Platforms-as-a-Product:

A Crossplane'd Vision

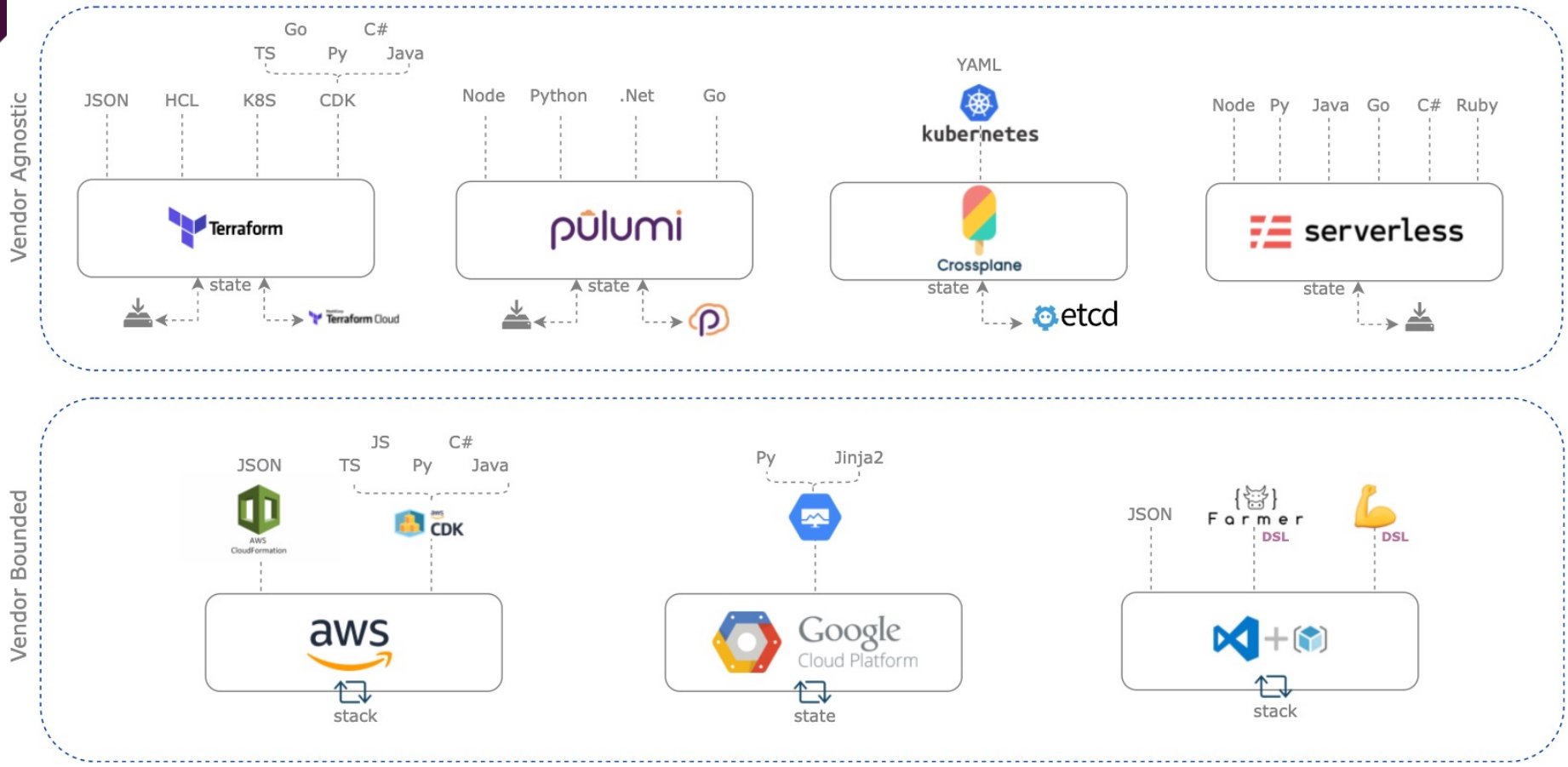
Lucas Albuquerque | 01-02-2022

Opening Remarks

- Special interest on **Platform Engineering** and **IaC** Tooling/Practices
- The **vision behind the tool** matters most
- Helps on “**connecting the dots**”
- **Insights** on how to envision and deliver **Platform resources as products**
- **Not a Elevation Pitch** for any tool
- **No PoC**... at least for this presentation

45 min

IaC Landscape

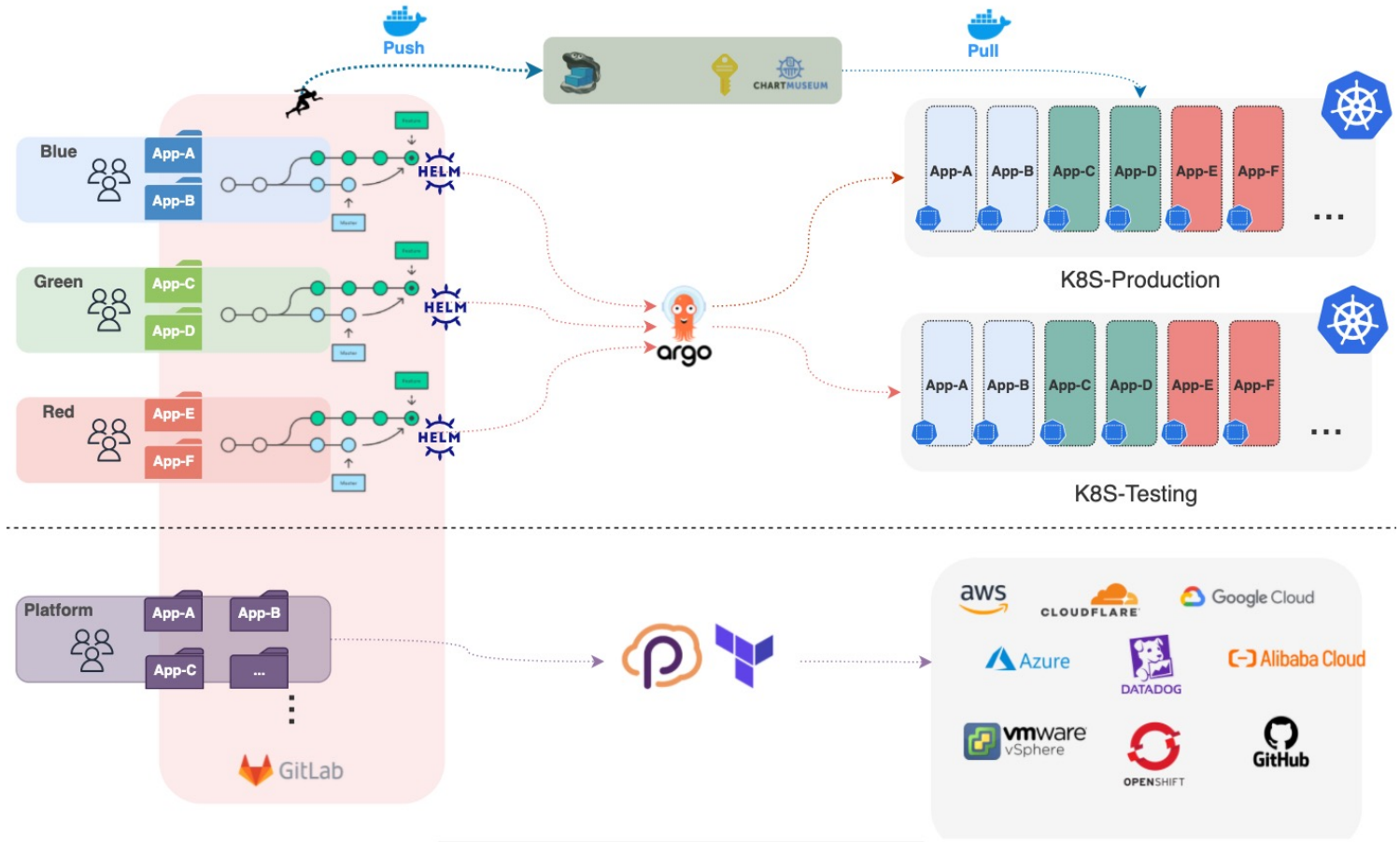


Cloud-Native Scenario

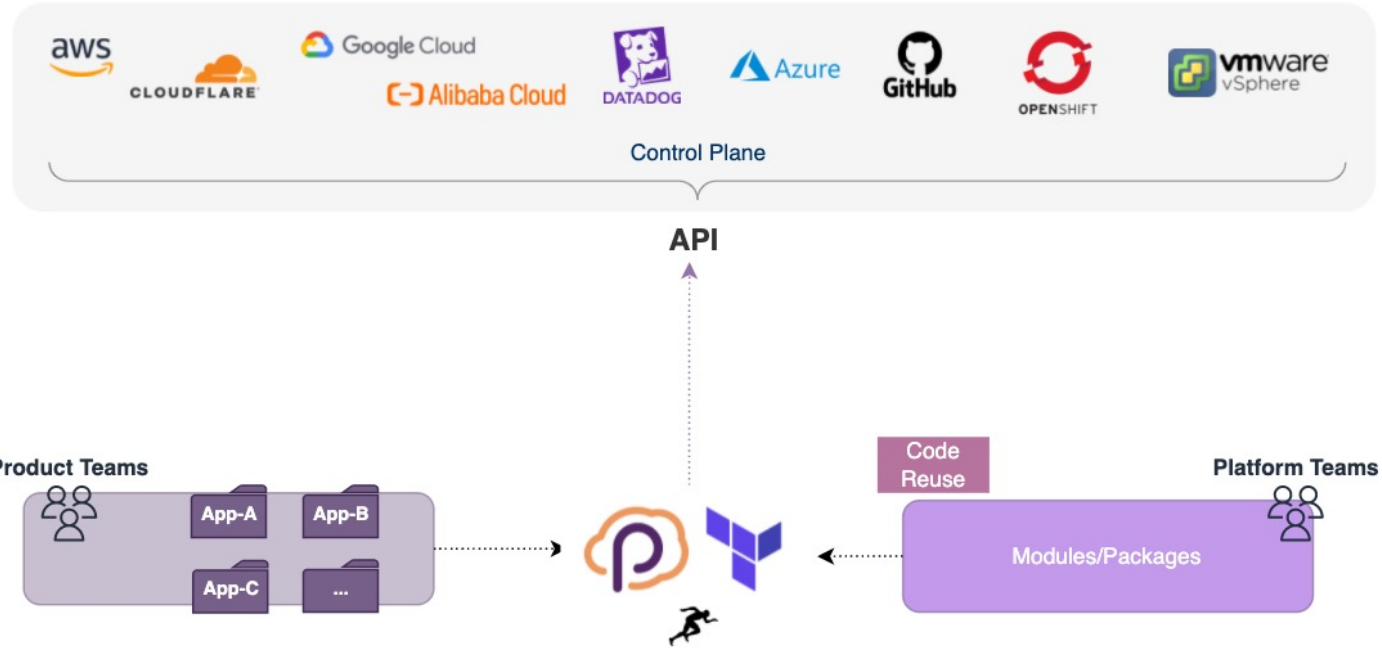
Application Layer

Unpaired Lifecycles

Platform Layer



APIs and Control-Planes




```
import  { StaticWebsite } from "static-website-aws";

const website  = new StaticWebsite ("browserhack", {
    pathToContent:"./browserhack",
    custom404Path:"/404.html",
});
```

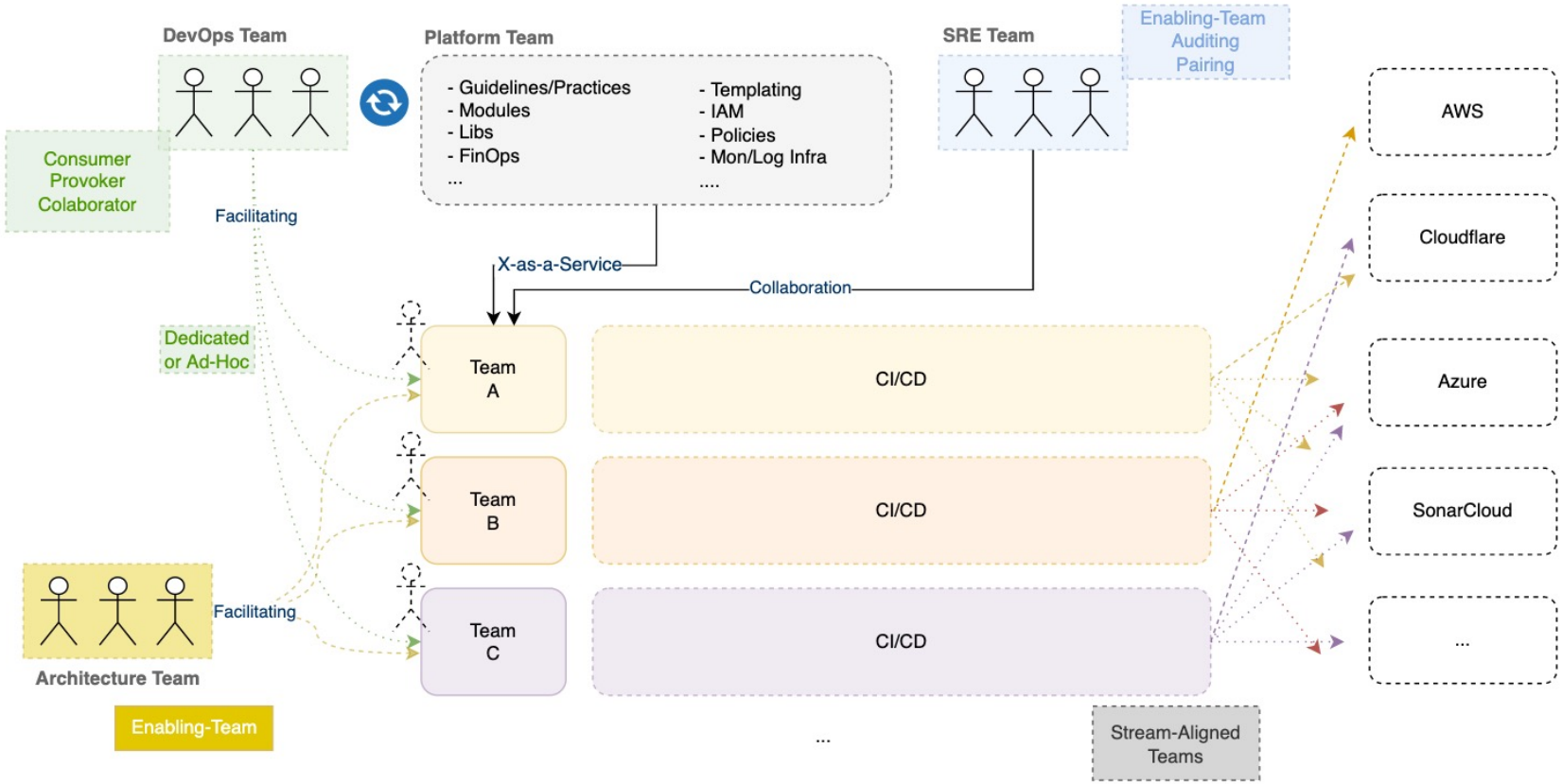
```
module "consul" {
  source  = "hashicorp/consul/aws"
  version = "0.0.5"

  servers = 3
}
```

Does the Job, but…

- Platform **resources declared** differently from Applications
  - ✓ Helm/Kustomize for Apps
  - ✓ Tfvars or *pulumi config set* for Platform
- Runners have **access to Modules/Packages Code**
  - ✓ Potential of subverting business logic
- **IAM** Aspects
  - ✓ Runner Access to Credentials
  - ✓ Managing Credentials and Roles lifecycle
- **State files mgmt, Continuous Delivery and reconciliation**
  - ✓ TF Cloud
  - ✓ Pipelines

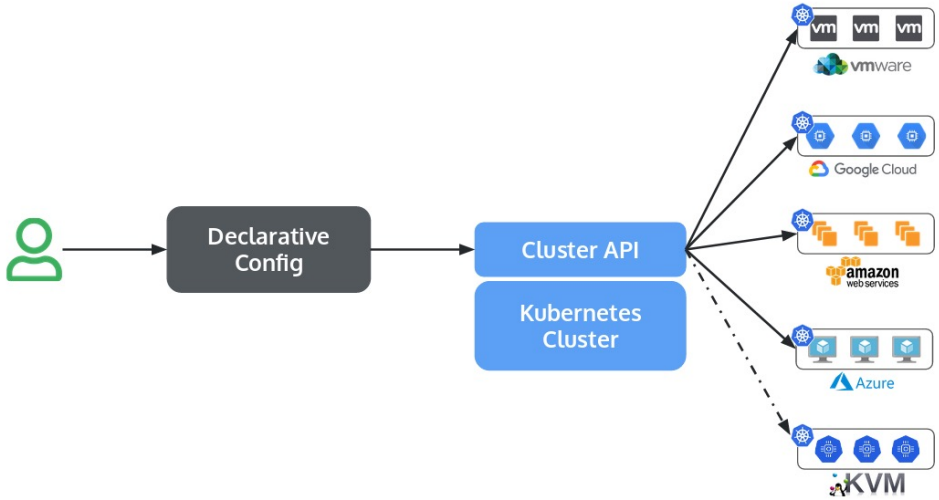
Scaling can turns out on being a challenge!

Teams' Interactions



Looking Down the Road…

- Bringing **infra mgmt** to the same level as **app mgmt**
- **K8S is here to stay**… is not a matter of being good or bad!
- **K8S becoming transparent** fo the Consumers… **get the job done** is what matters!
- **Detach K8S API and Custom Operators** from container-related tasks only



Why not manage Providers control planes through K8S control plane?

And then Crossplane comes...

- Crossplane as a tool for **creating opinionate tools**
- K8S API as a **Universal Cloud API**
- **Not a Novel Idea**... but well put and at the right timing!
- Neither the only kid in the block... e.g. **Rancher**
- **Upbound received 60M funds**

But we can start with the trivial....

Beyond the Trivial...

- The trivial just makes Crossplane yet another IaC tool
  - https://github.com/kube-champ/terraform-operator
  - https://www.pulumi.com/blog/pulumi-kubernetes-operator/
- This approach works as a wrapper-only
- This underutilize the real K8S API's potential

```yaml
apiVersion: run.terraform-operator.io/v1alpha1
kind: Terraform
metadata:
  name: first-module
spec:
  terraformVersion: 1.0.2

  module:
    source: IbraheemAlSaady/test/module
    ## optional module version
    version:

   ## a terraform workspace to select
  workspace:

  ## a custom terraform backend
  backend: |
    backend "local" {
      path = "/tmp/tfmodule/mytfstate.tfstate"
    }

  ## a custom providers config
  providersConfig:
```

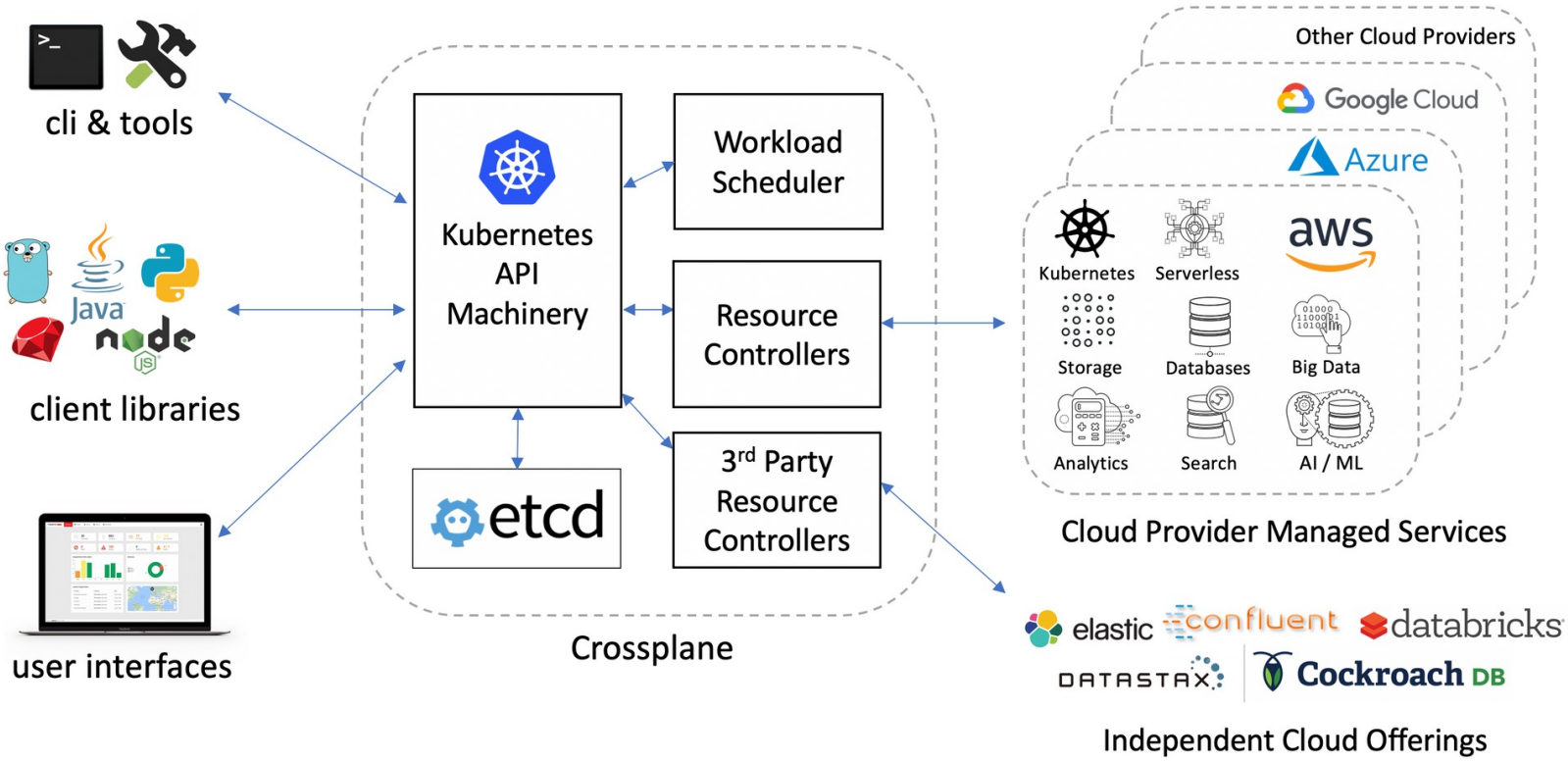
```yaml
  ## a list of terraform variables to be provided
  variables:
    - key: length
      value: "16"

    - key: AWS_ACCESS_KEY
      valueFrom:
        ## can be configMapKeyRef as well
        secretKeyRef:
          name: aws-credentials
          key: AWS_ACCESS_KEY
      environmentVariable: true

  ## files with ext '.tfvars' or '.tf' that will be mounted into the ter
  ## to be passed to terraform as '-var-file'
  variableFiles:
    - key: terraform-env-config
      valueFrom:
        ## can also be 'secret'
        configMap:
          name: "terraform-env-config"
```

Beyond the Trivial...

- Define, compose and offer **your own infra API**

- **Hide** infrastructure **complexity** and **add** policy **guardrails**
- All **Declarative**. No code required

Beyond the Trivial...



Beyond the Trivial...

Deploying an GKE Cluster

Step 1: Set the Provider

```yaml
apiVersion: gcp.crossplane.io/v1beta1
kind: ProviderConfig
metadata:
  name: default
spec:
  credentials:
    secretRef:
      key: credentials.json
      name: gcp-credentials
      namespace: crossplane-system
    source: Secret
  projectID: crossplane-playground
```

Beyond the Trivial...

Deploying an GKE Cluster

Step 2: Set the Network

```yaml
apiVersion: multik8s.platformref.crossplane.io/v1alpha1
kind: Network
metadata:
  name: network-gcp
spec:
  id: multik8s-network-gcp
  clusterRef:
    id: multik8s-cluster-gcp
  compositionSelector:
    matchLabels:
      provider: GCP

```

Beyond the Trivial...

Deploying an GKE Cluster

Step 3: Provision the Cluster

```yaml
apiVersion: multik8s.platformref.crossplane.io/v1alpha1
kind: Cluster
metadata:
  name: multik8s-cluster-gcp
spec:
  compositionSelector:
    matchLabels:
      provider: GCP
  id: multik8s-cluster-gcp
  parameters:
    nodes:
      count: 3
      size: small
    services:
      operators:
        prometheus:
          version: "10.0.2"
    networkRef:
      id: multik8s-network-gcp
  writeConnectionSecretToRef:
    name: cluster-conn-gcp

```

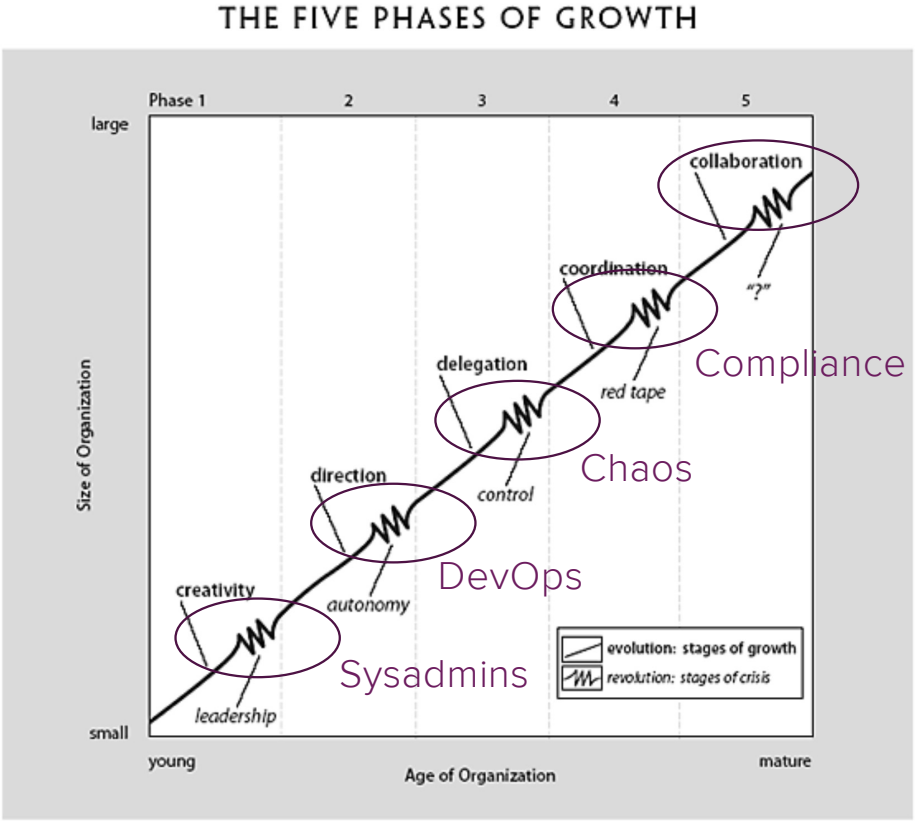
When not to use Crossplane approach

- If you are **happy with your current tool** and workflow, just fine.
- If **you don't have K8S**, there's no point on provisioning one just for that.
- If **you don't have apps running on K8S**... cognitive load.
- If you **don't have a mature platform** team.
- If **learning curve** is something that matters for you.
- If you are on **MVP stage**, when you need to start simple.

Maturity Path

- Article: “Evolution and revolution as organizations grow.” 1972. (Larry E. Greiner) [1]
- Presents the stages of Organization’s Growth and Challenges

Platform Teams somehow imitate the same trends.



As-a-Service

What Comes Next…

- Proof-of-Technology (PoT)
- Proof-of-Concept (PoC)
- Boilerplating (e.g. BackStage.io)

Xebia

Thanks for Joining
Questions?

Lucas Albuquerque | 11-12-2020


Xebia