

GitOps First Steps

Ian Miell

Twitter: @ianmiell

ian.miell@gmail.com



Part I - Introduction

- **About Me**
- **Course Overview**
- **Discussion**
- **What is GitOps in One Slide**

GitOps and Me

- **Worked in GitOps in Banks**
- **Consulting, working with smaller companies to make GitOps happen**
- **Blog on GitOps and related subjects at <https://zwischenzugs.com>**
- **Written books on Git, Docker, Terraform, Bash**

Course Overview

- **Part I - Introduction**
- **Part II - Why GitOps?**
- **Part III - Defining GitOps**
- **Break**
- **Part IV - Key GitOps Tools**
- **Part V - Demo**
- **Part VI - Implementation Challenges**

Discussion

- **How do you deploy software now?**
- **What are the challenges/bottlenecks?**
- **What tools do you use, and what are their characteristics?**
- **What do you understand by GitOps?**

What is GitOps in One Slide

- **Can be defined in various ways, but at core:**
 - **1) Everything as Code**
 - **2) Declarative system operation definition**
 - **3) Control Loop**
- **Packaging of older ideas (eg DevOps, Scripting, Versioned Source Control, configuration management, Make) into an opinionated movement.**

Part II - Why GitOps?



What Problems Does GitOps Solve?

- **Various deployment anti-patterns**
 - **Deployment by hand**
 - **State in a spreadsheet**
 - **Pipeline by GUI**
 - **'It's all up here'**
- **Anti-patterns cost money**
- **GitOps (like DevOps) reduces cost of deployment**

‘Deployment By Hand’ Anti-Pattern

- **Place code into environments through a manual process**
- **May be sped up by scripting**
- **Documents with deployment commands in them are still not uncommon**

‘State in a Spreadsheet’ Anti-Pattern

- **State of system stored in a spreadsheet**
- **Often associated with firewall or proxy rule management**
- **Also often associated with traditional change control systems, eg ServiceNow**

‘Pipeline By GUI’ Anti-Pattern

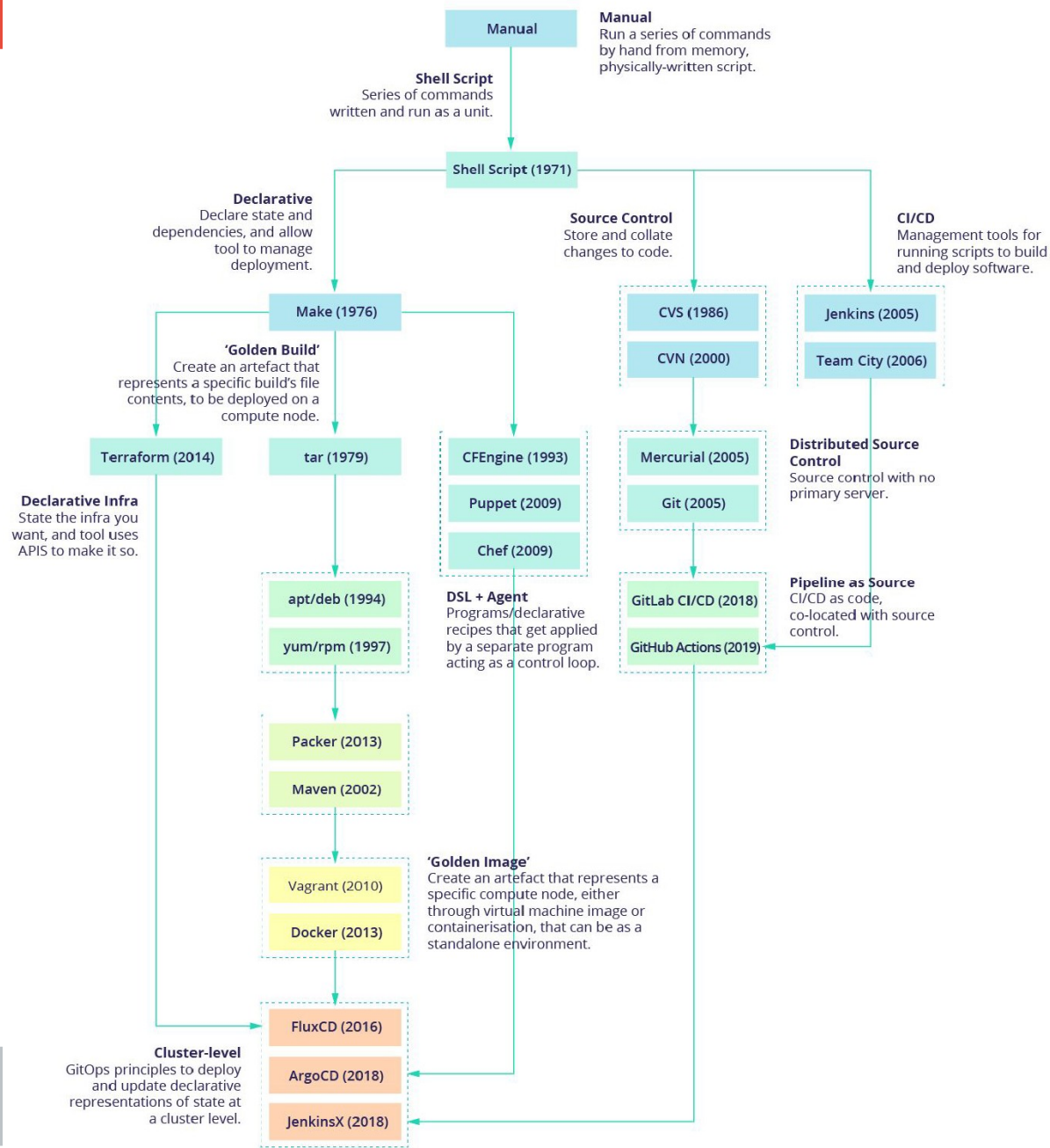
- **Rise of DevOps brought Jenkins et al to the fore**
- **Pipeline was step forward in automation but often stateful through configuration by GUI**

Anti-Pattern Outcomes

- **Uncertain Desired State**
- **Uncertain Actual State**
- **Gap Between Desired and Actual State**
- **Control Challenges**



Part III - Defining GitOps



What Exactly Is GitOps?

- **Official Weave definition (2017)**
 - **An operating model for Kubernetes and other cloud native technologies, providing a set of best practices that unify deployment, management and monitoring for containerized clusters and applications.**
 - **A path towards a developer experience for managing applications; where end-to-end CI/CD pipelines and Git workflows are applied to both operations, and development.**

The Three Key Concepts of GitOps?

- **Audited source control and configuration management**
- **Declarative data definition of system configuration**
- **Reconciling control loop for configuration management**



Declarative

- **Declarative**
 - **Code that declares the desired configuration statically, rather than dynamically based on switching procedures**
- **eg**
 - **Make**
 - **Puppet**
 - **YAML**
 - **JSON**

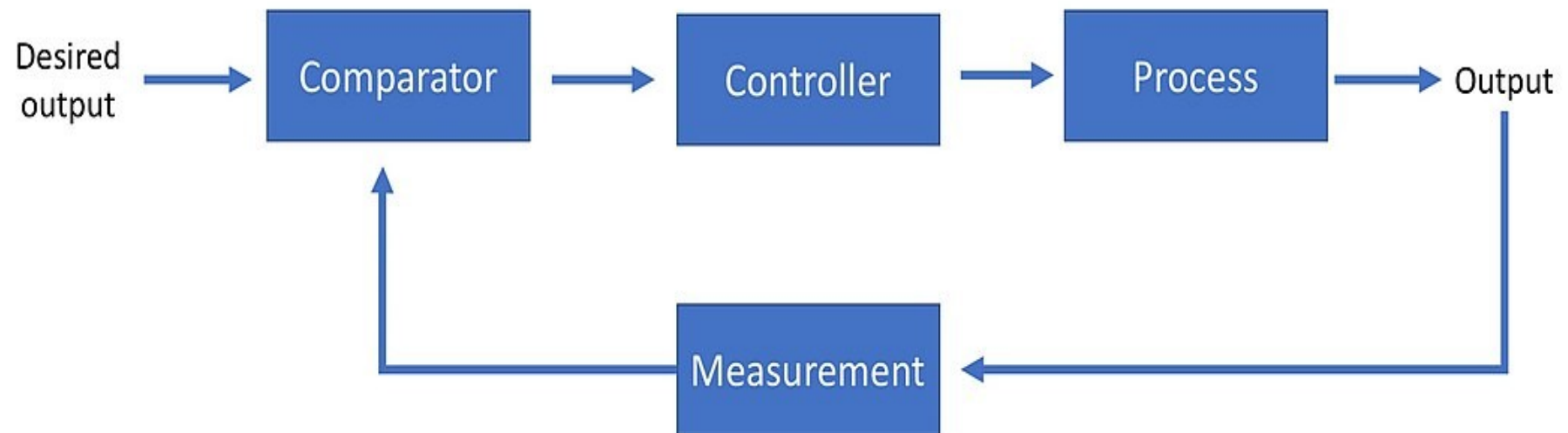
Source Controlled (I)

- **SCM (Source Control Management) tool should provide:**
 - **Easy and cheap branching of code**
 - **Change history integrity**
 - **Change sharing protocols**
 - **Integration with identity systems**

Source Controlled (I)

- **Git is the *de facto* tool here**
 - **Branching is $O(1)$**
 - **SHA hashing of commit contents**
- **GitHub, GitLab et al have evolving standards and enterprise integrations**
- **Extremely widespread adoption**
 - **Stack Overflow: 83% of devs on GitHub**

Control Loop (I)



•

Control Loop (II)

- **A control loop:**
 - **Checks whether system is in desired state**
 - **If it is not, effects changes to get into desired state**
- **eg thermostat**
- **‘Controller’ a familiar concept to Kubernetes users - K8s name itself means ‘governor’, a similar engineering concept around speed of system**

How GitOps Helps (I)

- **These concepts, together, improve:**
 - **Reliability**
 - **Automated, zero-touch, self-correcting systems**
 - **Fewer ad hoc, unmonitored system changes**
 - **Reduced bespoke logic**
 - **Auditability and accountability**
 - **Full audit history via source control**

How GitOps Helps (II)

- **Which deliver benefits:**
 - **Improved productivity and lower cost of system ownership**
 - **Less time spent debugging systems in unknown state**
 - **Less time maintaining systems with recurring problems**
 - **Cheaper and simpler workflow/approval systems**
 - **Easier to implement automated testing**

Part IV - Key GitOps Tools

- **Fast-growing space**
 - **Kubernetes (Deployment Platforms)**
 - **Terraform (Infrastructure Provisioning)**
 - **ArgoCD/FluxCD ('Pull' tools)**
 - **Kustomize / kubectl ('Push' tools)**
 - **JenkinsX (Curated GitOps products)**

Kubernetes

- **Runs Docker (or industry standard) containers**
- **Deployment configurable by code (YAML/JSON)**



Terraform

- **Specifies and maintains infrastructure setup**
- **Declarative language (HCL)**
- **Source-code friendly**

ArgoCD / FluxCD

- **Two similar ‘control loop’ solutions**
- **Within Kubernetes clusters, these applications track git repositories and apply changes to cluster**
- **Projects’ efforts are consolidating to <https://github.com/argoproj/gitops-engine>**

JenkinsX

- **Monolithic ‘all-in-one’ GitOps solution**
- **Nothing (much) to do with Jenkins CI**

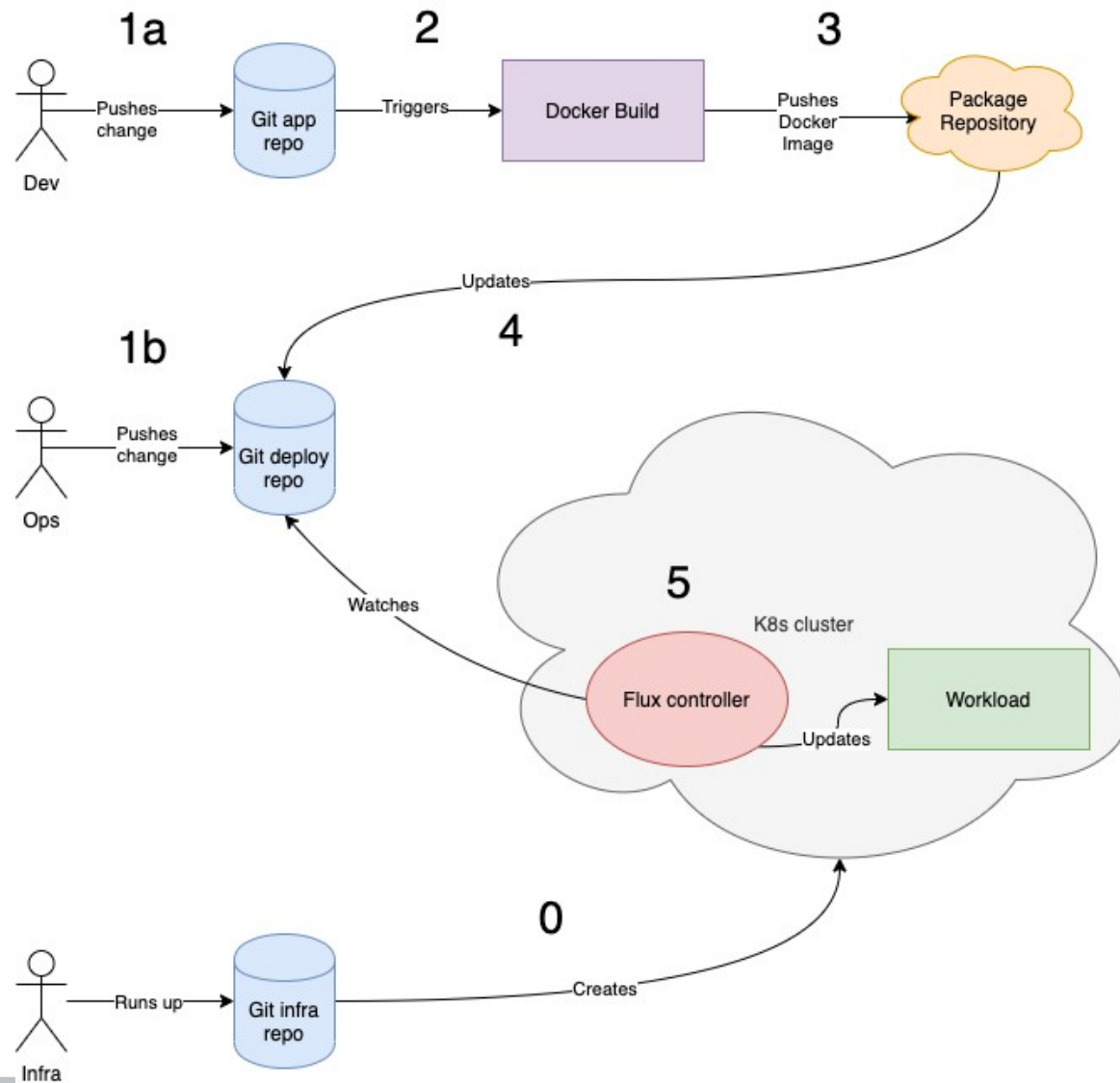
Part V - Demo



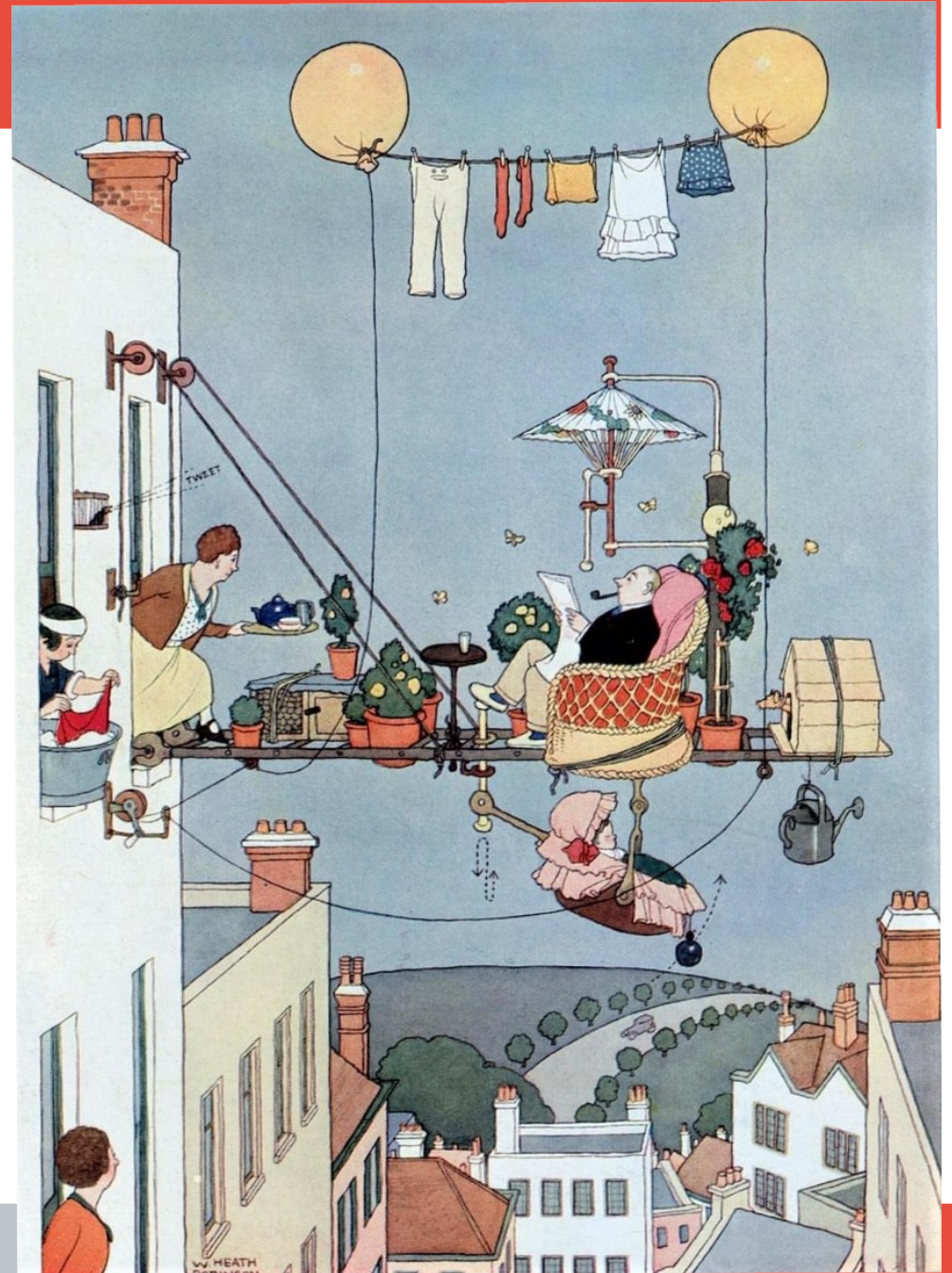
Demo Resources

- <https://github.com/ianmiell/gitops-example>
- **Uses:**
 - **GitHub**
 - **GitHub Actions**
 - **Terraform (optional)**
 - **Kubernetes**
 - **Shell**
 - **Flux**
 - **Docker**

Demo Overview



Part VI - GitOps Implementation Challenges



Challenges - Technical

- **Many new technologies to learn/master**
- **Git, Terraform, Kubernetes (none of these are trivial)**
- **Many small decisions need to be made when building up your GitOps capability**

Challenges - Cultural

- **Need to change delivery mindset**
- **'Hero' culture of 'logging in and fixing' needs to be challenged**
- **The deployment process/code is king and discipline needs to be maintained**
- **Onboarding new teams to this way of working can generate a lot of friction if they are not prepared**

Challenges - Business

- **GitOps work is very front-loaded**
- **Can take a long time to 'bed in' good practices within an organization before seeing a return on investment**
- **Benefits are not immediately obvious to non-technical people**
- **Emergent area: there is no 'safe, proven choice' for a GitOps approach**

Challenges - Solutions

- **Technical:** Invest in spreading expertise and gaining experience across teams. Ensure documentation and pairing etc used to pass on knowledge.
- **Cultural:** Invest early in outreach, bring staff with you. Point out opportunities for growth and development.
- **Business:** Be realistic, don't over-promise, and measure costs and compare old/new costs to demonstrate business value.

Thank you!

Ian Miell

Twitter: @ianmiell

ian.miell@gmail.com

