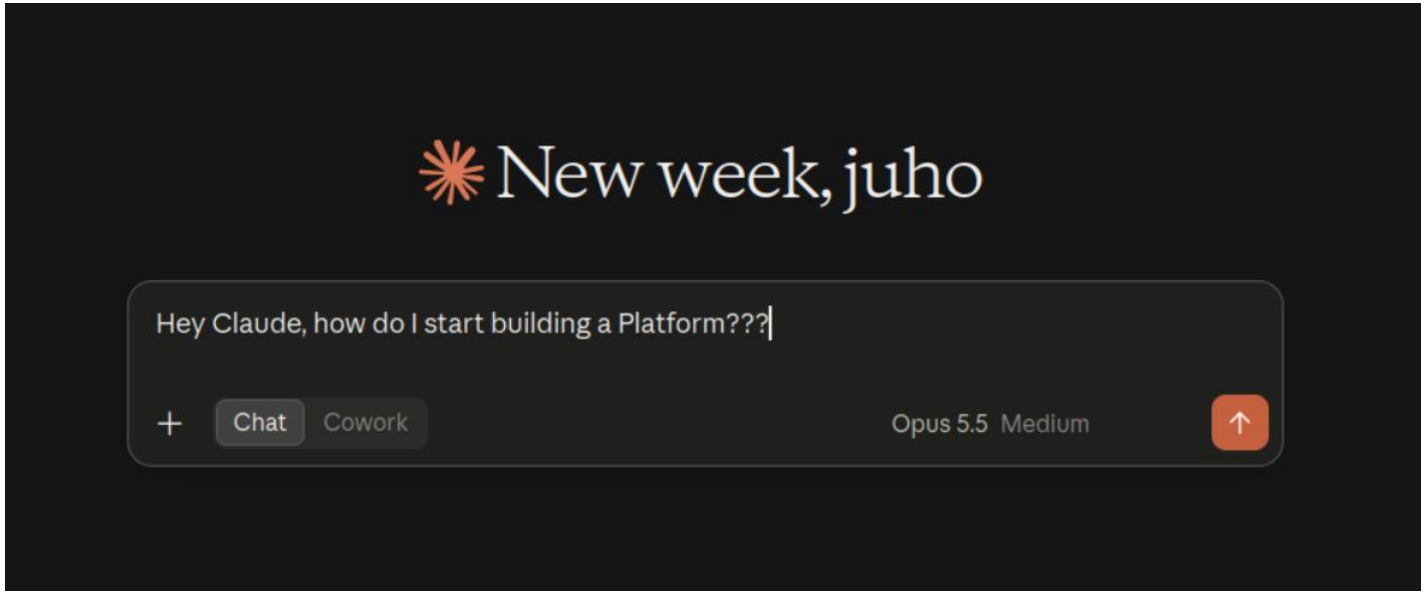


Juho Syrjänen 👋

Senior Platform Engineer @ ŌURA 💍

CNCF Platform Engineering UG Core Contributor

So you want to build a platform?


New week, juho
Hey Claude, how do I start building a Platform???
Chat
Cowork
Opus 5.5 Medium

CNCF Platforms White Paper

v1.1 (latest)

Download PDF

Version info

Introduction

Platform engineering has emerged as a necessary optimization for enterprises that experienced the value of autonomous teams with the emergence of DevOps, but want to reduce the costs, security risks, and inefficiencies that came with that autonomy. Platforms curate and present foundational capabilities, frameworks, and experiences to facilitate and accelerate the work of internal customers such as application developers, data scientists, and information workers. Most importantly, platforms help enterprises realize the cloud native values of fast product releases, portability across infrastructures, and greater developer productivity across their entire technology estate. Platforms achieve this by codifying what is unique to their business but common across their internal teams, creating an internal economy of scale that their software teams can leverage.

This paper intend
computing. We
sustainability
implement p

ort enterprise leaders, enterprise architects and platform team leaders to advocate for, investigate and plan internal platforms for cloud
ignificantly impact enterprises' actual value streams, but only indirectly, so leadership consensus and support is vital to the long-te
. In this paper we'll enable that support by discussing what the value of platforms is, how to measure that value, and h

https://cloudnativeplatforms.com/whitepapers/platforms/

Backstage

Platform Engineering Maturity Model

Presented by the Cloud Native Computing Foundation Working Group Platforms

Version 1.0.0
Released October 31st, 2023

https://cloudnativeplatforms.com/whitepapers/platform-eng-maturity-model/

Measure from the start. Track DORA metrics,

A modern and sovereign Private Cloud «

Communication
Platform Engineering
Service-Delivery
Telecommunication
Cloud_native
Software
Cloud
Private_cloud
Kubernetes
Switzerland

Crossplane

vice» for Swiss-based e

Enabling Allianz Direct's scaling through Platform Engineering

Financial
Platform Engineering
Finance
Insurance
Insurance
Europe
Public_cloud
Aws

End-to-End Cloud Native Telco Platform Auto

Communication
Telco
Cloud_native
Telecommunication
Platform Engineering
Telco
5g
Gitops
Kubernetes
Private_cloud

t Swisscom

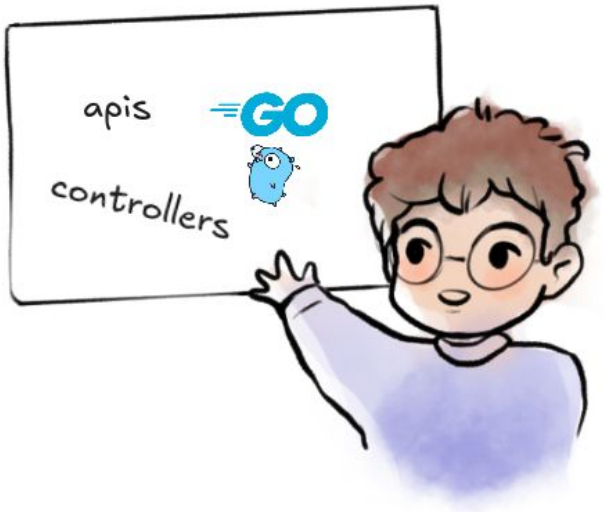
https://architectur

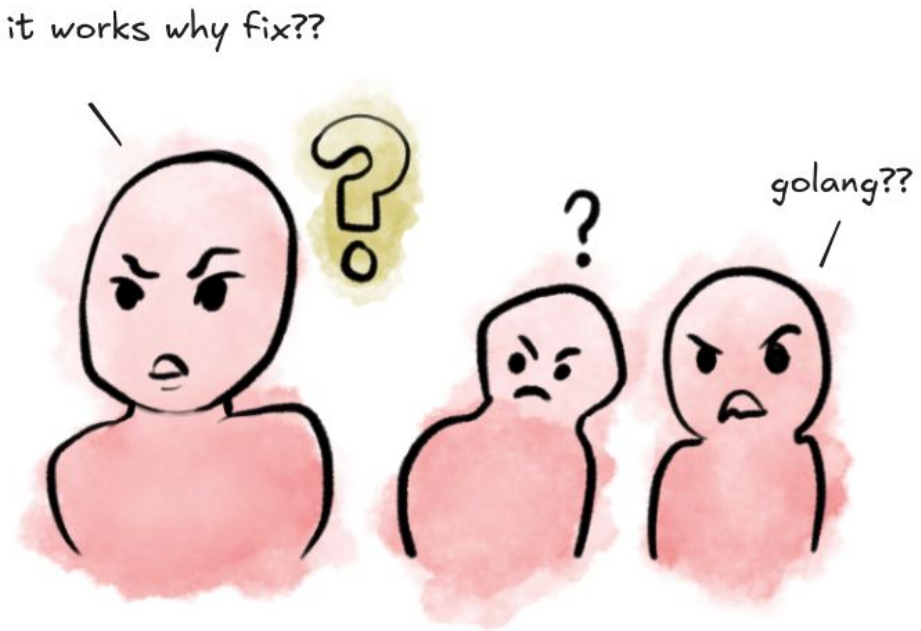
erence_architectures/platform-engineering/

Build the Thinnest Viable Platform first. In Team Topologies terms,

KubeCon CloudNativeCon
North America 2026


let's build this!
CNCF Platforms White Paper
Introduction
Platform Engineering Maturity Model
Presented by the Cloud Native Computing Foundation Working Group: Platforms
Backstage
Measure from the start. Track DORA metrics,
Crossplane
Build the Thinnest Viable Platform first, to Team Topologies terms


apis
GO
controllers


it works why fix??
golang??

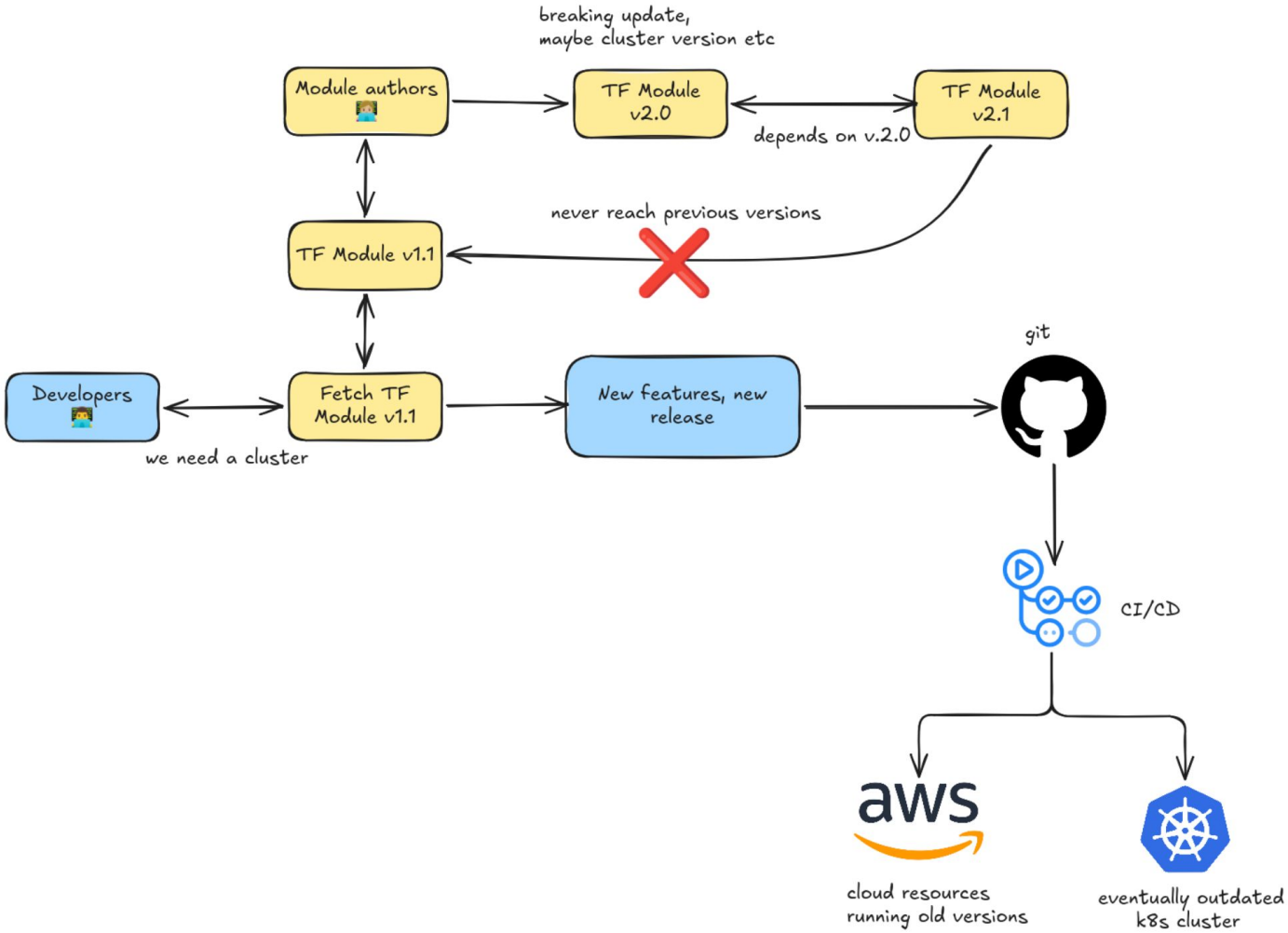


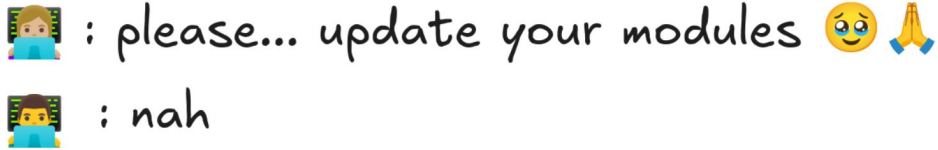
back to the trenches

Platform Engineering ≠ Build it again

Platform Engineering = Evolve existing

A central infra team cannot scale linearly with increased demand


breaking update,
maybe cluster version etc
Module authors
TF Module v2.0
TF Module v2.1
depends on v.2.0
never reach previous versions
TF Module v1.1
Developers
Fetch TF Module v1.1
we need a cluster
New features, new release
git
CI/CD
aws
cloud resources
running old versions
eventually outdated
k8s cluster


: please... update your modules
: nah


: update the module.. or else 🔪

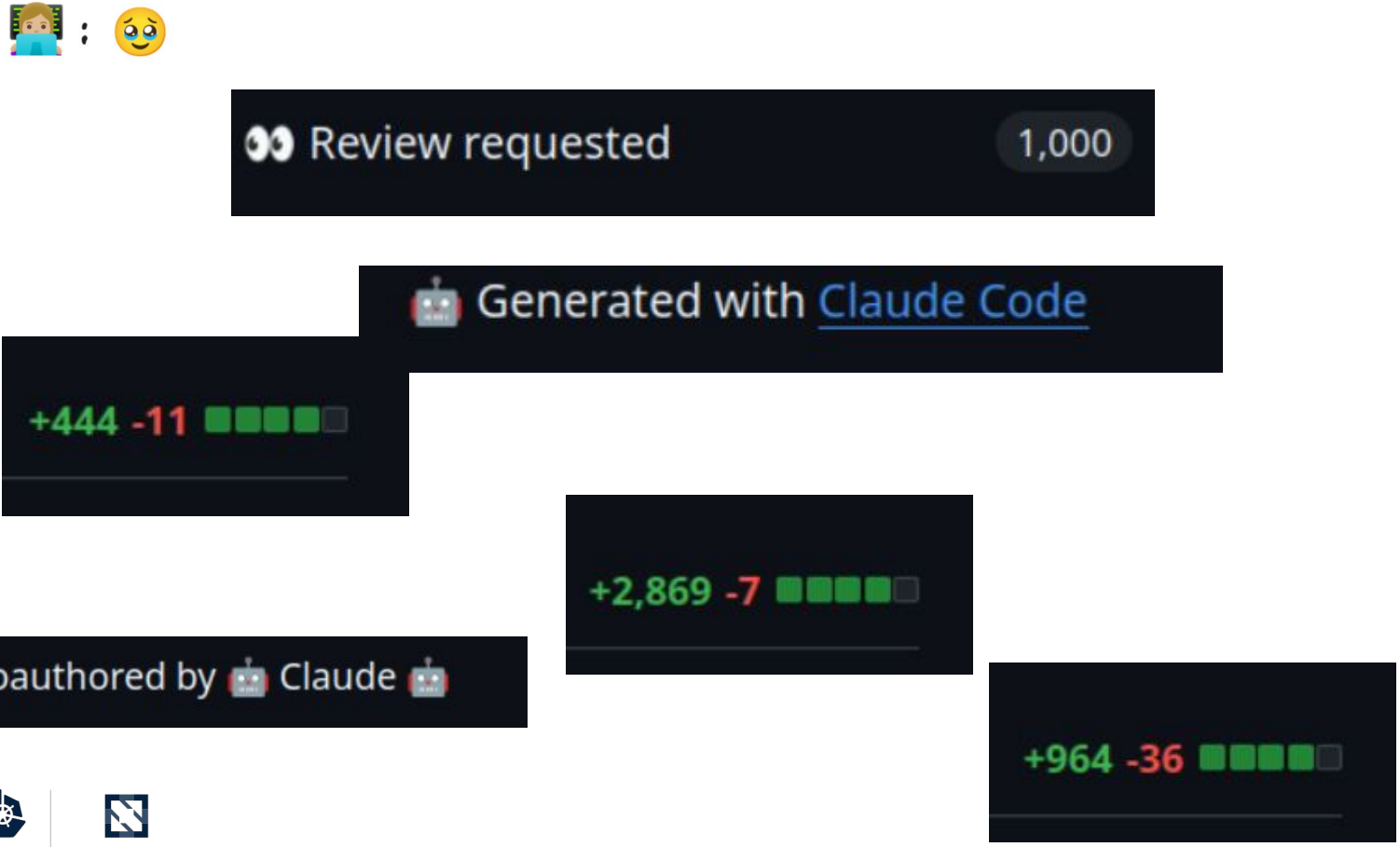

: fine

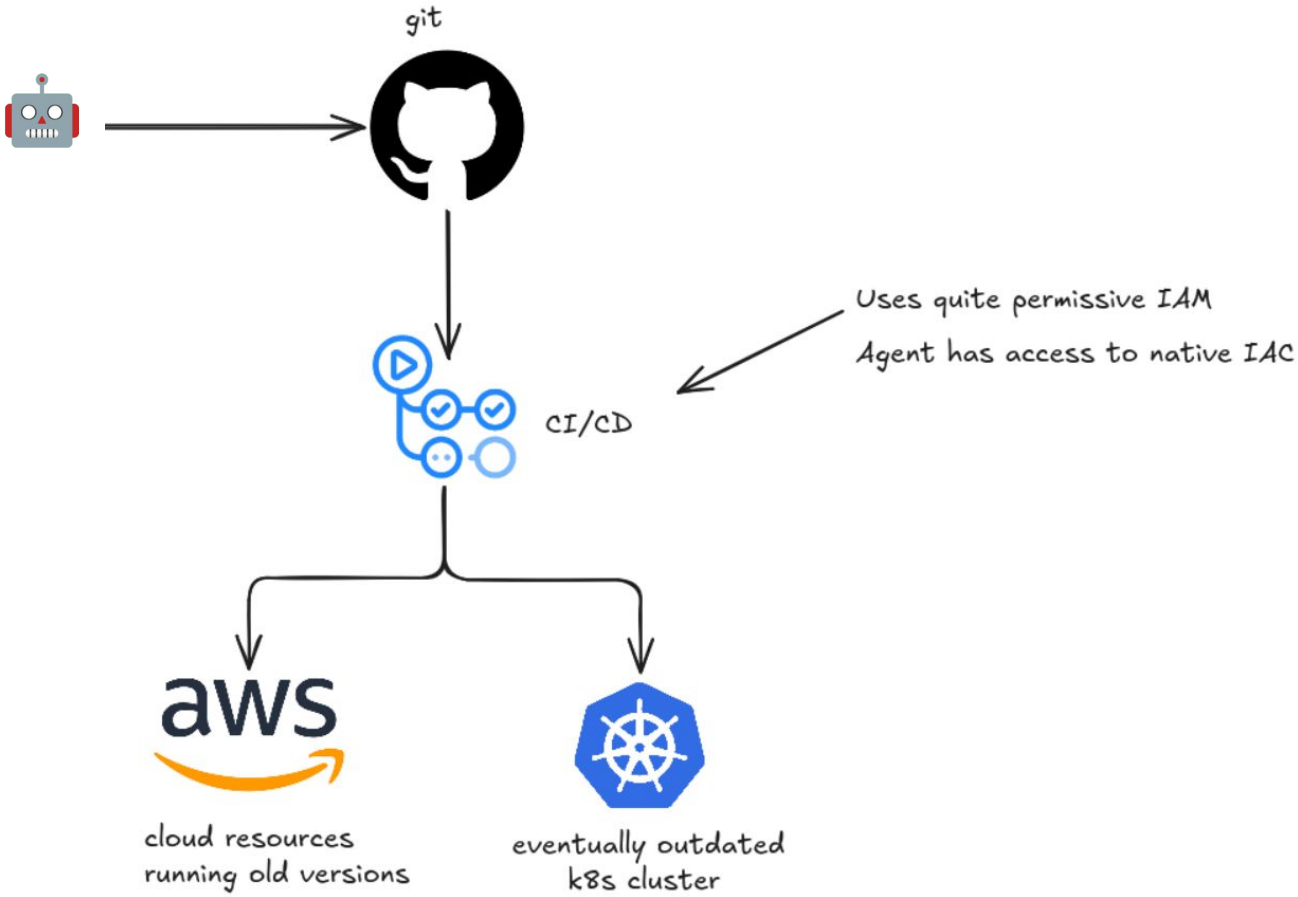


```
juho: code ❯ claude
 ▐▛███▜▌   Claude Code v2.1.289
▝▜█████▛▘  Sonnet 5.5 · Claude Enterprise
  ▘▘ ▝▝    ~/code

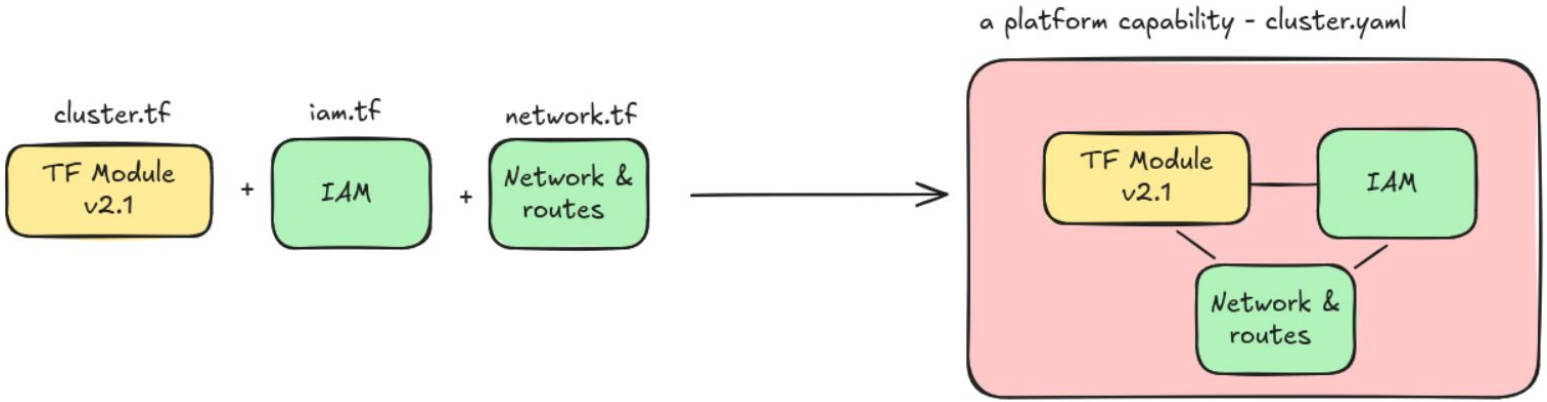
❯ update terraform modules, make no mistakes, open pr thx /yolo-mode

  Sonnet 5.5·xhigh  📁code
  ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓  0% of 200K · 200k left  $0.00 · ⏱ 0m0s  0k↓ 0k↑  ⎈dev
  ⏵⏵ accept edits on (shift+tab to cycle)
```


👩‍💻 ; 🥹
👀 Review requested 1,000
🤖 Generated with Claude Code
+444 -11
+2,869 -7
Coauthored by 🤖 Claude 🤖
+964 -36


git
Uses quite permissive IAM
Agent has access to native IAC
CI/CD
aws
cloud resources
running old versions
eventually outdated
k8s cluster

Could we make this better for everyone involved?


cluster.tf
TF Module v2.1
+
iam.tf
IAM
+
network.tf
Network & routes
a platform capability - cluster.yaml
TF Module v2.1
IAM
Network & routes



```
  cals {
    env_config = merge(
      yamldecode(file("${path.module}/envs/_defaults.yaml")),
      yamldecode(file("${path.module}/envs/${var.environment}.yaml")),
    )
    cluster_name = "${var.team}-cluster-${var.region_short}"
    azs          = slice(data.aws_availability_zones.this.names, 0, local.env_config.az_count)

    private_subnets = [for i, az in local.azs : cidrsubnet(var.vpc_cidr, 4, i)]
    public_subnets  = [for i, az in local.azs : cidrsubnet(var.vpc_cidr, 8, i + 48)]

    enabled_addons = {
      for name, cfg in local.env_config.addons : name => cfg
      if try(var.components[name], cfg.default_enabled)
    }

    common_tags = merge(var.tags, {
      "platform.example.com/team"        = var.team
      "platform.example.com/environment" = var.environment
      "karpenter.sh/discovery"           = local.cluster_name
    })
  }

  module "vpc" {
    source          = "terraform-aws-modules/vpc/aws"
    name            = local.cluster_name
    cidr            = var.vpc_cidr
    azs             = local.azs
    private_subnets = local.private_subnets
    public_subnets  = local.public_subnets

    single_nat_gateway = var.environment != "prod"
    private_subnet_tags = merge(local.common_tags, {
      "kubernetes.io/role/internal-elb" = "1"
    })
    # ... flow logs, endpoints, NACLs
  }

  module "eks" {
    source                         = "terraform-aws-modules/eks/aws"
    cluster_name                   = local.cluster_name
    cluster_version                = local.env_config.k8s_version
    vpc_id                         = module.vpc.vpc_id
    subnet_ids                     = module.vpc.private_subnets
    bootstrap_self_managed_addons = !try(var.components.cilium, false)

    access_entries = {
      for role in var.admin_role_arns : basename(role) => {
        principal_arn = role
        policy_associations = {
          admin = {
            policy_arn   = "arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy"
            access_scope = { type = "cluster" }
          }
        }
      }
    }
    # ... node groups, IRSA, encryption, logging
  }

  data "aws_iam_policy_document" "external_dns" {
    statement {
      actions   = ["route53:ChangeResourceRecordSets"]
      resources = [aws_route53_zone.cluster.arn]
    }
    statement {
      actions   = ["route53:ListHostedZones", "route53:ListResourceRecordSets"]
      resources = ["*"]
    }
  }

  resource "aws_iam_role" "external_dns" {
    name = "${local.cluster_name}-external-dns"
    assume_role_policy = templatefile("${path.module}/templates/irsa-trust.json.tpl", {
      oidc_provider   = module.eks.oidc_provider
      oidc_arn        = module.eks.oidc_provider_arn
      service_account = "system:serviceaccount:external-dns:external-dns"
    })
    inline_policy {
      name   = "route53"
      policy = data.aws_iam_policy_document.external_dns.json
    }
  }
```



```
apiVersion: platform.example.com/v1alpha1
kind: Cluster
metadata:
  name: team-a-cluster-euc1
spec:
  environment: dev
  region: euc1
  baseDomain: team-a.cluster.example.com
  components:
    gateway-api: true
    argocd: true
    karpenter: true
    cilium: true
```

Yes. But how?


apiVersion: platform.example.com/v1alpha1
kind: Cluster
metadata:
name: team-a-cluster-eu1
spec:
environment: dev
region: eu1
baseDomain: team-a.cluster.example.com
components:
gateway-api: true
argocd: true
karpenter: true
cilium: true



```
locals {
  env_config = merge(
    yamldecode(file("${path.module}/envs/_defaults.yaml")),
    yamldecode(file("${path.module}/envs/${var.environment}.yaml")),
  )
  cluster_name = "${var.team}-cluster-${var.region_short}"
  azs          = slice(data.aws_availability_zones.this.names, 0, local.env_config.az_count)

  private_subnets = [for i, az in local.azs : cidrsubnet(var.vpc_cidr, 4, i)]
  public_subnets  = [for i, az in local.azs : cidrsubnet(var.vpc_cidr, 8, i + 48)]

  enabled_addons = {
    for name, cfg in local.env_config.addons : name => cfg
    if try(var.components[name], cfg.default_enabled)
  }

  common_tags = merge(var.tags, {
    "platform.example.com/team"        = var.team
    "platform.example.com/environment" = var.environment
    "karpenter.sh/discovery"           = local.cluster_name
  })
}

module "vpc" {
  source          = "terraform-aws-modules/vpc/aws"
  name            = local.cluster_name
  cidr            = var.vpc_cidr
  azs             = local.azs
  private_subnets = local.private_subnets
  public_subnets  = local.public_subnets

  single_nat_gateway = var.environment != "prod"
  private_subnet_tags = merge(local.common_tags, {
    "kubernetes.io/role/internal-elb" = "1"
  })
  # ... flow logs, endpoints, NACLs
}

module "eks" {
  source                         = "terraform-aws-modules/eks/aws"
  cluster_name                   = local.cluster_name
  cluster_version                = local.env_config.k8s_version
  vpc_id                         = module.vpc.vpc_id
  subnet_ids                     = module.vpc.private_subnets
  bootstrap_self_managed_addons = !try(var.components.cilium, false)

  access_entries = {
    for role in var.admin_role_arns : basename(role) => {
      principal_arn = role
      policy_associations = {
        admin = {
          policy_arn   = "arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy"
          access_scope = { type = "cluster" }
        }
      }
    }
  }
  # ... node groups, IRSA, encryption, logging
}

data "aws_iam_policy_document" "external_dns" {
  statement {
    actions   = ["route53:ChangeResourceRecordSets"]
    resources = [aws_route53_zone.cluster.arn]
  }
  statement {
    actions   = ["route53:ListHostedZones", "route53:ListResourceRecordSets"]
    resources = ["*"]
  }
}

resource "aws_iam_role" "external_dns" {
  name = "${local.cluster_name}-external-dns"
  assume_role_policy = templatefile("${path.module}/templates/irsa-trust.json.tpl", {
    oidc_provider   = module.eks.oidc_provider
    oidc_arn        = module.eks.oidc_provider_arn
    service_account = "system:serviceaccount:external-dns:external-dns"
  })
  inline_policy {
    name   = "route53"
    policy = data.aws_iam_policy_document.external_dns.json
  }
}
```

We've seen ~93% reduction in customer facing code with this method

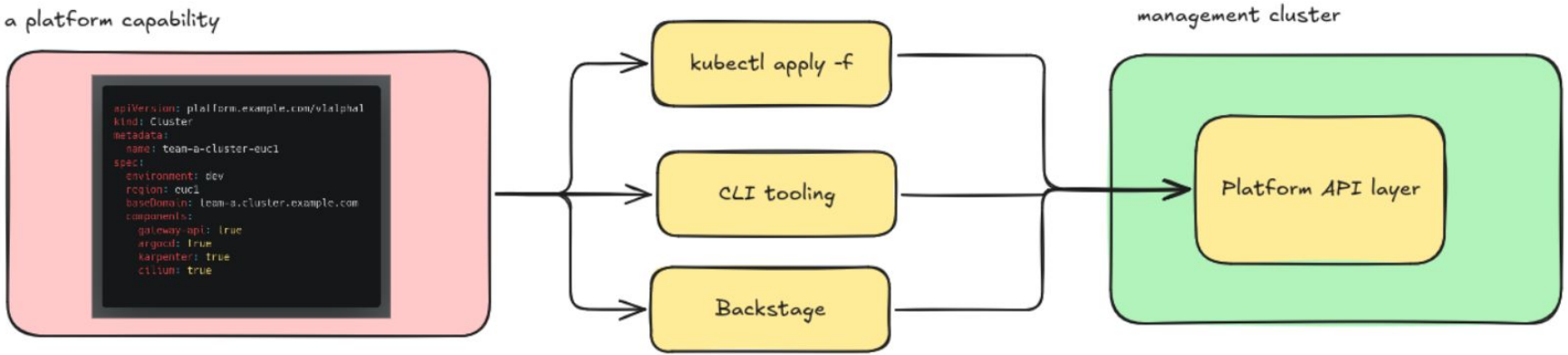
```
apiVersion: platform.example.com/v1alpha1
kind: Cluster
metadata:
  name: team-a-cluster-euc1
spec:
  environment: dev
  region: euc1
  baseDomain: team-a.cluster.example.com
  components:
    gateway-api: true
    argocd: true
    karpenter: true
    cilium: true
```

We don’t need new tooling, we need a framework

human & agentic users



```yaml
apiVersion: platform.example.com/v1alpha1
kind: Cluster
metadata:
  name: team-a-cluster-euc1
spec:
  environment: dev
  region: euc1
  baseDomain: team-a.cluster.example.com
  components:
    gateway-api: true
    argocd: true
    karpenter: true
    cilium: true
```


a platform capability
apiVersion: platform.example.com/v1alpha1
kind: Cluster
metadata:
name: team-a-cluster-eucl
spec:
environment: dev
region: eucl
baseDomain: team-a.cluster.example.com
components:
gateway-api: true
argocd: true
karpenter: true
cilium: true
kubectl apply -f
CLI tooling
Backstage
management cluster
Platform API layer

```go
func main() {
    sdk := kratix.New()

    // the user's request: the Cluster YAML
    cluster := must(sdk.ReadResourceInput())
    spec := func(field string) any {
        v, _ := cluster.GetValue("spec." + field)
        return v
    }

    // the platform picks the module version, per environment
    version := "v2-1-0"
    if spec("environment") == "prod" {
        version = "v2-0-3"
    }

    // wrap the existing module in a Tofu Controller resource
    tofu := map[string]any{
        "apiVersion": "infra.contrib.fluxcd.io/v1alpha2",
        "kind":       "Terraform",
        "metadata":   map[string]any{"name": cluster.GetName()},
        "spec": map[string]any{
            "interval":           "10m",
            "approvePlan":        "auto",
            "sourceRef":          map[string]any{"kind": "GitRepository", "name": "eks-cluster-" + version},
            "path":               "./modules/eks-cluster",
            "serviceAccountName": "tofu-clusters",
            "vars": []map[string]any{
                {"name": "environment", "value": spec("environment")},
                {"name": "region_short", "value": spec("region")},
                {"name": "base_domain", "value": spec("baseDomain")},
                {"name": "components", "value": spec("components")},
            },
        },
    }
    check(sdk.WriteOutput("cluster.yaml", must(yaml.Marshal(tofu))))

    // and tell the user what they got
    status := kratix.NewStatus()
    check(status.Set("moduleVersion", version))
    check(sdk.WriteStatus(status))
}

func must[T any](v T, err error) T {
    check(err)
    return v
}

func check(err error) {
    if err != nil {
        log.Fatal(err)
    }
}
```

platform api layer

this could be Python, Bash, whatever you're comfortable with

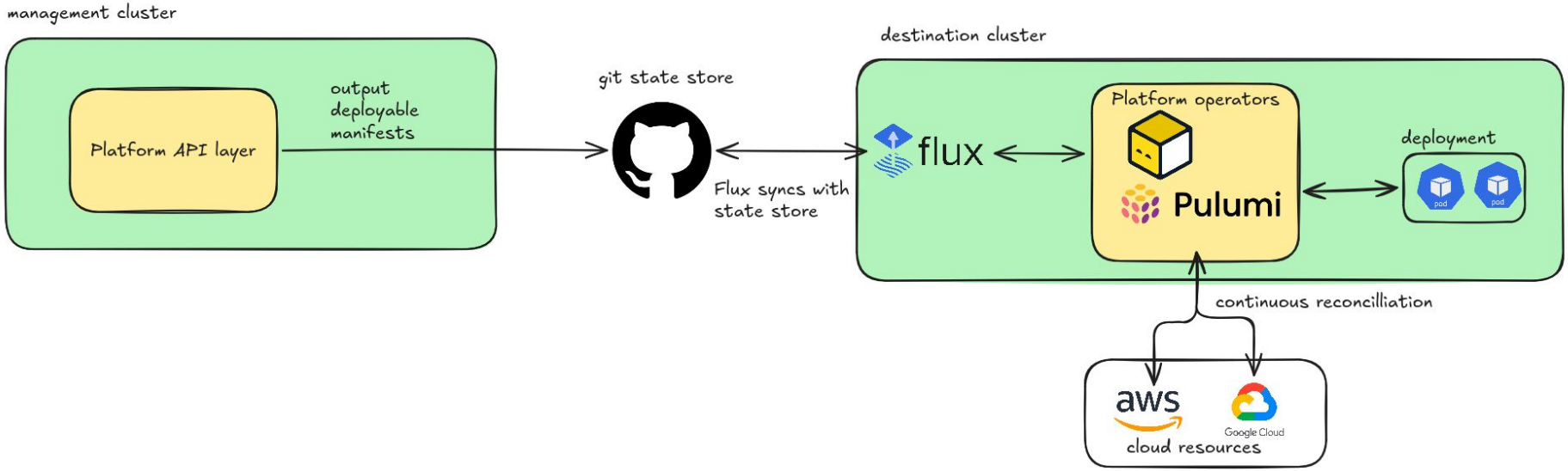
platform maintainers/contributors

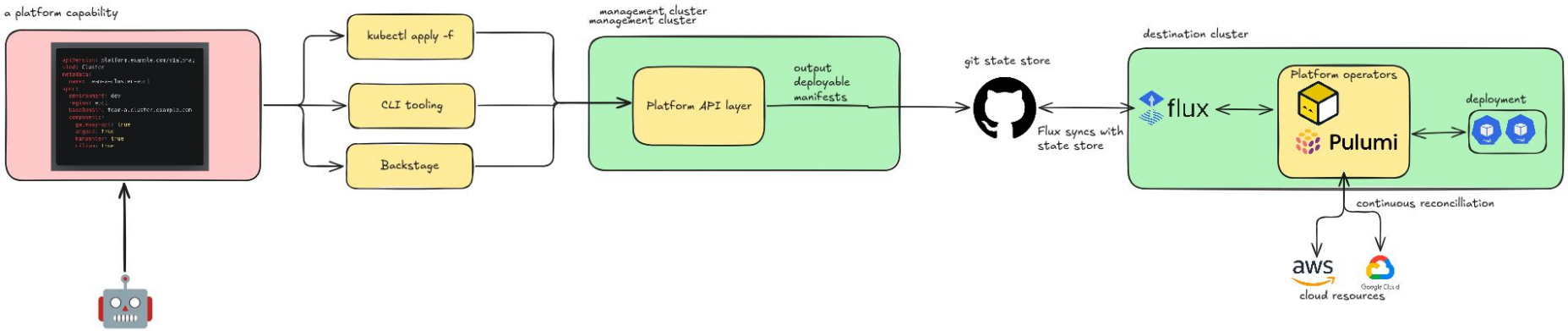


platform layer output

cluster.yaml

```yaml
apiVersion: infra.contrib.fluxcd.io/v1alpha2
kind: Terraform
metadata:
  name: team-a-cluster-euc1
spec:
  interval: 10m
  approvePlan: auto
  sourceRef:
    kind: GitRepository
    name: eks-cluster-v2-1-0          # ① version, set by the platform
  path: ./modules/eks-cluster         # ② your existing module
  serviceAccountName: tofu-clusters   # ③ IAM lives here
  vars:
    - { name: environment,  value: dev }
    - { name: region_short, value: euc1 }
    - { name: base_domain,  value: team-a.cluster.example.com }
    - name: components
      value: {"gateway-api":true,"argocd":true,"karpenter":true,"cilium":true}
```


management cluster
Platform API layer
output deployable manifests
git state store
Flux syncs with state store
destination cluster
flux
Platform operators
Pulumi
deployment
pod
pod
continuous reconcilliation
aws
Google Cloud
cloud resources


a platform capability
kubectl apply -f
CLI tooling
Backstage
management cluster
management cluster
Platform API layer
output
deployable
manifests
git state store
Flux syncs with
state store
destination cluster
flux
Platform operators
Pulumi
deployment
continuous reconcilliation
aws
Google Cloud
cloud resources

Better together

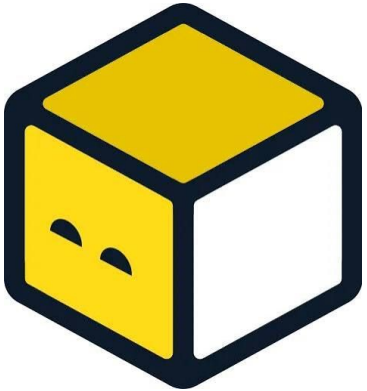




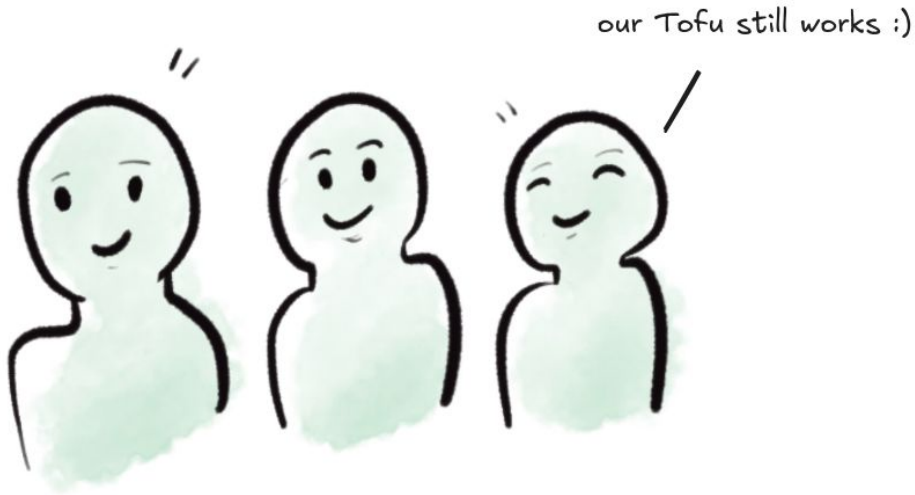


Pulumi

Crossplane




Flux
Pulumi
faster, safer way to build
in the age of AI


our Tofu still works :)

KubeCon CloudNativeCon
North America 2026

Thank you!

https://www.linkedin.com/in/juhosyrjanen/

https://github.com/juhosyrjanen

Special thanks to Aino for the illustrations!

