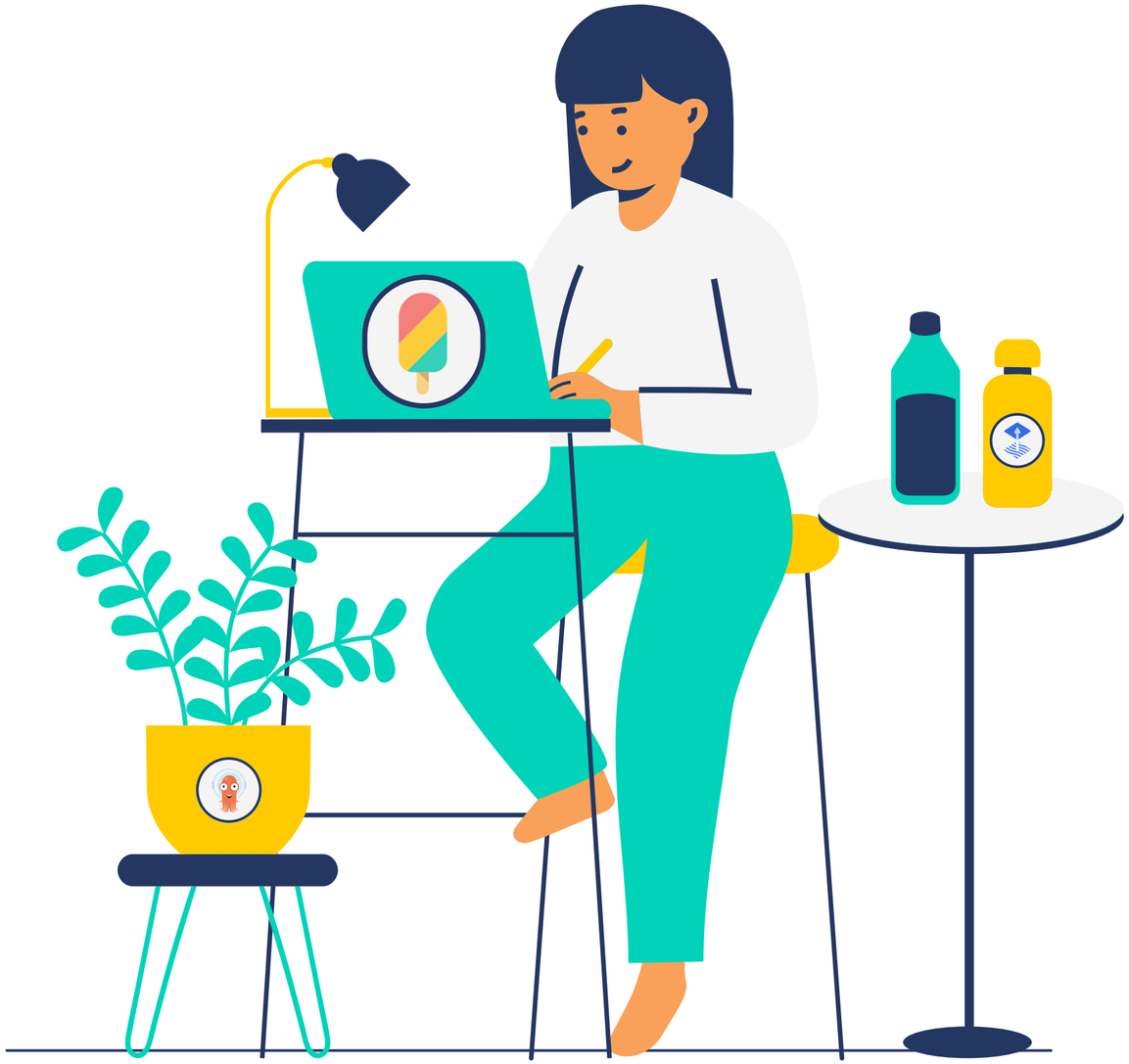
Crossplane & GitOps

A perfect match?

Katharina Sick

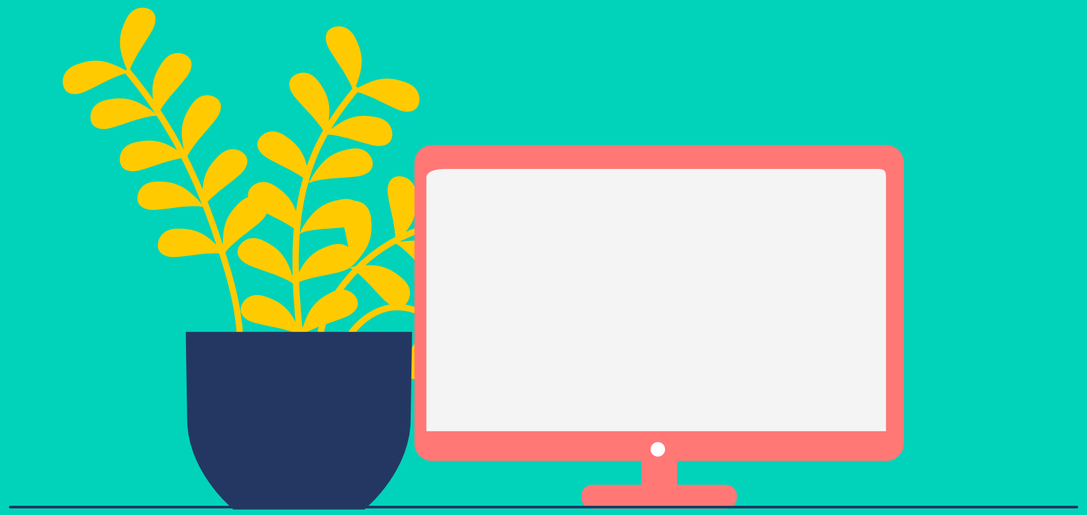
Cloud Native Linz | March 28, 2023



Hello, I'm Katharina

Software Engineer

@Dynatrace

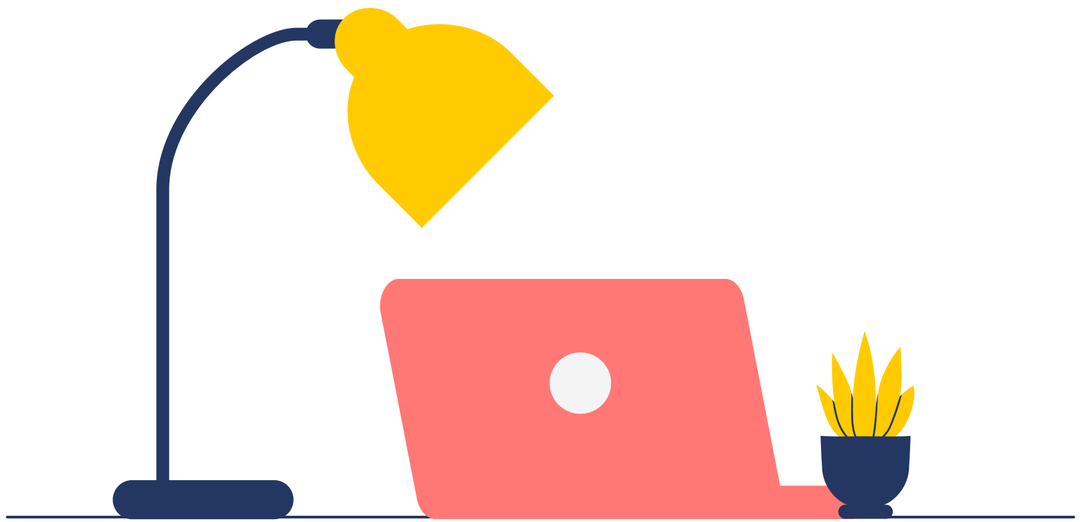


Get in touch

https://www.ksick.dev

About today

Let's take a look at the questions
I'll try to answer in this talk



Some basics about GitOps and Infrastructure as Code

What are they? Why is the combination of Crossplane and GitOps so powerful?

Our story

How did we unlock those technologies in our team?

Our learnings

What would we do differently now and what are we going to change?

Wait, what's GitOps?

An operational framework for the continuous deployment of applications and infrastructure

Principles

Declarative configuration

- The state of the system is described in a declarative way
- In practice: Kubernetes manifests

Versioned and immutable

- The desired state is stored in a system with version control, immutability, and auditing
- In practice: Git

Pulled automatically

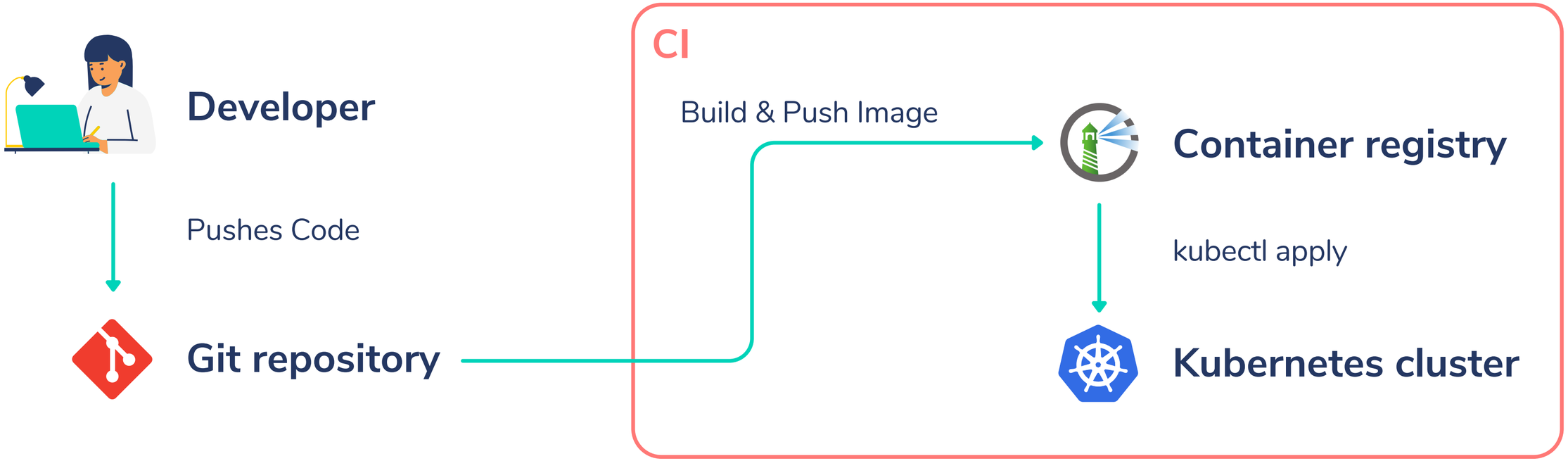
- The configuration is automatically pulled and applied to the target system
- In practice: ArgoCD, Flux, Kubevela,...

Continuously reconciled

- Software agents continuously monitor the system to ensure that no drift happens
- In practice: ArgoCD, Flux, Kubevela,...

KubeVela

Traditional Deployment



GitOps Deployment

Developer

Pushes code

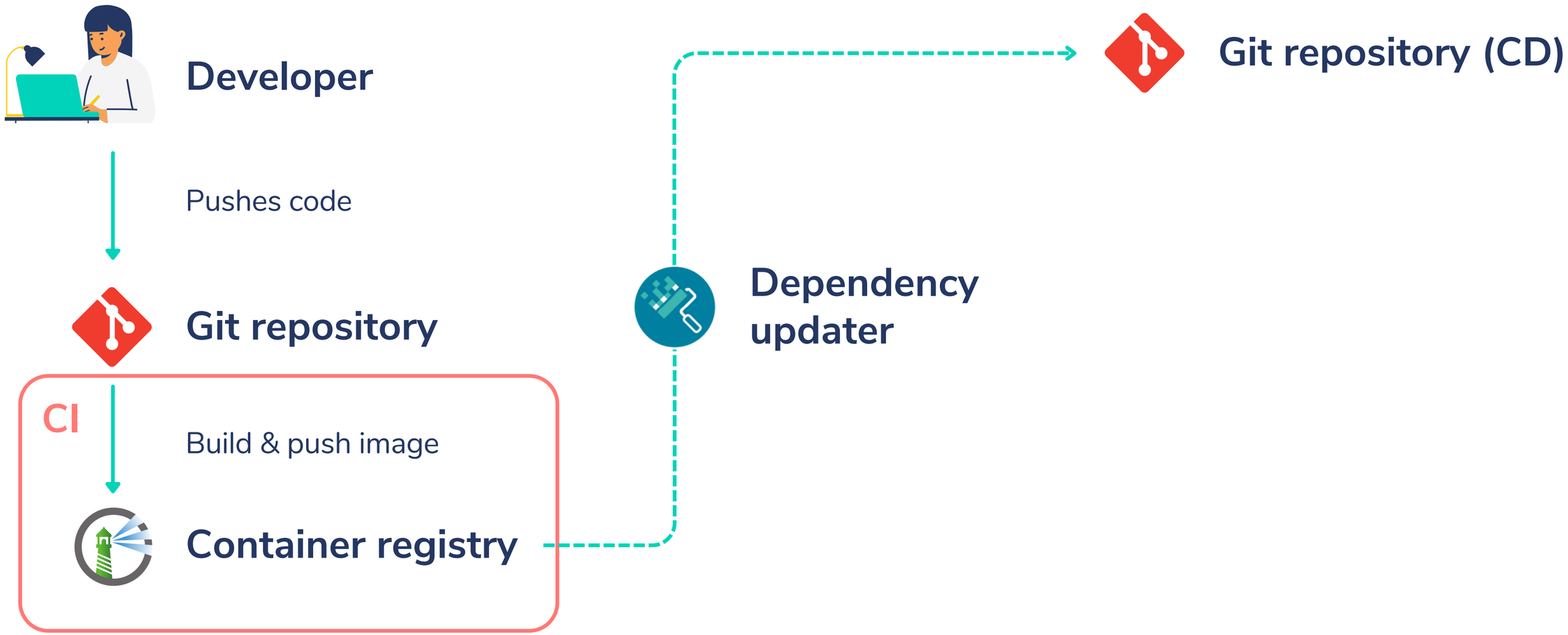
Git repository

CI

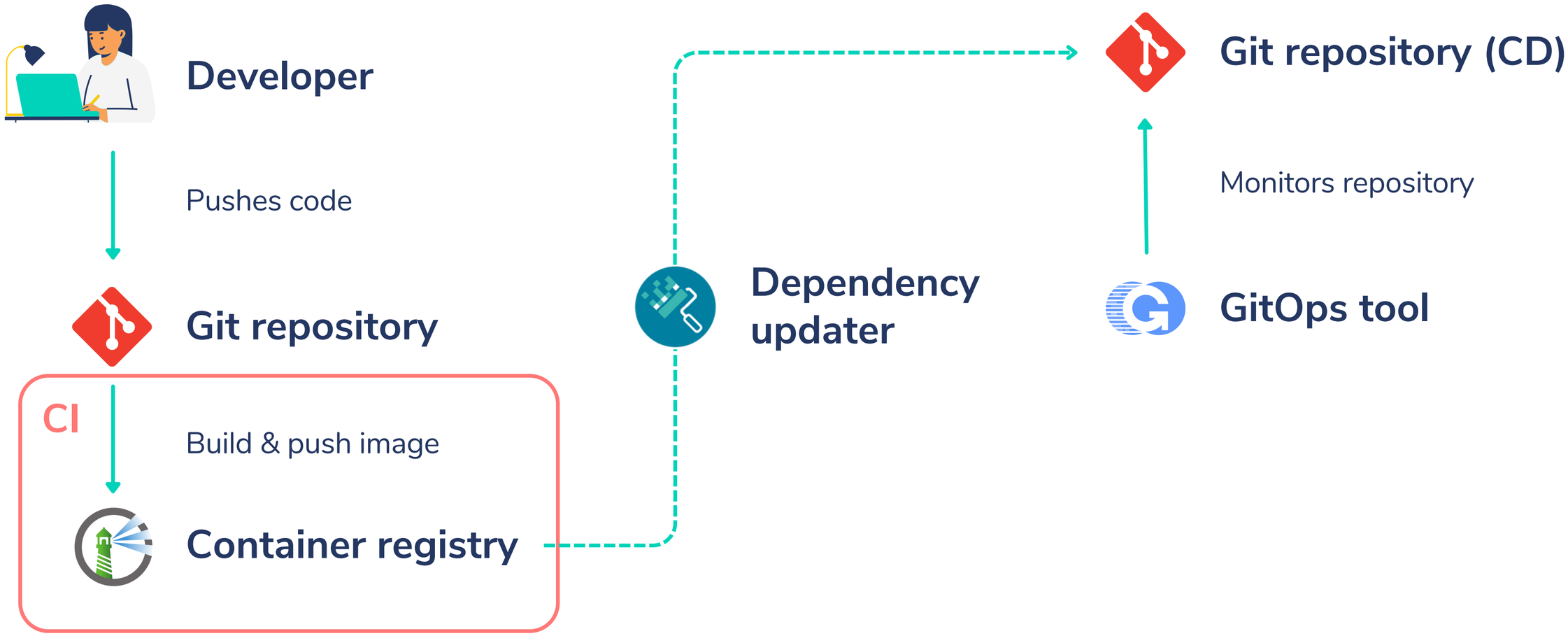
Build & push image

Container registry

GitOps Deployment



GitOps Deployment



GitOps Deployment

Developer

Pushes code

Git repository

CI

Build & push image

Container registry

Dependency updater

Git repository (CD)

Monitors repository

GitOps tool

Applies config

Kubernetes cluster

Main Benefits

Single source of truth

- The repository reliably shows the deployed resources
- No configuration drift

Faster deployments and rollbacks

- Easily deploy with a git commit
- Rollback by reverting a git commit

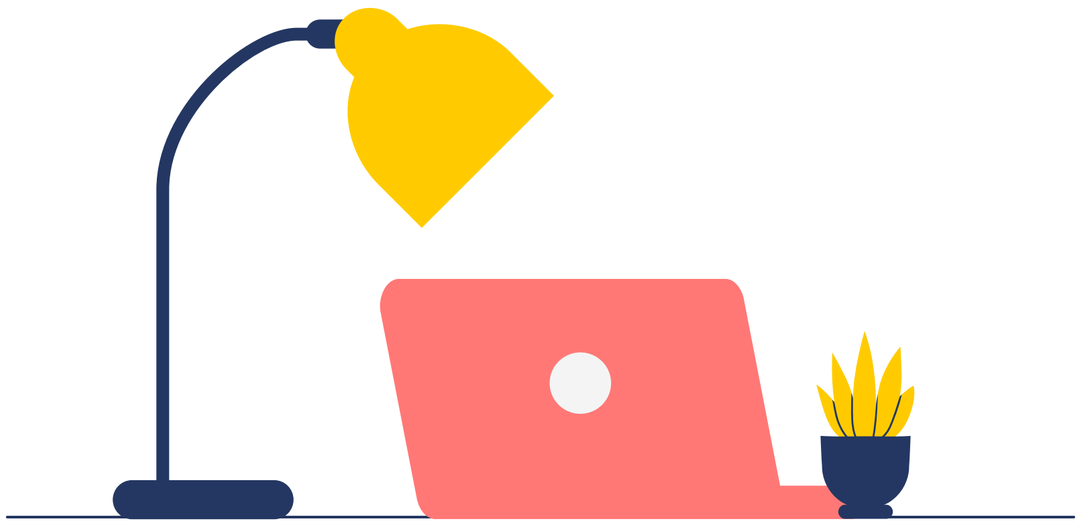
Better collaboration

- By incorporating software development best practices for deployments

Security and Compliance

- No direct access to Kubernetes clusters
- Four eyes principle (pull requests)
- Version information including who deployed and why

Demo Time

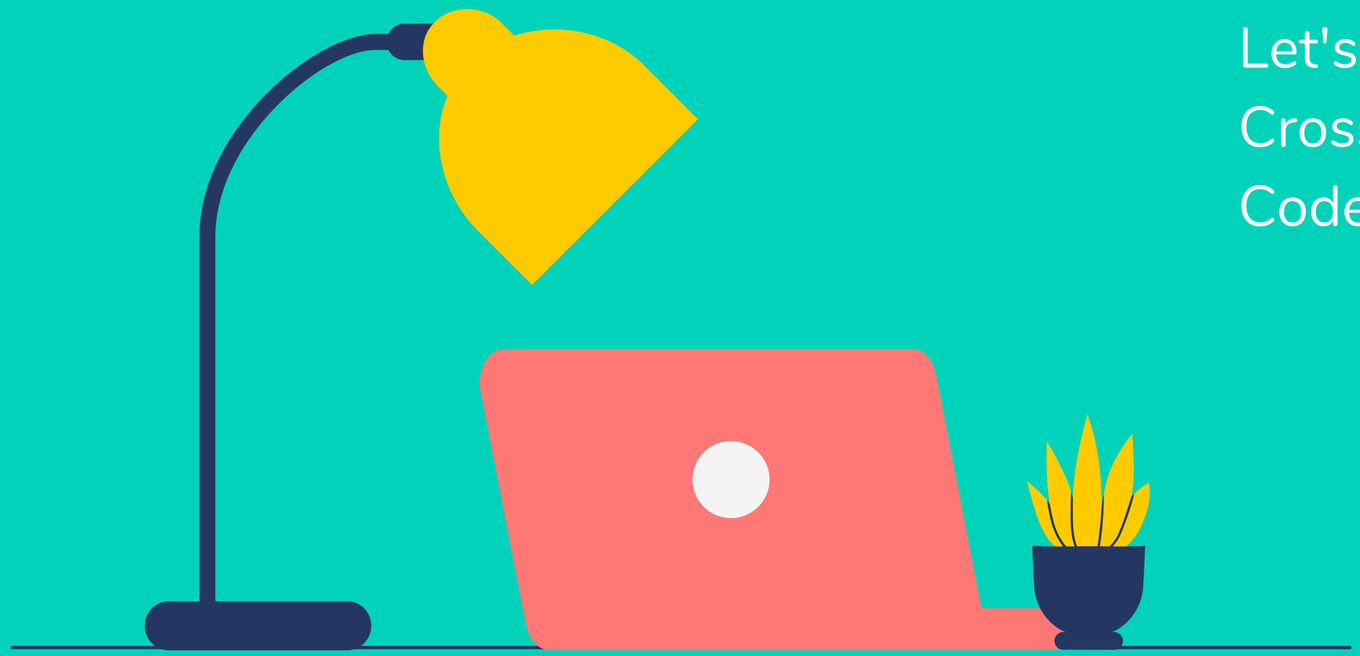


Simple deployment with ArgoCD

Let's deploy a simple application with GitOps

And Crossplane?

Let's take a brief look at Crossplane and Infrastructure as Code



Infrastrucuture as Code

Manage infrastructure through code

- No need for developers to manually provision and manage servers, operating systems, database connections, storage, and other infrastructure

Declarative or imperative

- Declarative: What should be applied?
- Imperative: Which steps should be taken to apply the infrastructure?

Many benefits

- Version control
- Increase deployment speed
- Consistency

Terraform

Declarative configuration

- The desired state is described with the Hashicorp Configuration Language (*.hcl)

Multiple environments

- Platform agnostic
- Using providers enables us to deploy to any target system

Large community

- Terraform has a very large open source community

Apply with Terraform CLI

- `terraform apply`

bucket.tf

```
resource "google_storage_bucket" "demo" {
  name          = "ksick-cncf-linz-demo-terraform"
  location      = "EU"
  force_destroy = true

  lifecycle_rule {
    condition {
      age = 3
    }
    action {
      type = "Delete"
    }
  }

  lifecycle_rule {
    condition {
      age = 1
    }
    action {
      type = "AbortIncompleteMultipartUpload"
    }
  }
}
```

HashiCorp Terraform

Crossplane

Cloud Native control plane

- Allows to extend a Kubernetes cluster to provision, manage and orchestrate infrastructure and services

Declarative configuration

- The desired state is described in a declarative way through Kubernetes manifests

Multiple environments

- Platform agnostic
- Using providers enables us to deploy to any target system

Apply with kubectl

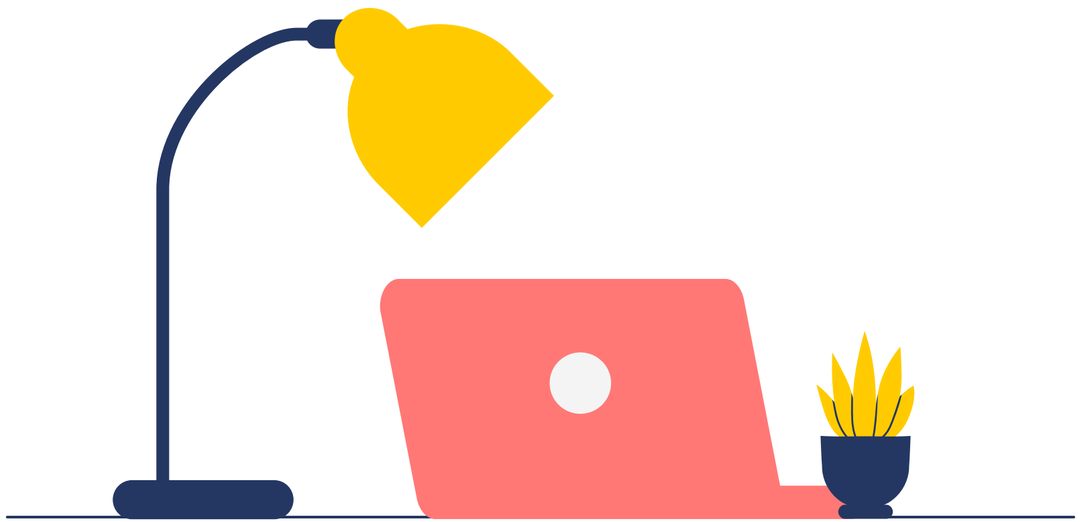
- `kubectl apply -f bucket.yaml`

bucket.yaml

```yaml
apiVersion: storage.gcp.upbound.io/v1beta1
kind: Bucket
metadata:
  name: ksick-cncf-linz-demo-crossplane
spec:
  forProvider:
    location: EU
    forceDestroy: true
    lifecycleRule:
      - condition:
          - age: 3
        action:
          - type: Delete
      - condition:
          - age: 1
        action:
          - type: AbortIncompleteMultipartUpload
  providerConfigRef:
    name: default
```

Crossplane

Demo Time



Deploy a bucket with Terraform and Crossplane

Let's compare those two tools

Taking the next step

You can deploy any Kubernetes manifest with GitOps

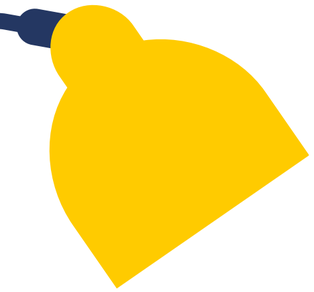
- Native Kubernetes manifests (or Helm charts) for applications
- Crossplane resources for infrastructure

Works with your Kubernetes toolset

- Your GitOps tool is just another Kubernetes controller
- You can still use your standard tools like Helm, Kustomize, and many others

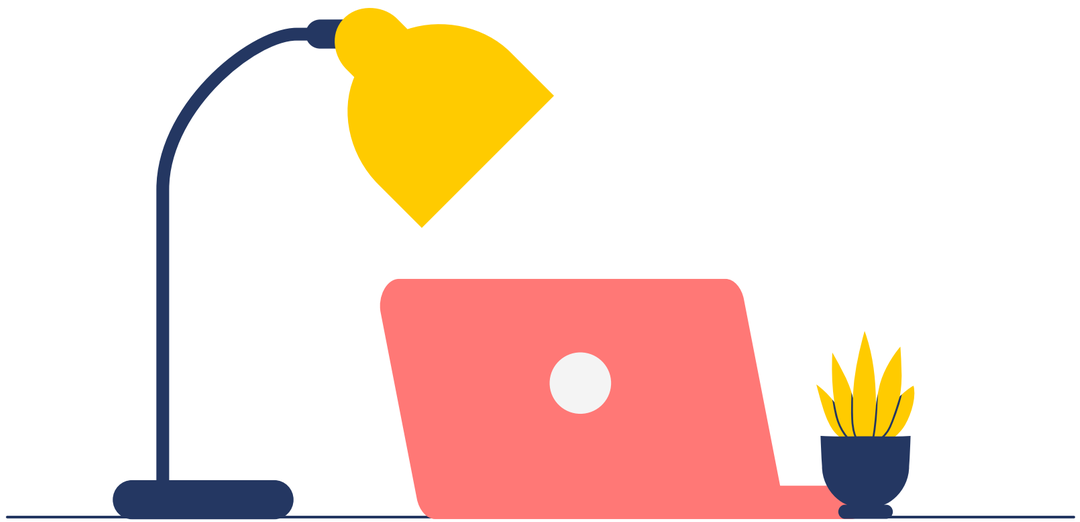
Benefit from GitOps for infrastructure as well

- Crossplane is managing resources and their lifecycle
- The GitOps tool makes sure to keep everything in sync



With Crossplane, you can benefit from GitOps for infrastructure as well

Demo Time

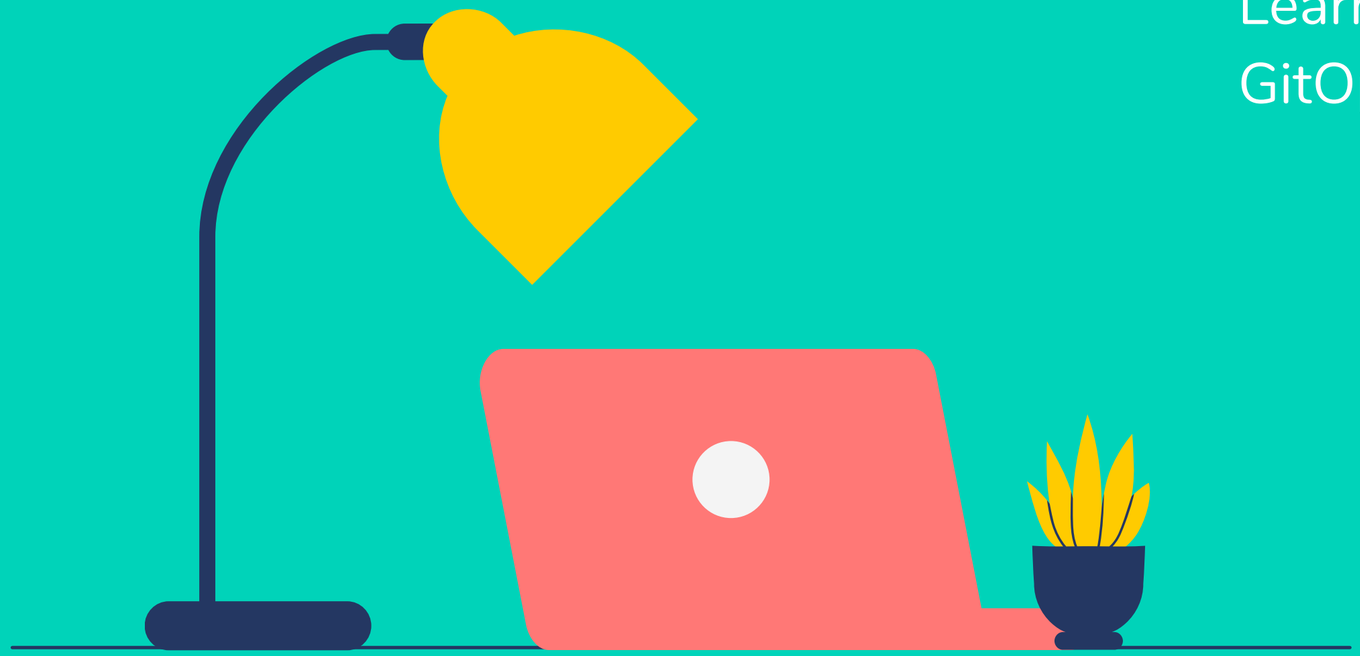


Infrastructure deployment with ArgoCD and Crossplane

Use GitOps to deploy an application with some infrastructure

In practice

Learn how our team is using
GitOps and Infrastructure as Code



Our main topics

Engineering Productivity

- We're providing various services, infrastructure, and frameworks to increase developer productivity in our company

CI Infrastructure

- We're running around 40 Jenkins instances with additional infrastructure

Backstage

- We're currently establishing Backstage as an internal developer portal
- Enabling teams to create templates to scaffold new projects easily

Other

- Test frameworks, automatic issue generation, pipeline monitoring and statistics, test infrastructure, ...

CI Infrastructure

Jenkins instances

- Owned by the developers that are using it
- We're providing updates and common functionalities

Additional tooling

- Renovate Bot, Dynatrace Dashboard, and one or more dedicated Vault engines
- Optional Gradle Cache or Sonarqube

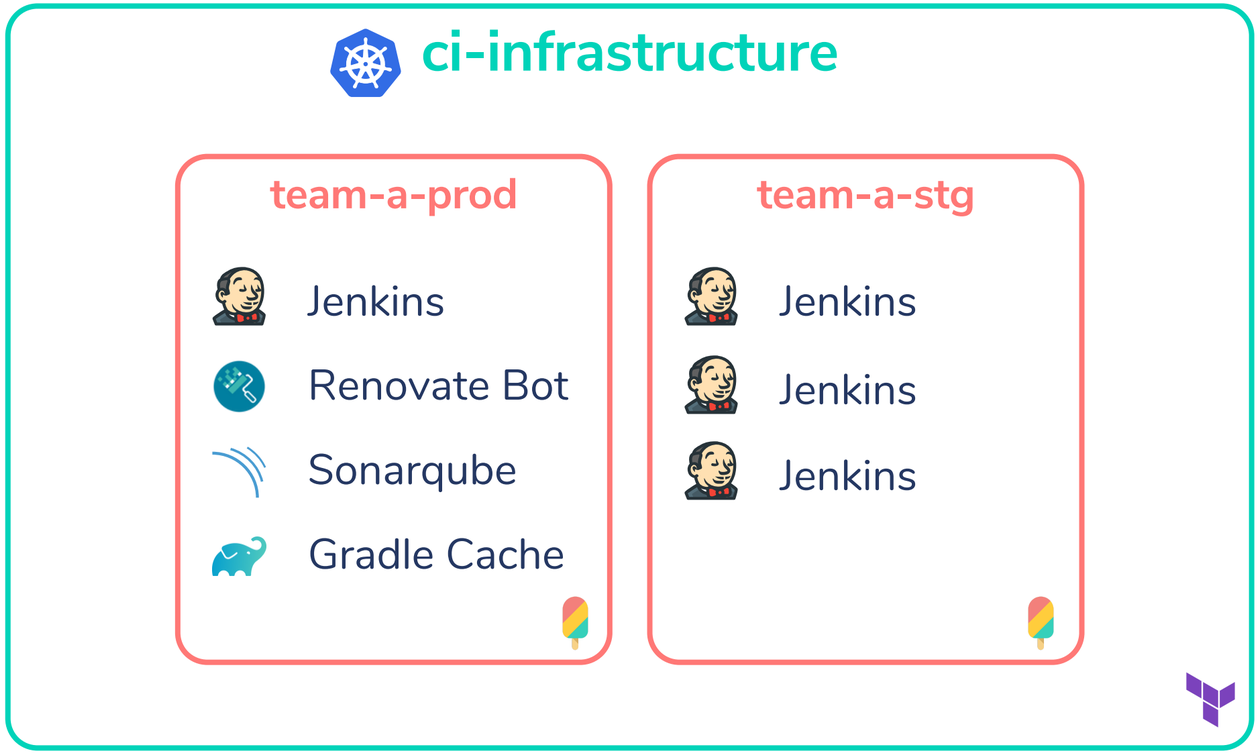
Dedicated Kubernetes cluster

- CI and Test Infrastructure
- CI applications and tools (for example Backstage)

Easy setup via templating

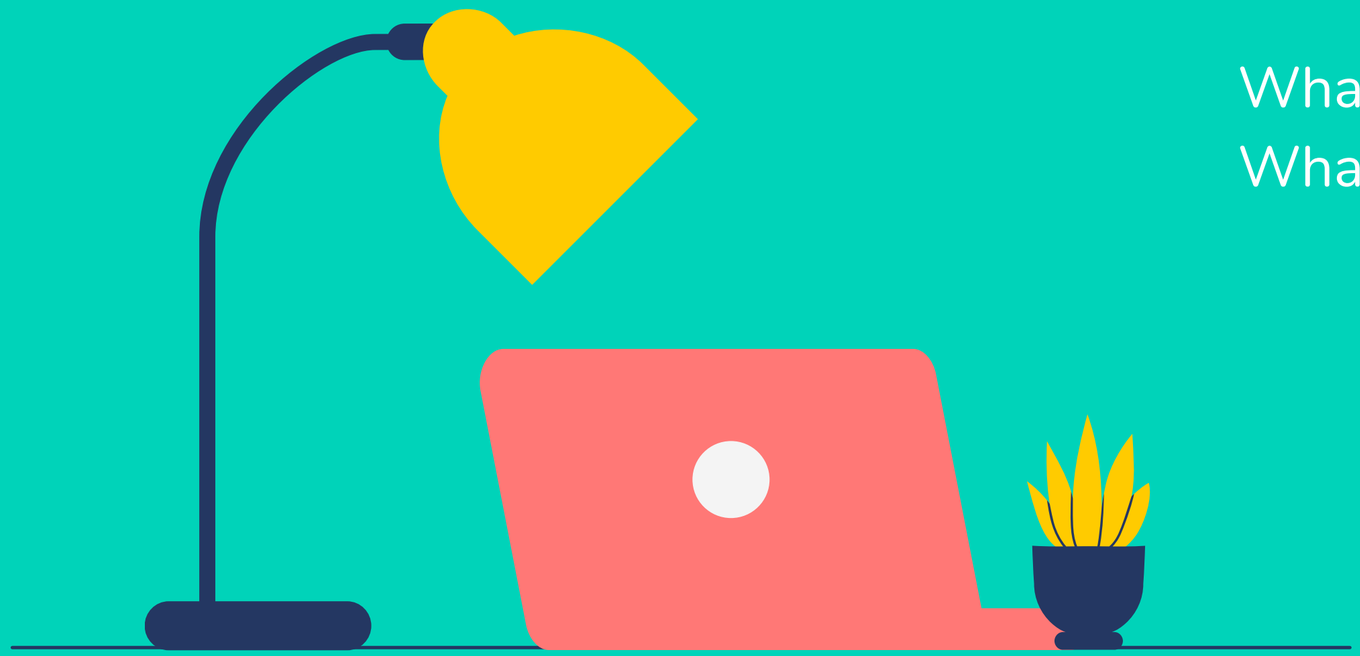
- It takes around an hour to provision the CI Infrastructure for a new team/topic
- We are aiming to reduce this time by improving our templates

CI Environment: Cluster



Our Experiences

What have we learned?
What would we solve differently now?



Start small

Start with small components to get to know Crossplane

- E.g. databases, caches, node groups, certificates,...

We attempted to create a production-ready EKS cluster to get started

- Too much complexity as a first task
- Frustration because resource inputs and outputs didn't match

Don't stop reiterating

- Not everything has to be perfect from the beginning on
- We are now revisiting creating a cluster with a lot more knowledge and confidence

The chicken-egg problem

You need a Kubernetes cluster to use Crossplane

- We initialized our cluster with Terraform
- Could be solved by spinning up a local cluster in a script or CI build and letting the cluster manage itself

The chicken-egg problem

You need a Kubernetes cluster to use Crossplane

- We initialized our cluster with Terraform
- Could be solved by spinning up a local cluster in a script or CI build and letting the cluster manage itself

Temporary

local (e.g.kind)
Kubernetes cluster

GitOps tool

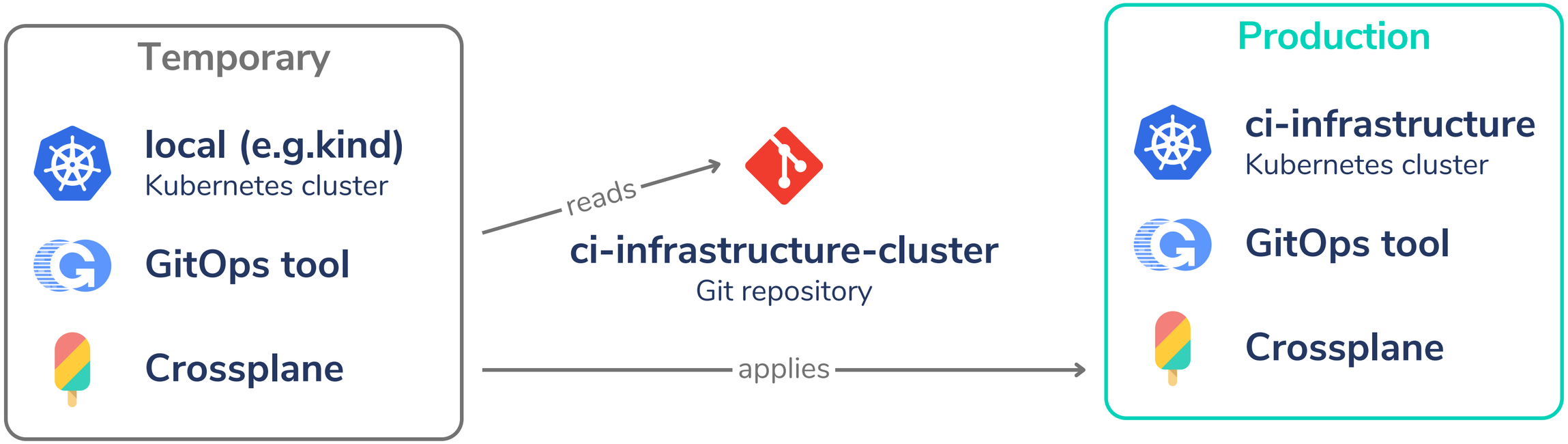
Crossplane

ci-infrastructure-cluster
Git repository

The chicken-egg problem

You need a Kubernetes cluster to use Crossplane

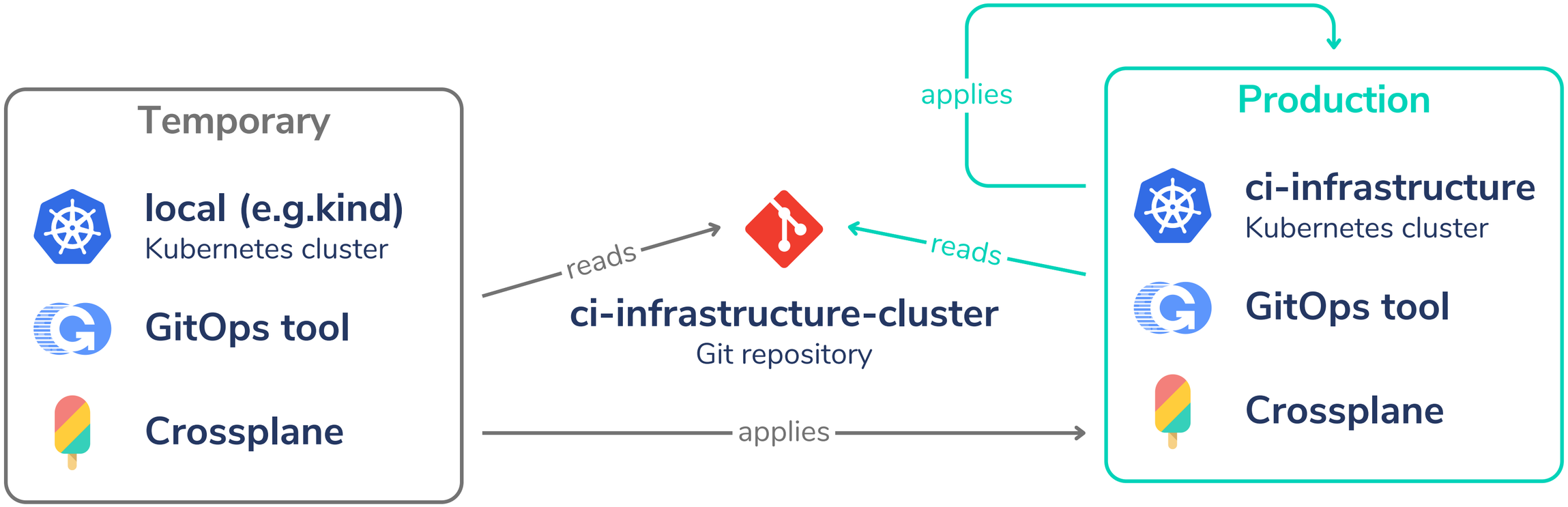
- We initialized our cluster with Terraform
- Could be solved by spinning up a local cluster in a script or CI build and letting the cluster manage itself



The chicken-egg problem

You need a Kubernetes cluster to use Crossplane

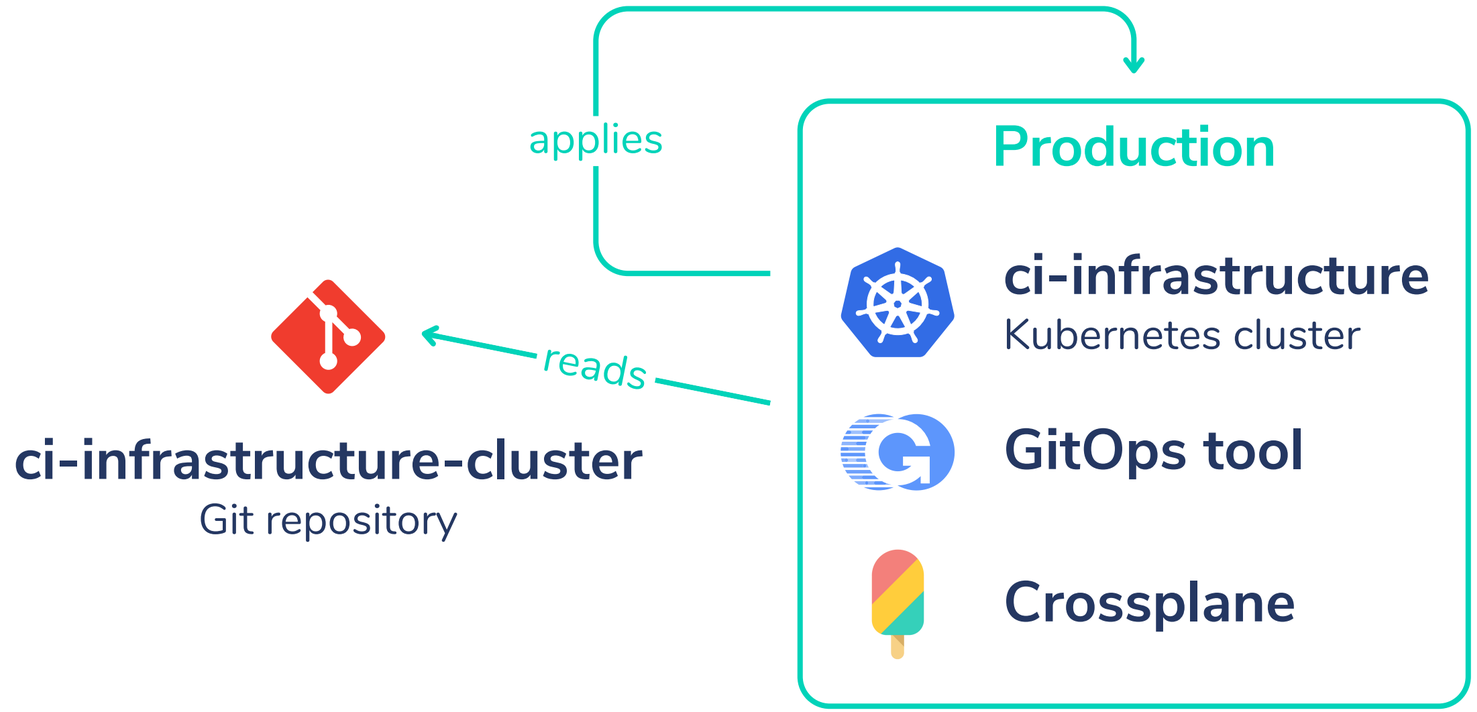
- We initialized our cluster with Terraform
- Could be solved by spinning up a local cluster in a script or CI build and letting the cluster manage itself



The chicken-egg problem

You need a Kubernetes cluster to use Crossplane

- We initialized our cluster with Terraform
- Could be solved by spinning up a local cluster in a script or CI build and letting the cluster manage itself



Sync vs maintenance windows

Don't use the maintenance windows of your cloud provider

- Avoid confusion
- Can be achieved with sync windows
  - Works out of the box in ArgoCD
  - Needs some workarounds to work in Flux

Use compositions for everything

Managed resources are cluster scoped

- That makes things like role-based access control a lot harder
- Only composite resources are namespace scoped
- Upbound/Crossplane recommends to always compositions

Many resources need patches anyways

- Setting up a KMS key encryption for certain resources doesn't work without creating a composition as you need patches for this

bucket.yaml

```yaml
apiVersion: apiextensions.crossplane.io/v1
kind: Composition
metadata:
  ...
spec:
  ...
  resources:
    ...
    - name: test-resource
      ...
      patches:
        - fromFieldPath: status.kmsKeyARN
          toFieldPath: spec.forProvider.kmsKeyId
    - ...
```

Enforce certain rules

Follow the principle of least privilege

- Use multiple provider configurations
- No need for one huge service account
- Each has a different service account

ArgoCD project permissions

- Explicitely allow or block resources that can be applied in certain projects

Cluster Policies

- Guard your Kubernetes clusters with policies (e.g. with Kyverno)
- E.g. restrict provider config usage

How to package applications?

How to build Helm charts with Crossplane?

- We always try to keep our Helm charts fully self-contained
- Can we assume the necessary composite resource definitions are present in the target cluster, or should we include them in the package?

Or should we get rid of Helm?

- The Kubernetes Crossplane provider allows to include of standard Kubernetes resources to Compositions
  - Using the Kubernetes Crossplane provider feels wrong

Are we beta testers?

[AWS] Inconsistent in- & outputs of managed resources

- E.g. the 'id' output needs to be set to the 'arn' input
- E.g. necessary secret input is not written to output secret

[AWS] Documentation lacks important information

- Required inputs are not marked as required in the documentation quite often
- [crossplane-contrib/provider-aws]: The resource was not updatable because of an undocumented TODO

[AWS] Many small bugs

- E.g. ElastiCache:
  - Can't create a user when a password is given
  - Does not write credentials to secret reference

Reading logs and errors

Rely on events for errors

- [AWS] The resource condition does not always represent the actual condition
- Events show details about problems

Reading logs

- You can read the logs of Crossplane and its providers by reading the pod logs of the corresponding pod

Hello again, Terraform

Many providers are generated from Terraform

- Don't be surprised to see Terraform log messages when increasing log verbosity
- Once we couldn't delete a resource because 'prevent_destroy' was set
- Documentation is also based on Terraform -> sometimes examples won't work

log.json

```
{
  "@level": "info",
  "@message": "Apply complete! Resources: 0 added, 1 changed, 0 destroyed.",
  "@module": "terraform.ui",
  "timestamp": "2023-03-26717:11:31.873396Z",
  "changes": {
    "add": 0,
    "change": 1,
    "remove": 0,
    "operation": "apply"
  },
  "type": "change_summary"
}

```

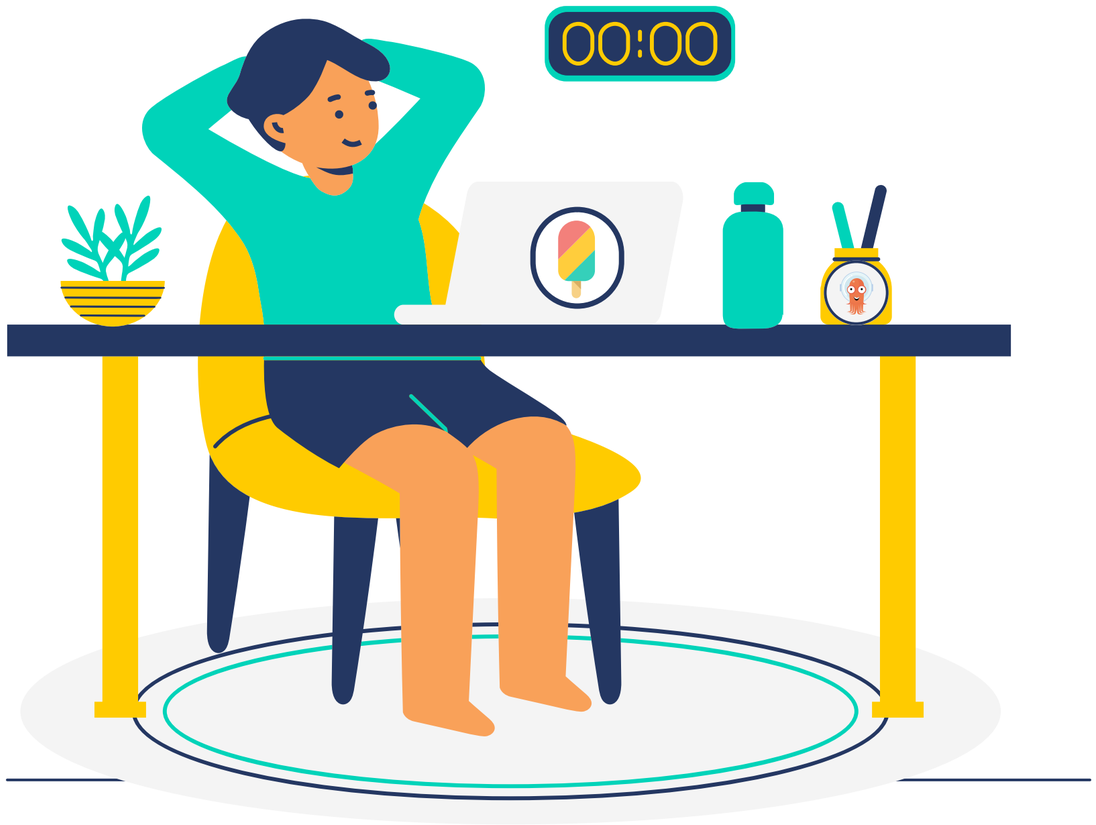
To sum it up

It is hard for us to trust Crossplane fully

- There are many minor issues
- Updates of resources sometimes don't happen

Crossplane opens awesome opportunities

- We love that we can use the GitOps workflow for everything with Crossplane
- We still need to gather much knowledge, but we will definitely continue using Crossplane



Thank you! :)

Resources

- **Live demos/examples**: https://github.com/KatharinaSick/PresentationMaterials/tree/main/20230328-CloudNativeLinz

- **OpenGitOps**: https://opengitops.dev/
- **General information**:
  - https://about.gitlab.com/topics/gitops/
  - https://www.redhat.com/en/topics/automation
  - https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac
  - https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code
- **Flux sync windows**: https://github.com/fluxcd/flux2/discussions/870
- **ArgoCD**: https://argo-cd.readthedocs.io/
- **Flux CD**: https://fluxcd.io/
- **Kubevela**: https://kubevela.io/
- **Terraform**: https://www.terraform.io/
- **Crossplane**: https://www.crossplane.io/
- **Renovate Bot**: https://docs.renovatebot.com/
- **Helm**: https://helm.sh/
- **Crossplane** Kubernetes Provider: https://github.com/crossplane-contrib/provider-kubernetes
- **Backstage**: https://backstage.io/
- **Kyverno**: https://kyverno.io/
- **OSS Insights**: https://ossinsight.io/
- **Showcode**: https://www.showcode.io/