Introduction to GitOps

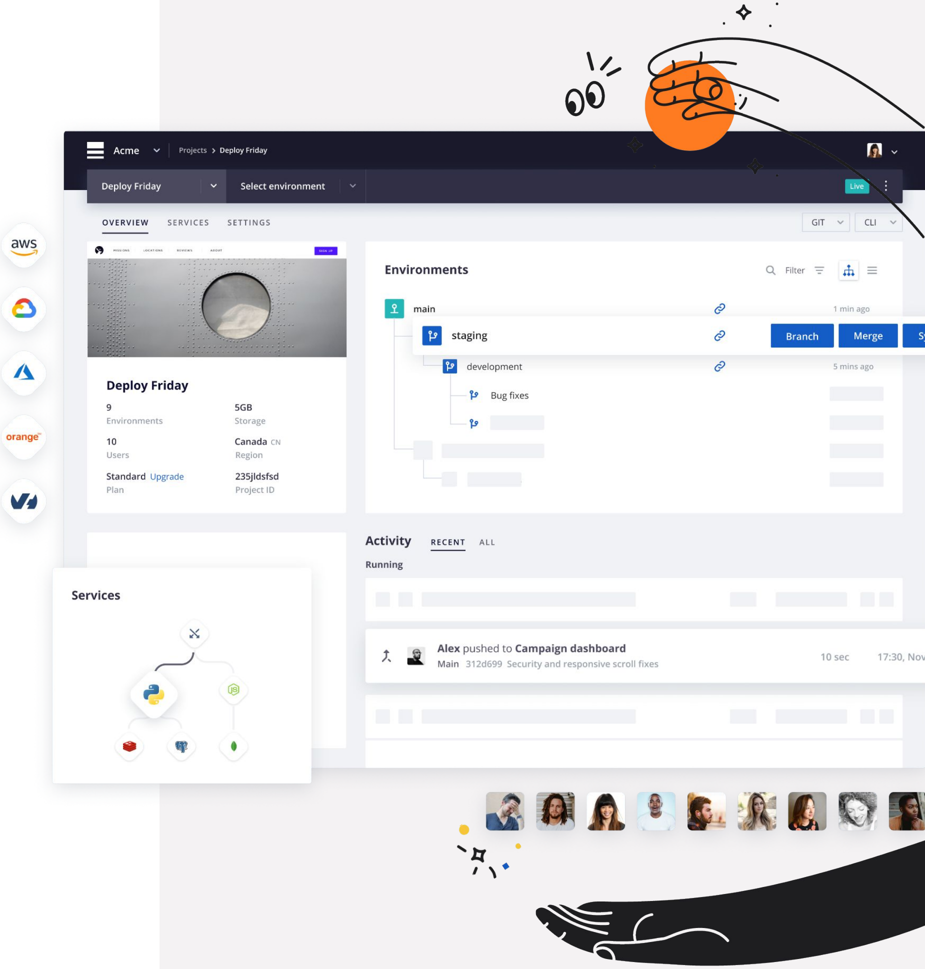
Paul Gilzow

Columbia, MO (USA)

Developer Relations Engineer

paul.gilzow@platform.sh

platform.sh



Deploy Friday

9 Environments

5GB Storage

10 Users

Canada CN Region

Standard Upgrade Plan

235jldsfsd Project ID

Alex pushed to Campaign dashboard
Main 312d699 Security and responsive scroll fixes
10 sec
17:30, Nov

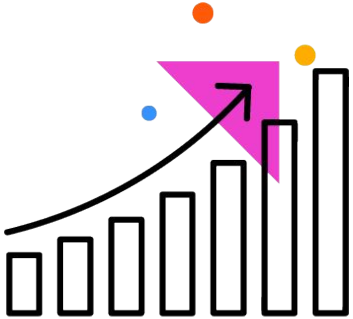
you're one of them software developin' types, aren't you?
i reckon y'all should git
imgflip.com

Outline

- The challenge(s) / history
- GitOps defined
- Principles of GitOps
- How is GitOps different from DevOps?
- Why GitOps
- How does GitOps benefit developers (aka "what's in it for me?")
- Disadvantages of GitOps

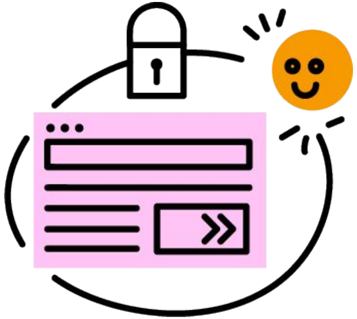
The Challenge(s)

In order to thrive, you need to innovate.



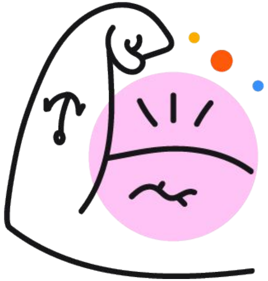
Drive enrollment growth

→



Deliver amazing digital experiences

But with innovation, comes complexity...



Lacking talent

Limited budget

Time Constraints

And yet the complexity doesn’t end there.

Increased Deployment Frequency

Developer-Driven Stack

Stack Diversity

Virtualization / Container Orchestration

New software architectures (microservices, decoupled)

Security & compliance

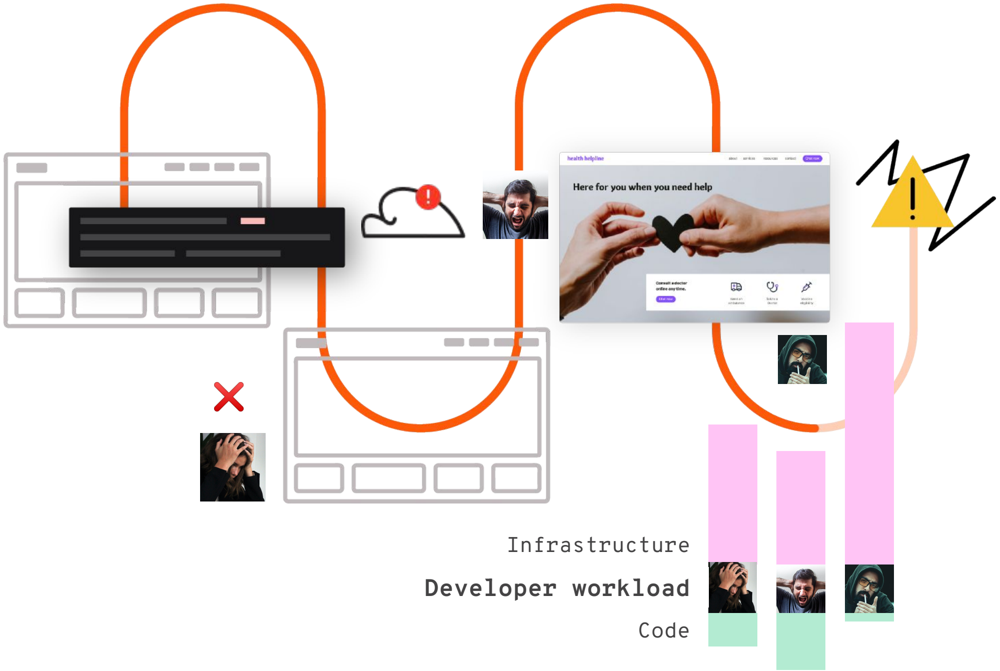
Complexity is **challenging** to adapt to...

Challenge #1

Hard to… ***move fast***

- Applications are managed across different vendors and tools
- Development to staging to production silos cause bottlenecks
- Teams spend more time on infrastructure than innovation

=> Slow response

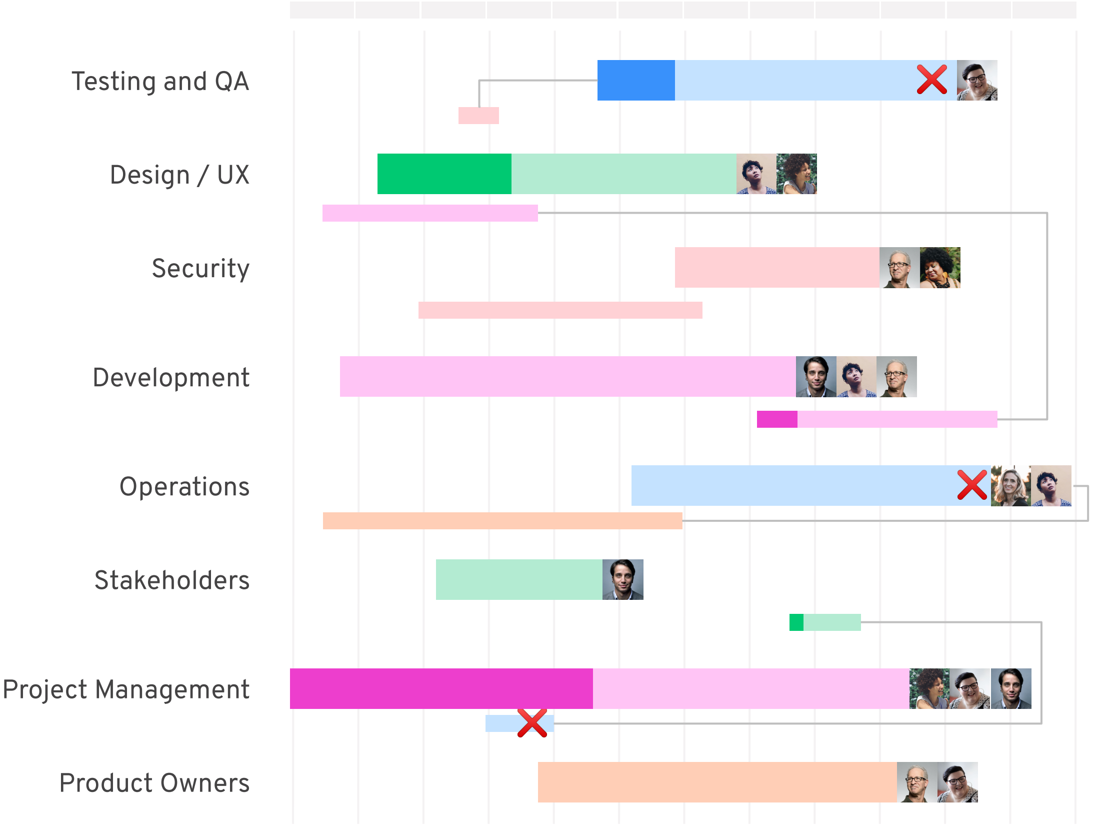


Challenge #2

Hard to… ***collaborate***

- No central control.
- Siloed teams, all doing their own things with their own tools.
- Disconnected technologies, etc. that are not hosted all in one place.

=> Causing delays and frustration

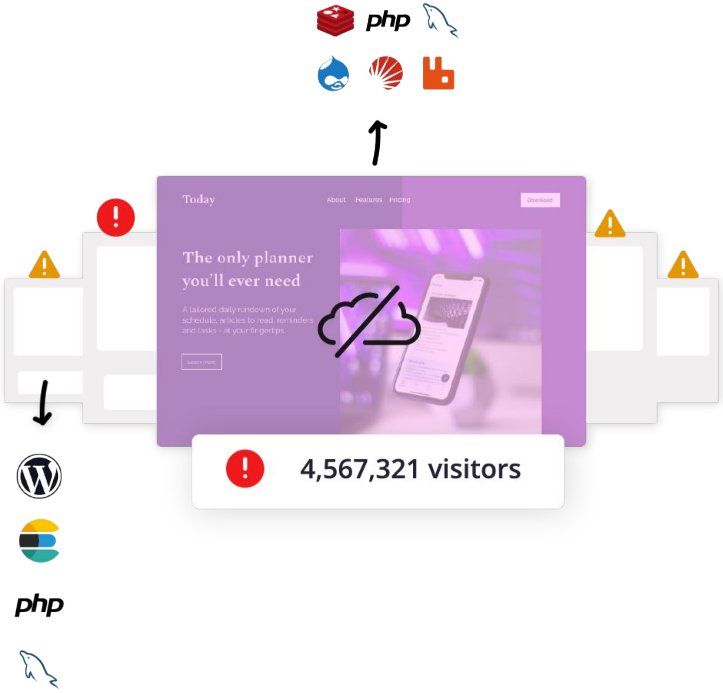


Challenge #3

Hard to… *scale*

- Outages due to large spikes in traffic
- Need to adapt workflows to each technology being used
- Time is spent on labor intensive steps and decisions that distract developers from building and deploying new features

=> Downtime and erosion of governance


php
Today
About Features Pricing
Download
The only planner you'll ever need
A tailored daily rundown of your schedule, articles to read, reminders and tasks - at your fingertips.
Learn more
4,567,321 visitors
php

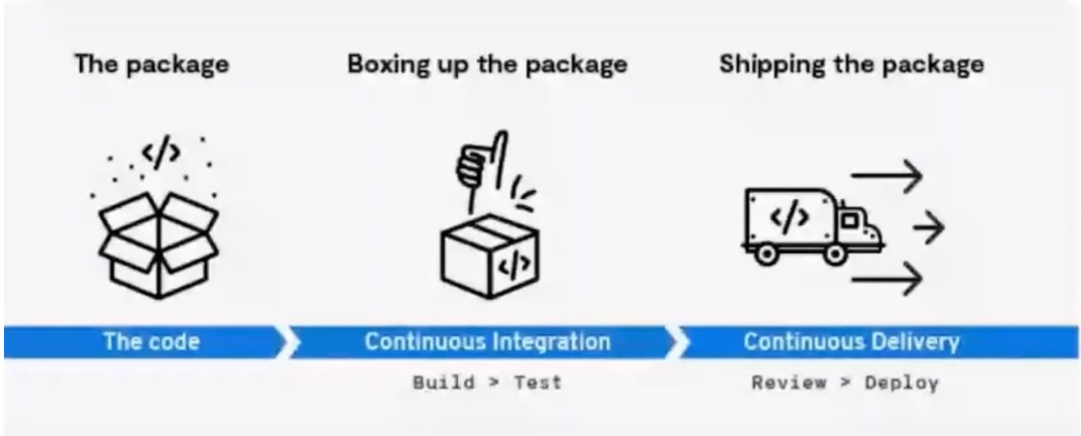
DevOps to the Rescue

Features:

- Collaboration
- Automation
- Continuous Improvement

Advantages:

- Improved Communication and Collaboration
- Shorter Development Cycles
- Reduced Deployment Failures
- Faster delivery of features
- More time to innovate



What about Infrastructure?

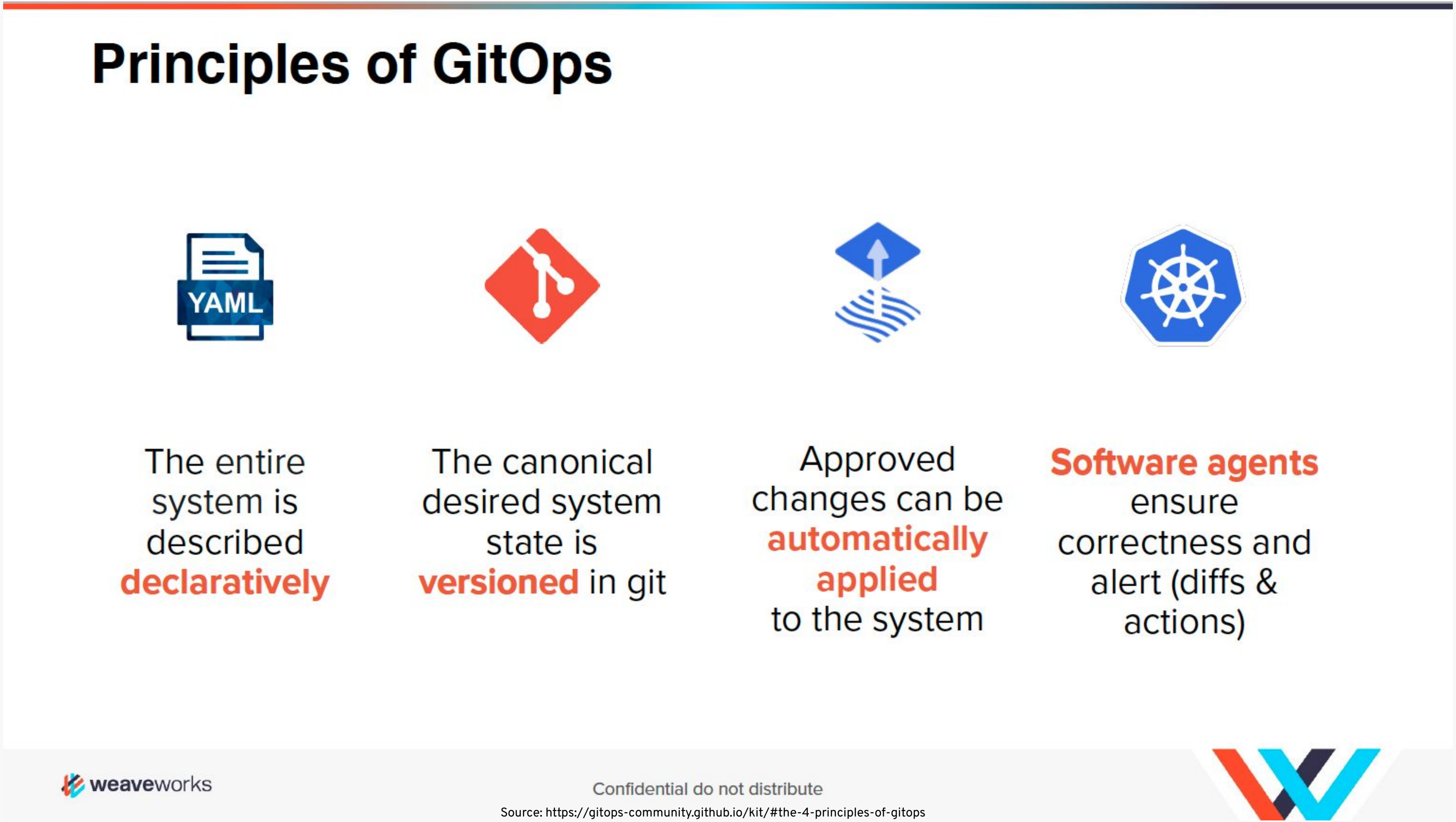
- Physical Hardware
- Virtual Machines
- First Generation Configuration Management
- Second Generation Configuration Management
- Cloud Infrastructure
- Infrastructure as Code

What is GitOps?

GitOps defined

An operational framework that combines DevOps best practices for application development (version control, collaboration, CI/CD tooling, etc) and applies it to infrastructure automation

> "Compare the running state of our system with the desired state - continuously - and whenever these get out of sync, force the running state to converge to the desired state"
>
> – Alexis Richardson
> CEO, WeaveWorks



Principles of GitOps

The entire system is described **declaratively**

The canonical desired system state is **versioned** in git

Approved changes can be **automatically applied** to the system

Software agents ensure correctness and alert (diffs & actions)

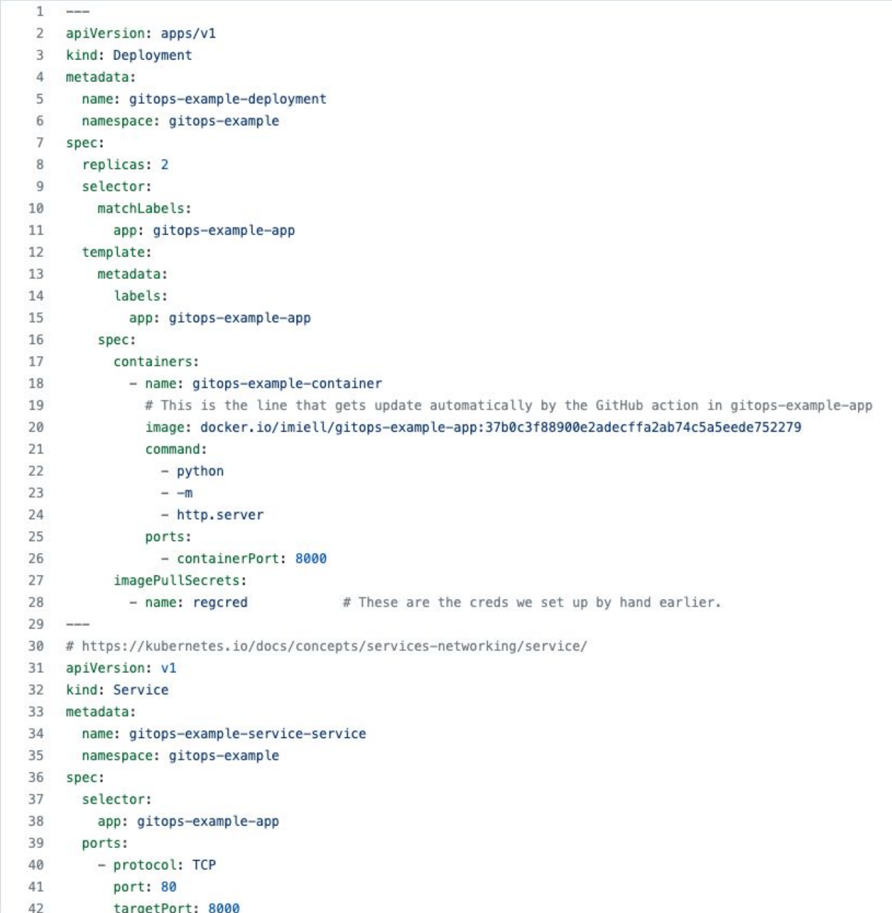
weaveworks

Confidential do not distribute

Source: https://gitops-community.github.io/kit/#the-4-principles-of-gitops

Principles of GitOps

A system managed by GitOps must have its desired state expressed declaratively.

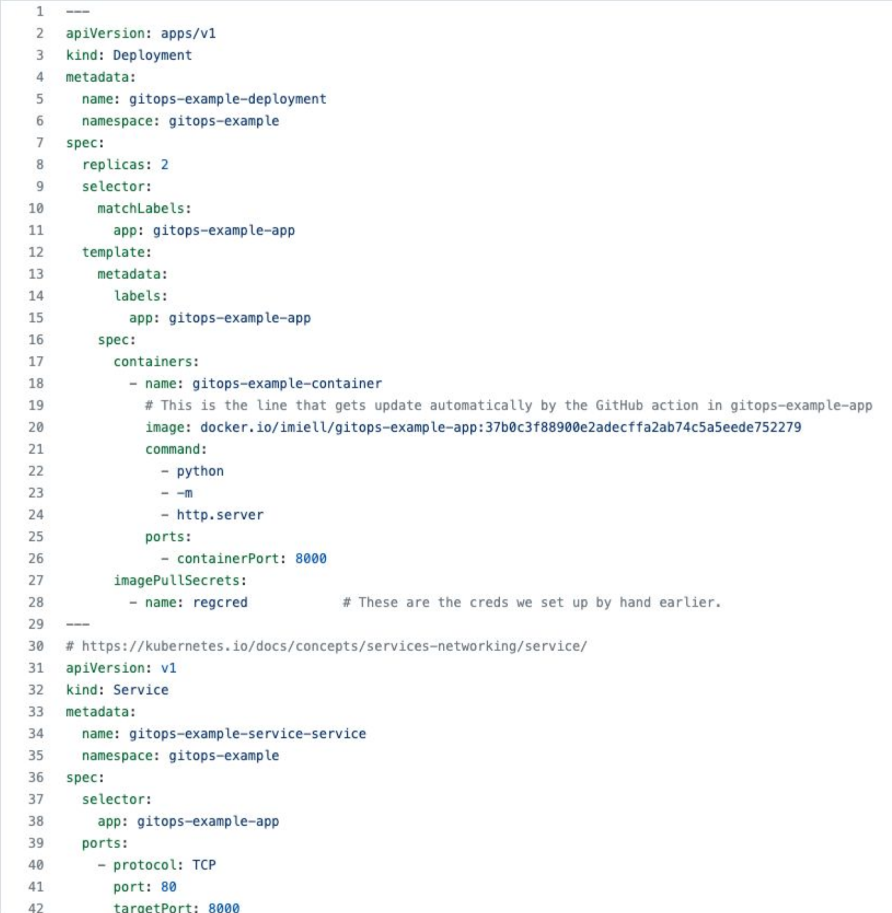


```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitops-example-deployment
  namespace: gitops-example
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gitops-example-app
  template:
    metadata:
      labels:
        app: gitops-example-app
    spec:
      containers:
        - name: gitops-example-container
          # This is the line that gets update automatically by the GitHub action in gitops-example-app
          image: docker.io/imiell/gitops-example-app:37b0c3f88900e2adecffa2ab74c5a5eede752279
          command:
            - python
            - -m
            - http.server
          ports:
            - containerPort: 8000
      imagePullSecrets:
        - name: regcred      # These are the creds we set up by hand earlier.
---
# https://kubernetes.io/docs/concepts/services-networking/service/
apiVersion: v1
kind: Service
metadata:
  name: gitops-example-service-service
  namespace: gitops-example
spec:
  selector:
    app: gitops-example-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

Principles of GitOps

System:

- One or more runtime environments consisting of resources under management
- The management agents within each runtime
- Policies for controlling access and management of repositories, deployments, runtimes

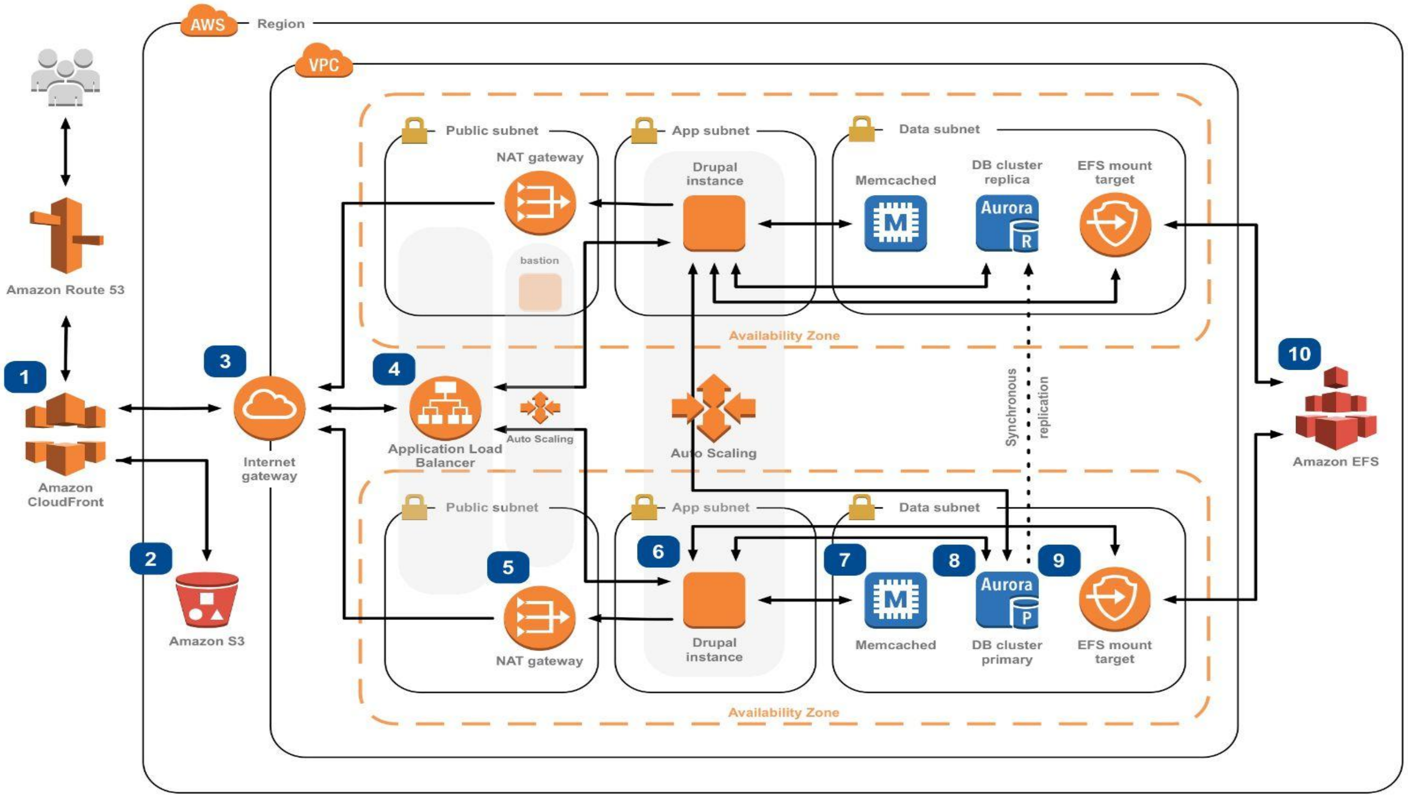


```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitops-example-deployment
  namespace: gitops-example
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gitops-example-app
  template:
    metadata:
      labels:
        app: gitops-example-app
    spec:
      containers:
        - name: gitops-example-container
          # This is the line that gets update automatically by the GitHub action in gitops-example-app
          image: docker.io/imiell/gitops-example-app:37b0c3f88900e2adecffa2ab74c5a5eede752279
          command:
            - python
            - -m
            - http.server
          ports:
            - containerPort: 8000
      imagePullSecrets:
        - name: regcred            # These are the creds we set up by hand earlier.
---
# https://kubernetes.io/docs/concepts/services-networking/service/
apiVersion: v1
kind: Service
metadata:
  name: gitops-example-service-service
  namespace: gitops-example
spec:
  selector:
    app: gitops-example-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

Drupal Hosting

Running Drupal on AWS

Drupal is a free, open-source web content management platform. The Drupal community is one of the largest open-source communities in the world with more than 1,000,000 passionate developers, designers, trainers, strategists, coordinators, editors, and sponsors. This reference architecture enables you to deploy a scalable and highly available Drupal site on AWS.



1 Use **Amazon CloudFront** to deliver static and dynamic content.

2 Use **Amazon S3** to store static content such as media, downloadable files, etc.

3 Attach an **internet gateway** to your VPC to enable communication between Amazon EC2 instances in your VPC and the internet.

4 Use an **Application Load Balancer** to distribute web traffic across an Auto Scaling Group of Amazon EC2 instances in multiple Availability Zones.

5 Create a **NAT gateway** in each public subnet to send traffic to the internet gateway from private subnets (app and data) in your VPC.

6 Run your Drupal site using an **Auto Scaling group** of **Amazon EC2 instances**. Install the latest versions of Drupal, Apache web server, PHP 7, and OPcache. Then, build an Amazon Machine Image (AMI) that the Auto Scaling group launch configuration can use to launch new instances in the group.

7 If database access patterns are read-heavy, consider using a Drupal plugin that takes advantage of a caching layer such as **Amazon ElastiCache for Memcached** in front of the database layer to cache frequently accessed data.

8 Run your database layer in **Amazon RDS** using either Aurora or MySQL to simplify your database administration.

9 After creating an Amazon EFS file system, create **mount targets.** Mount the file system on your Drupal Amazon EC2 instances in each Availability Zone in your VPC.

10 Use **Amazon EFS** for Drupal instances to access your shared, unstructured Drupal data such as PHP files, config, themes, plugins, etc.

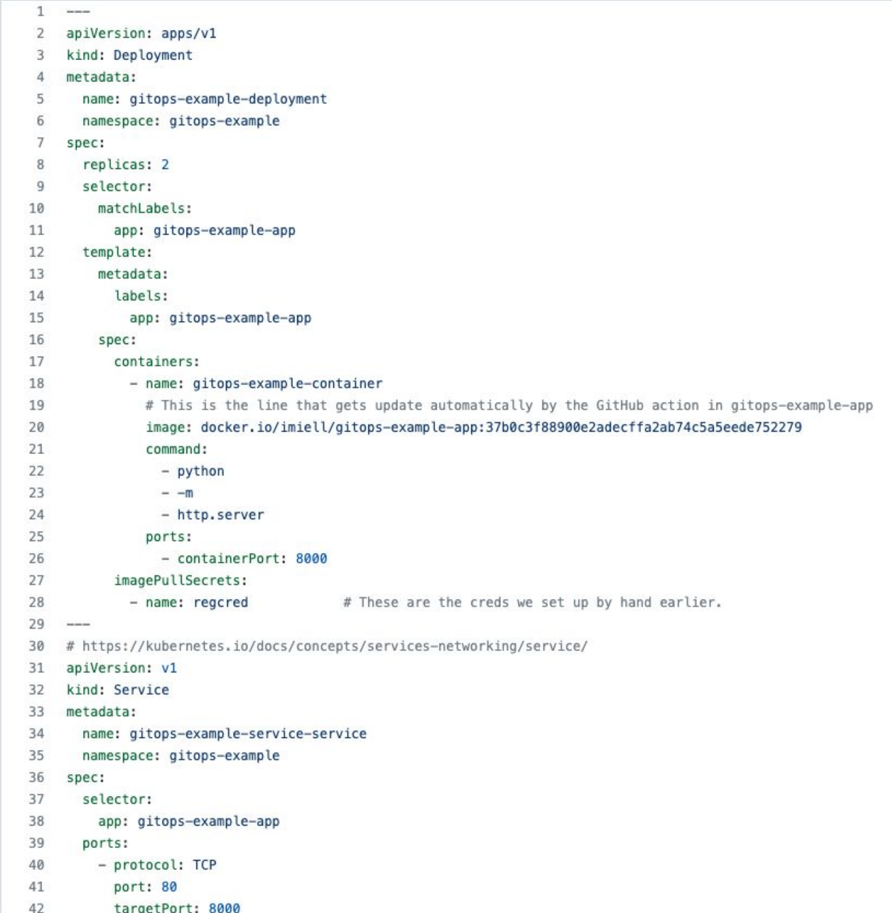
amazon web services

AWS Reference Architectures

Principles of GitOps

System:

- One or more runtime environments consisting of resources under management
- The management agents within each runtime
- Policies for controlling access and management of repositories, deployments, runtimes



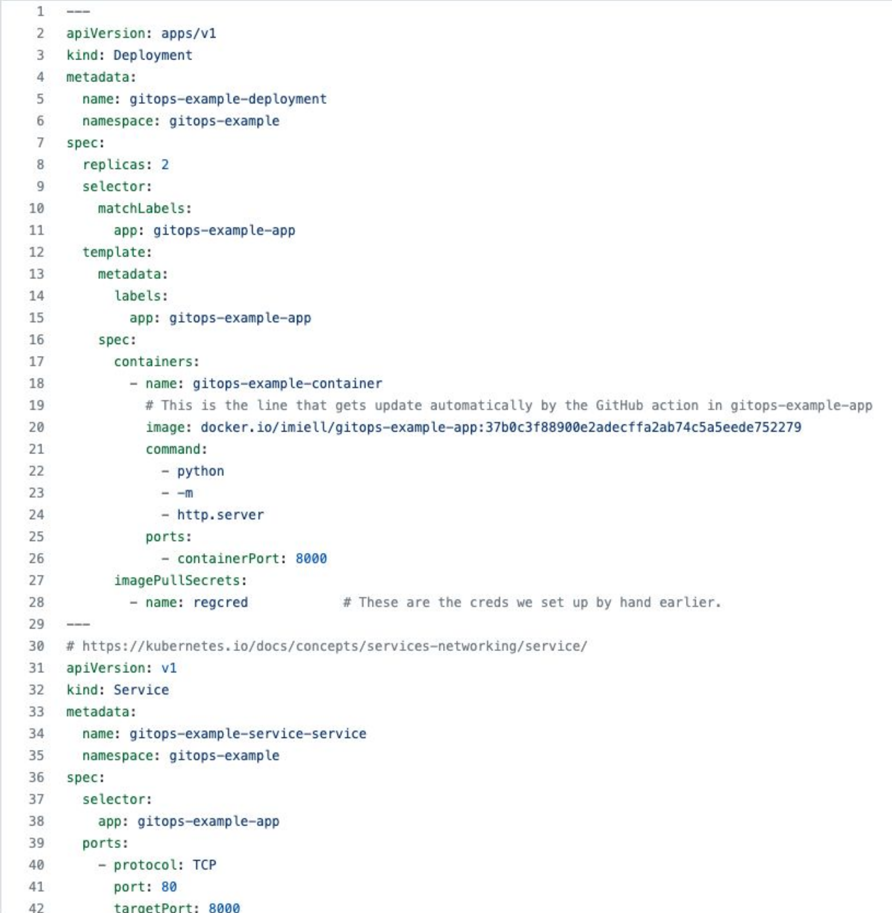
```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitops-example-deployment
  namespace: gitops-example
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gitops-example-app
  template:
    metadata:
      labels:
        app: gitops-example-app
    spec:
      containers:
        - name: gitops-example-container
          # This is the line that gets update automatically by the GitHub action in gitops-example-app
          image: docker.io/imiell/gitops-example-app:37b0c3f88900e2adecffa2ab74c5a5eede752279
          command:
            - python
            - -m
            - http.server
          ports:
            - containerPort: 8000
      imagePullSecrets:
        - name: regcred      # These are the creds we set up by hand earlier.
---
# https://kubernetes.io/docs/concepts/services-networking/service/
apiVersion: v1
kind: Service
metadata:
  name: gitops-example-service-service
  namespace: gitops-example
spec:
  selector:
    app: gitops-example-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

Principles of GitOps

A system managed by GitOps must have its desired state expressed declaratively.

Desired State:

The aggregate of all configuration data that is sufficient to recreate the system so that instances of the system are behaviourally indistinguishable.

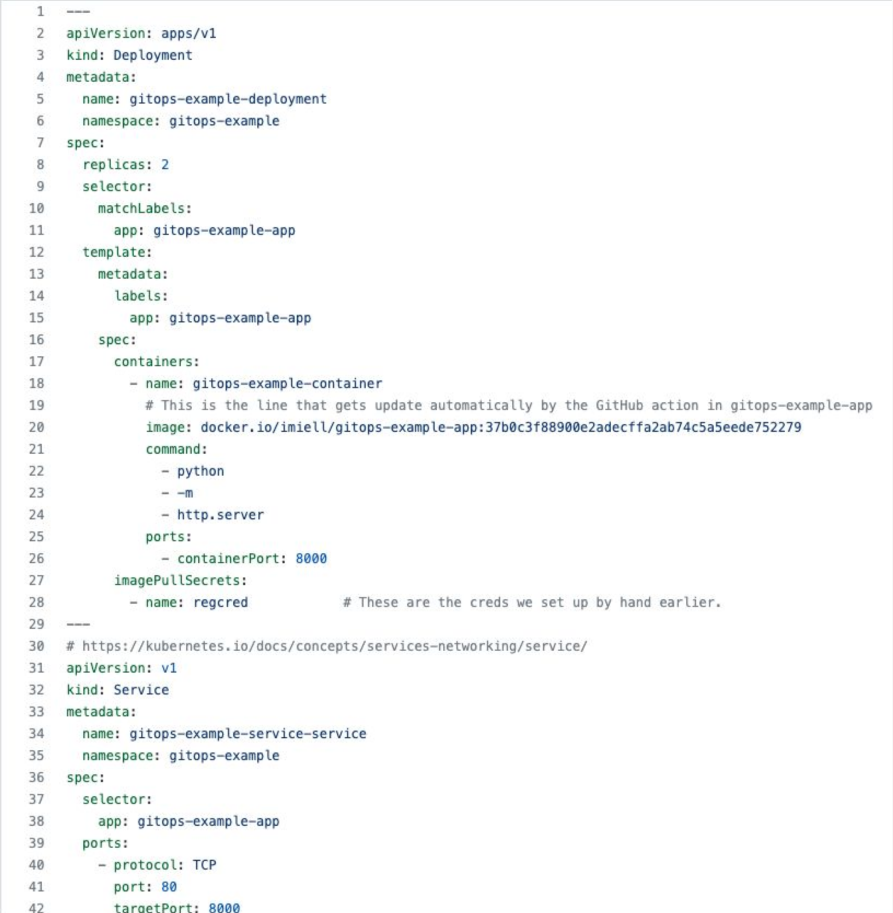


```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitops-example-deployment
  namespace: gitops-example
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gitops-example-app
  template:
    metadata:
      labels:
        app: gitops-example-app
    spec:
      containers:
        - name: gitops-example-container
          # This is the line that gets update automatically by the GitHub action in gitops-example-app
          image: docker.io/imiell/gitops-example-app:37b0c3f88900e2adecffa2ab74c5a5eede752279
          command:
            - python
            - -m
            - http.server
          ports:
            - containerPort: 8000
      imagePullSecrets:
        - name: regcred            # These are the creds we set up by hand earlier.
---
# https://kubernetes.io/docs/concepts/services-networking/service/
apiVersion: v1
kind: Service
metadata:
  name: gitops-example-service-service
  namespace: gitops-example
spec:
  selector:
    app: gitops-example-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

Principles of GitOps

A system managed by GitOps must have its desired state expressed declaratively.

Declarative Description:
A configuration that describes the desired operating state of a system without specifying procedures for how that state will be achieved.



```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitops-example-deployment
  namespace: gitops-example
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gitops-example-app
  template:
    metadata:
      labels:
        app: gitops-example-app
    spec:
      containers:
        - name: gitops-example-container
          # This is the line that gets update automatically by the GitHub action in gitops-example-app
          image: docker.io/imiell/gitops-example-app:37b0c3f88900e2adecffa2ab74c5a5eede752279
          command:
            - python
            - -m
            - http.server
          ports:
            - containerPort: 8000
      imagePullSecrets:
        - name: regcred            # These are the creds we set up by hand earlier.
---
# https://kubernetes.io/docs/concepts/services-networking/service/
apiVersion: v1
kind: Service
metadata:
  name: gitops-example-service-service
  namespace: gitops-example
spec:
  selector:
    app: gitops-example-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

Principles of GitOps

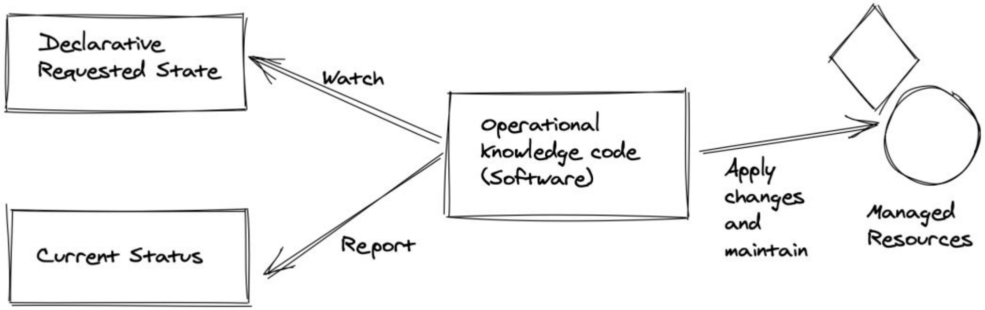
Desired state is stored in a way that enforces immutability, versioning and retains a complete version history.

Control system should be capable of

- Versioning
- Authentication
- Audit log
- Meta-data about why a change has been made

Principles of GitOps

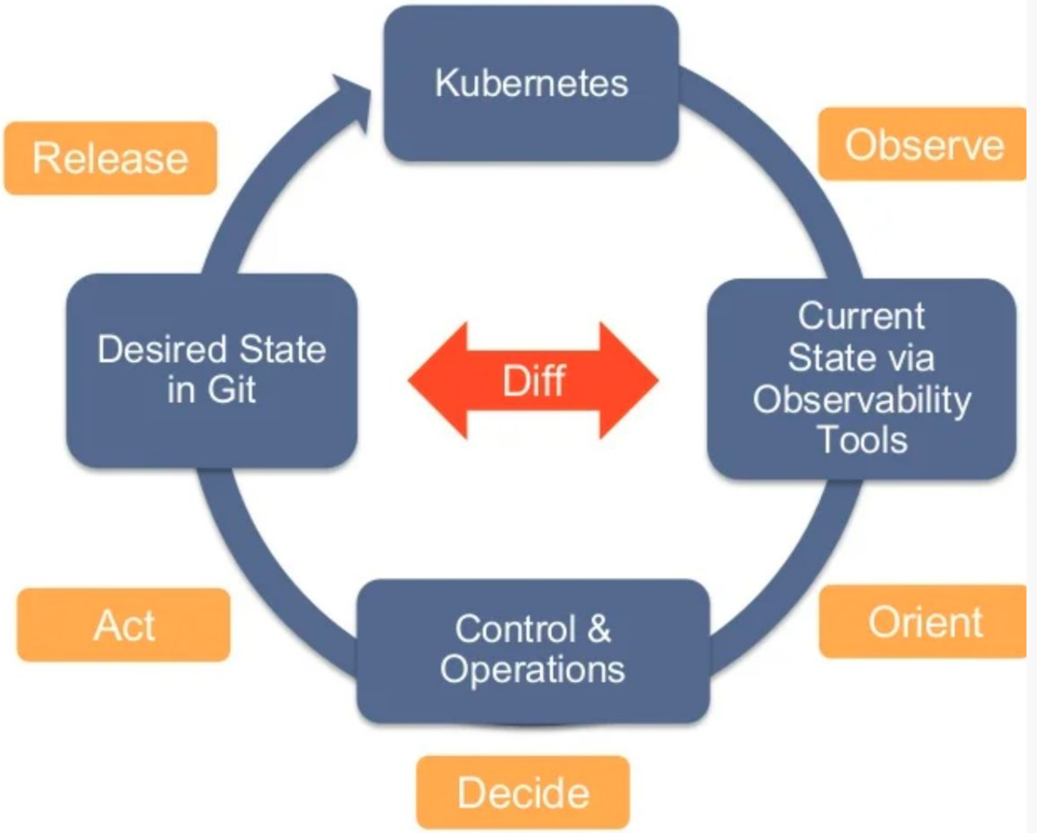
Software agents automatically apply the desired state declarations from the source.

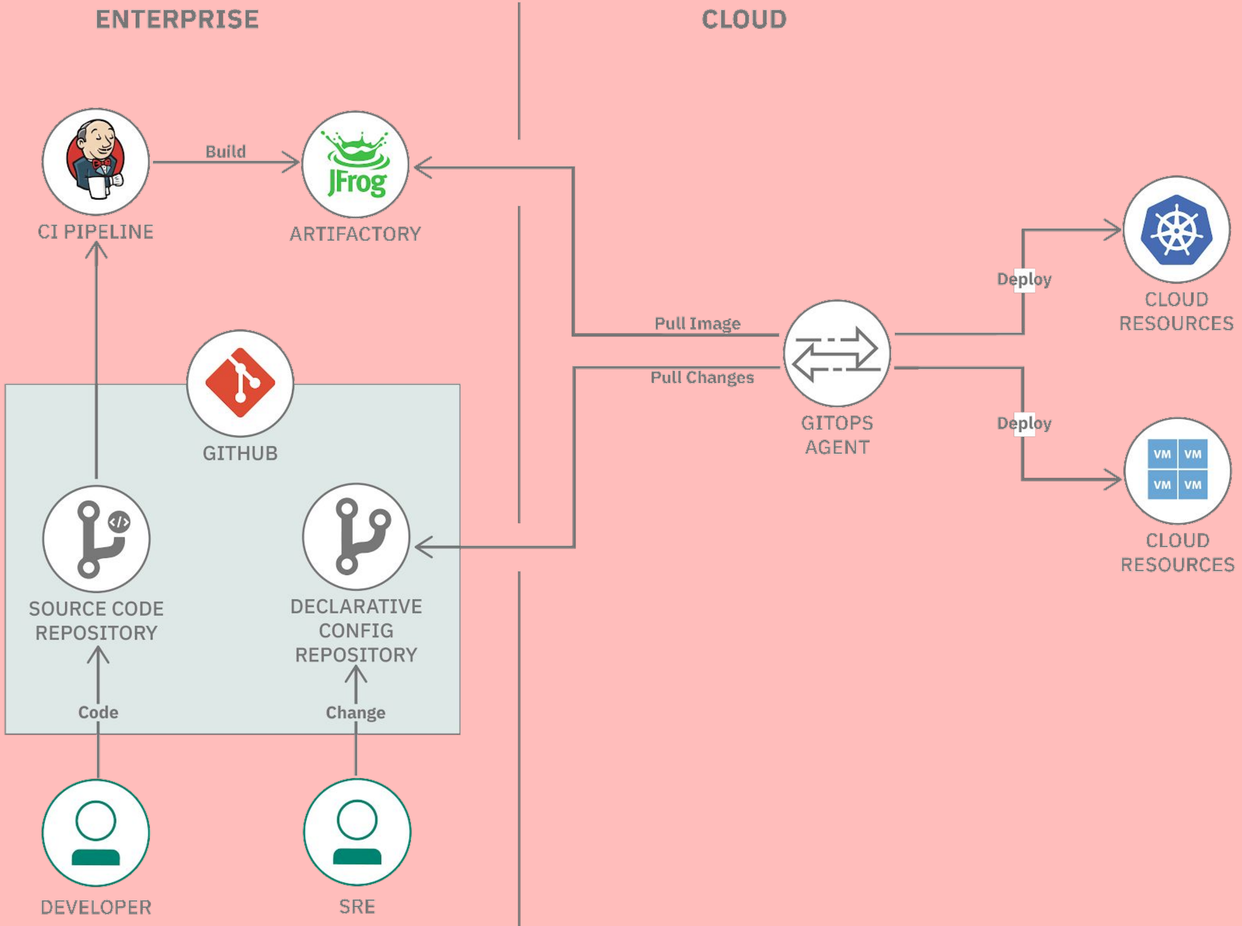

Declarative Requested State
Watch
Operational Knowledge code (Software)
Apply changes and maintain
Managed Resources
Current Status
Report

Source:
https://www.cncf.io/wp-content/uploads/2021/07/CNCF_Operator_WhitePaper.pdf

Principles of GitOps

Software agents continuously observe actual system state and attempt to apply the desired state.





Source: https://developer.ibm.com/blogs/gitops-best-practices-for-the-real-world/

How is GitOps different from DevOps?

GitOps vs DevOps

DevOps is about cultural change and providing a way for development teams and operations teams to work together collaboratively.

GitOps gives you tools and a framework to take DevOps practices, like collaboration, CI/CD, and version control, and apply them to infrastructure automation and application deployment.

Why GitOps?

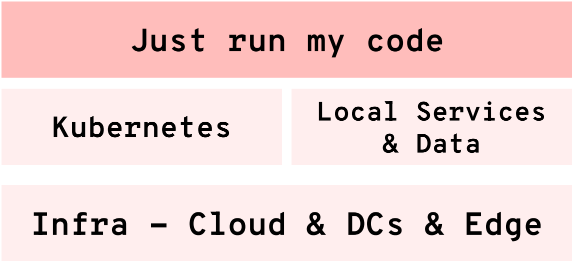
Why GitOps?

- Greater deployment frequency
- Reduced recovery time from failures
- Less error prone
- Precise operations across teams
- Improved Security and Compliance

How does GitOps benefit developers?

So you can

focus on what you love:


Just run my code

Kubernetes

Local Services & Data

Infra – Cloud & DCs & Edge

> "The key point of the developer experience is ‘git-push,'"

– Alexis Richardson
CEO, WeaveWorks

Developer Advantages

- Uses familiar tools and processes
- Easier onboarding
- Experimentation
- Faster iterations

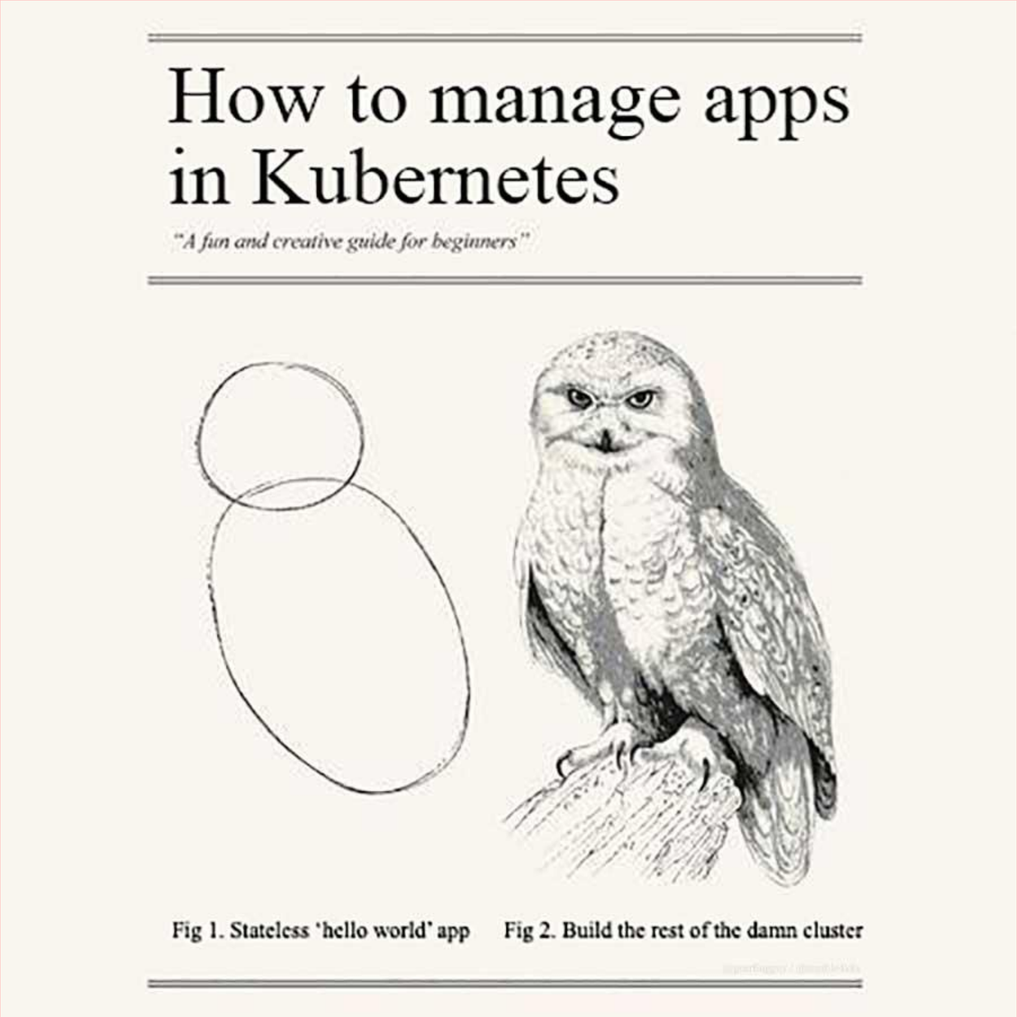
Disadvantages of GitOps

Disadvantages

- Need to be cloud native
- Proliferation of Git repositories
- Auditing can become complex

Disadvantages

- Lacks input validation
- Does not address persistent application data
- Doesn't solve centralized secret management



How to manage apps in Kubernetes

"A fun and creative guide for beginners"

Fig 1. Stateless 'hello world' app

Fig 2. Build the rest of the damn cluster

Complexity

Tools deprecate, and people on- and off-board – often those essential to understanding how the pieces fit together

Platform.sh

> "Platform.sh is, in a sense, a "not having to build and maintain a complicated GitOps pipeline"-as-a-Service."

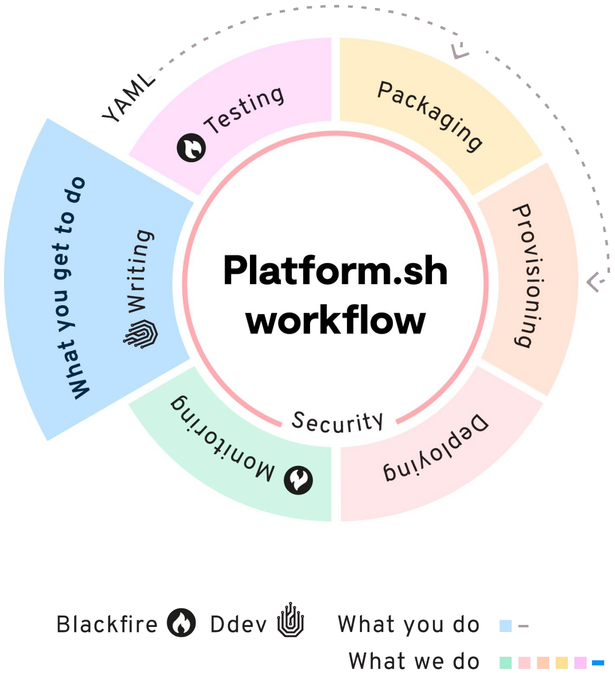
– Larry Garfield
Staff Engineer at TYPO3

platform.sh

Platform.sh

is focused on solving the complexity of managing application delivery for organizations.

We’re a unified, secure, enterprise-grade platform for building, running and scaling fleets of websites and applications.



Resources

- https://opengitops.dev/
- https://gitops-community.github.io/
- https://about.gitlab.com/topics/gitops/
- https://www.weave.works/blog/category/gitops/
- https://www.gitops.tech/
- https://www.atlassian.com/git/tutorials/gitops
- https://eks.handson.flagger.dev/

Thank you!

platform.sh

SCAN ME

Paul Gilzow

Developer Relations Engineer,
Platform.sh

paul.gilzow@platform.sh

platform.sh