



# GitOps School



# What is your understanding about deployment history?

- a) Curious but not familiar
- a) Delivered projects and want to learn more
- b) Used Shell scripts, Makefiles
- c) Used tools like Ansible, Chef, Puppet, Terraform, Saltstack, etc..
- d) Helm, ksonnet, kustomize, etc..

# GitOps School

## AGENDA



**DAY 1**

- **Introduction to Case Study, challenges**
- **Introduction to GitOps**
- **Understanding VCS and Github, Github Actions, Workflows**
- **Lab Activities**



**DAY 1**

- **GitOps Tools/operators**
- **ArgoCD Flow, scenarios**
- **Implementing GitOps using ArgoCD, Jenkins, Docker, Github, Kubernetes**
- **Lab Activities**
- **Insights of Github webhooks, Runners**



**DAY 2**

- **GitOps Push vs Pull based approach**
- **Understanding the scenarios for pull based model**
- **GitOps Practices and culture shift**
- **Lab activity**



**DAY 2**

- **Advantages and Key Functionalities**
- **Cloud Native Security using Kubescape**
- **Results and benefits achieved by the client after adopting to GitOps.**





## **The customer:**

**GOAppsFlyer**, the market leader in mobile marketing and analytics industry, caters to over 14,000 customers. Besides 1000 microservices, its architecture operates over 250,000 cloud resources, including VM instances and Kubernetes clusters. The AppsFlyer Engineering team of over 400 engineers is organized into smaller groups called Squads, with each Squad having complete autonomy.

## **Challenges:**

Manual and time-consuming build and deployment process

Developer autonomy was needed, but with greater alignment

Risk of the platform was becoming the developer's bottleneck

Operational challenges

Post-production security guardrails






# HOW WE MADE IT SIMPLE for the GoAppsFlyer?

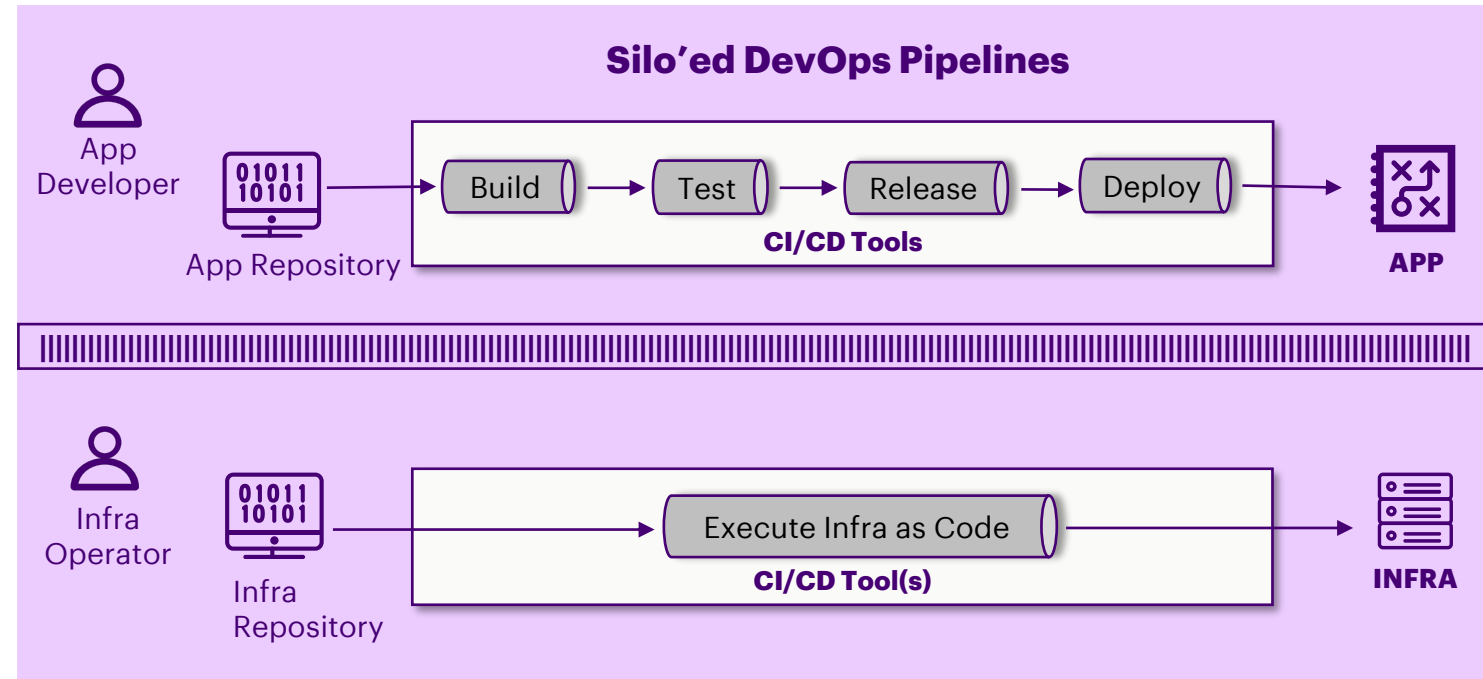
# **GITOPS**

# WHY GITOPS?

**GitOps makes “software-defined-everything” possible !**

**GitOps helps address key challenges arising from Dev & Ops automation silos, such as:**

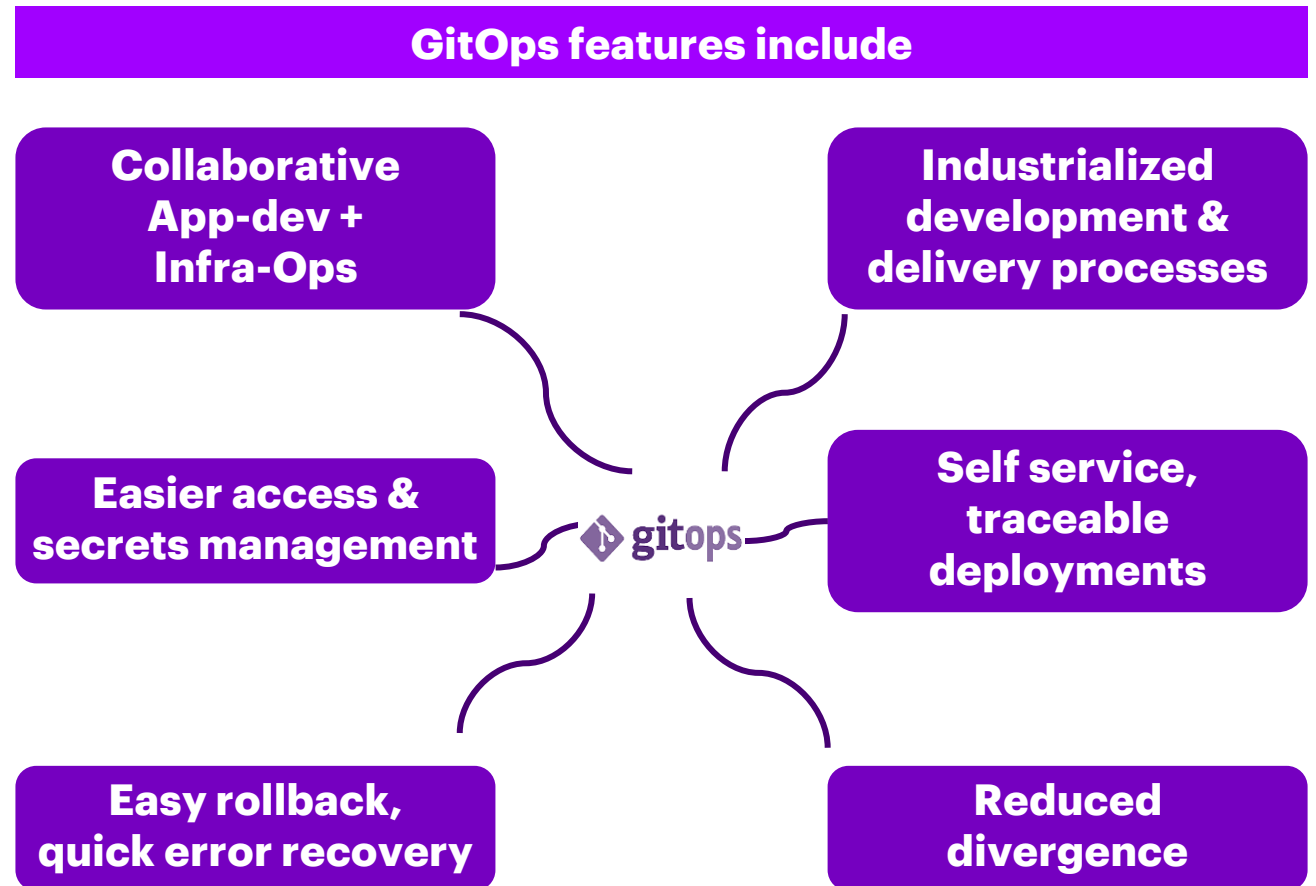
-  Decoupled CI/CD
-  Multiple Source Control Repositories
-  Too many tools
-  Multiple integration endpoints
-  Inconsistent rollback procedures



# HOW GITOPS HELPS...

**GitOps is a framework to optimize IT operations across application, infrastructure, cloud & security domains.**

**It offers a way of implementing true CI/CD for cloud native applications. With GitOps, you have a single source of truth & Git triggered workflows to achieve a software defined “desired state of system”.**





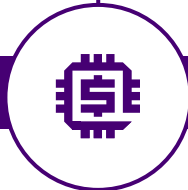
# BENEFITS

---

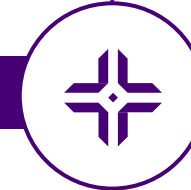
Lean toolchain  
translating to  
reduction in cost  
& administrative  
effort



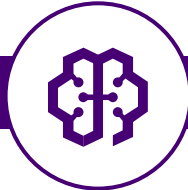
Transparent  
actions through  
pipeline stages



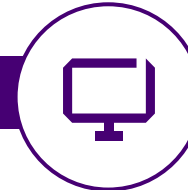
Easier rollbacks  
leading to lesser  
risk & downtimes



Collaborative  
changes to  
declarative  
infrastructure



Simplified  
traceability,  
auditing &  
compliance  
reporting



Faster time to  
market & better  
Access Control



# GUIDING PRINCIPLES

**GitOps == IaC + VCS + CI/CD**  
(Infra as Code + Version/Source Control System + CI/CD)



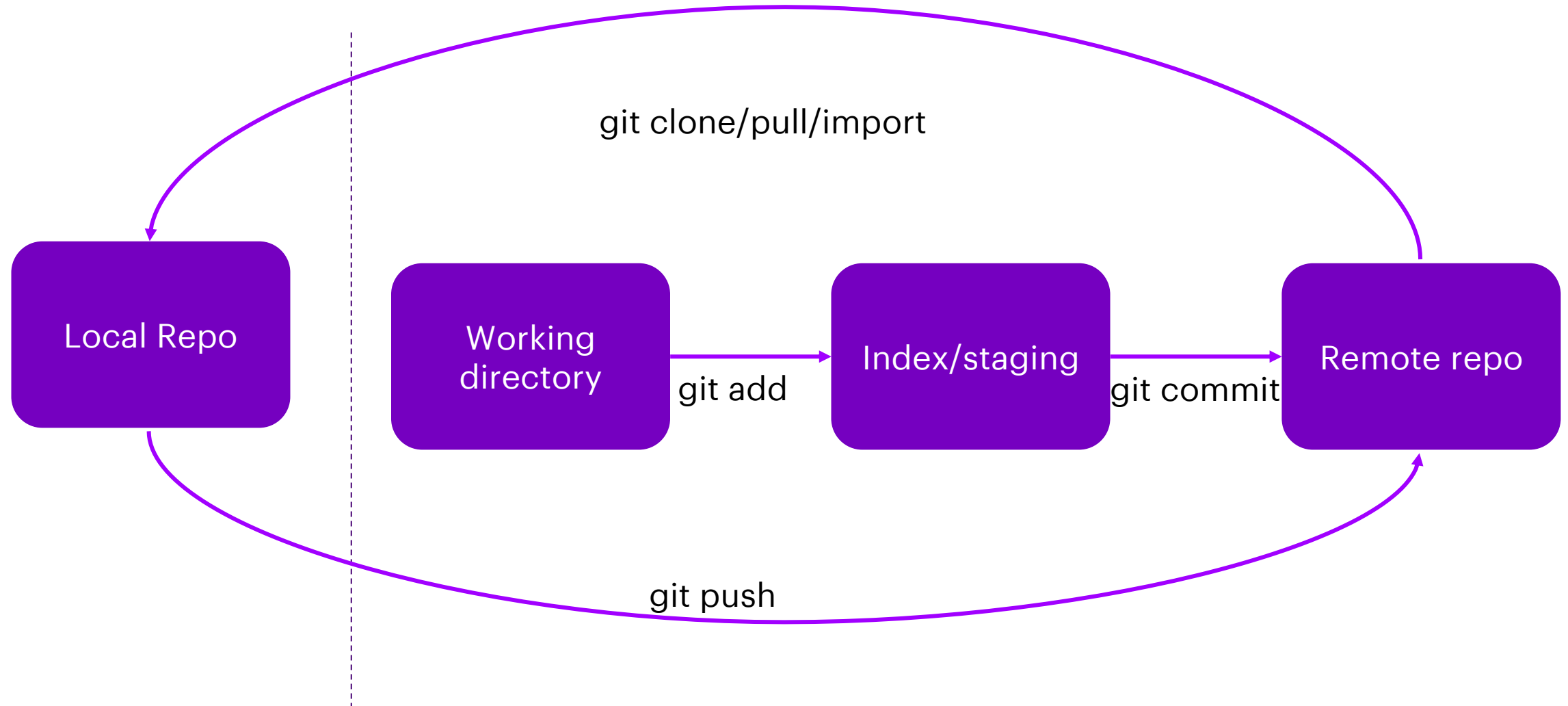
# Video – Version Controlling System

Click on the below link to understand features of Version Controlling System and Git

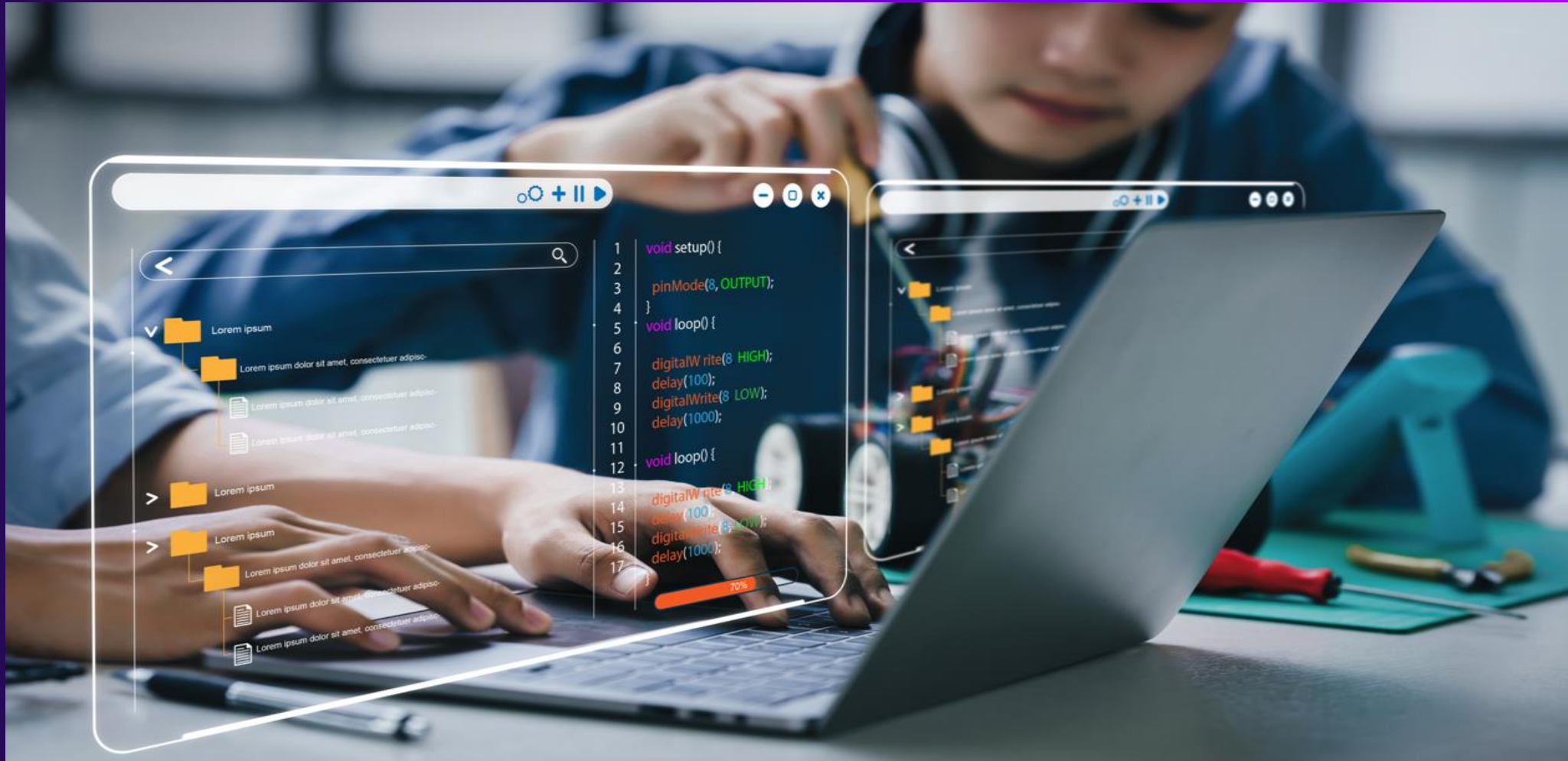
[Introduction to Version Controlling System and Git](#)



# SCM/VCS using Git-Github

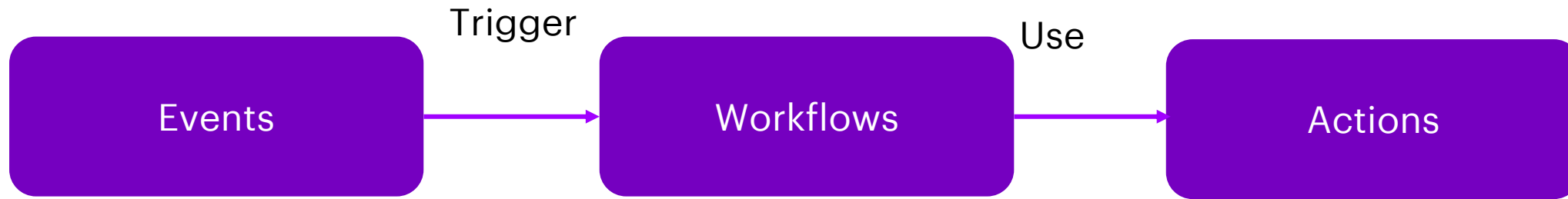


# Lab Activity 1



# GitHub Actions

# Github Actions



# Github Actions - Events

- GitHub triggered events:  
Push,pull\_request,public
- Scheduled events : schedule
- Manually triggered

```
1 # .github/workflows/weekly-radar.yml
2 name: Weekly Radar
3
4 on:
5   schedule:
6     - cron: 0 12 * * 1
7
8 jobs:
9
10  weekly_radar:
11    name: Weekly Radar
12    runs-on: ubuntu-latest
13    steps:
14
15    - name: weekly-radar
```





# Github Actions - Workflows

- Workflows are like pipelines
- Codify useful and customized processes
- .yaml syntax
- .github/workflows

```
1 # .github/workflows/build.yml
2 name: Node CI
3
4 on: [push]
5
6 jobs:
7   build:
8
9     runs-on: ${ matrix.os }
10
11     strategy:
12       matrix:
13         node-version: [8.x, 10.x, 12.x]
14         os: [macos-latest, windows-latest, ubuntu-18.04]
15
16     steps:
17     - uses: actions/checkout@v1
18     - name: Use Node.js ${ matrix.node-version }
19       uses: actions/setup-node@v1
20       with:
21         node-version: ${ matrix.node-version }
22     - name: npm install, build, and test
23       run: |
24         npm ci
25         npm run build --if-present
26         npm test
27     env:
28       CI: true
```



# Github Actions - Workflows

- Workflow files glue together Existing actions in sequence
- Listen for particular events
- Then run pre-existing actions
- Or shell scripts

```
1 # .github/workflows/build.yml
2 name: Node CI
3
4 on: [push]
5
6 jobs:
7   build:
8
9     runs-on: ${ matrix.os }
10
11     strategy:
12       matrix:
13         node-version: [8.x, 10.x, 12.x]
14         os: [macos-latest, windows-latest, ubuntu-18.04]
15
16     steps:
17     - uses: actions/checkout@v1
18     - name: Use Node.js ${ matrix.node-version }
19       uses: actions/setup-node@v1
20       with:
21         node-version: ${ matrix.node-version }
22     - name: npm install, build, and test
23       run: |
24         npm ci
25         npm run build --if-present
26         npm test
27     env:
28       CI: true
```



# Github Actions - Actions

- Actions runs in VMs (Linux,Win,Mac)
- Or Docker on Linux VM
- Logs streaming & artifacts
- Secret store with each repository or organization

```
1 # .github/workflows/build.yml
2 name: Node CI
3
4 on: [push]
5
6 jobs:
7   build:
8
9     runs-on: ${ matrix.os }
10
11     strategy:
12       matrix:
13         node-version: [8.x, 10.x, 12.x]
14         os: [macos-latest, windows-latest, ubuntu-18.04]
15
16     steps:
17     - uses: actions/checkout@v1
18     - name: Use Node.js ${ matrix.node-version }
19       uses: actions/setup-node@v1
20       with:
21         node-version: ${ matrix.node-version }
22     - name: npm install, build, and test
23       run: |
24         npm ci
25         npm run build --if-present
26         npm test
27     env:
28       CI: true
```



# CASE STUDY – GOAPPSFLYER

William just joined the team for his client GoAppsFlyer . His team develops many microservices and deploys them on cloud instances as well Kubernetes clusters. They are looking for solution for improvement towards manual and time-consuming build and deployment process, address operational as well security challenges for their services.

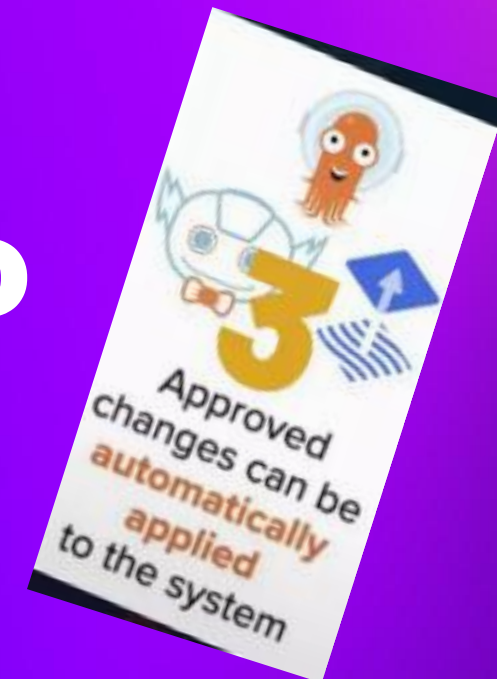
To begin, William has developed his first Java based application for GoAppsFlyer client. Let's assist him in building the artifacts using github platform.



# Lab Activity 2



# What do I need to do GitOps?





# GitOps Operators/Tools

# GitOps Operators

**Flux CD**



- Flux is an open source project that was originally developed by Weave Works, the same company that coined the term GitOps.
- FluxCD is a declarative deployment automation tool which is controlled by means of its CLI, fluxctl
- Is a CNCF sandbox project.
- <https://fluxcd.io>



# GitOps Operators

## Jenkins-X



- Jenkins X is a CI/CD solution for modern cloud applications on Kubernetes, an opensource opinionated way to do continuous delivery with Kubernetes, natively.
- Is a free open source tool with enterprise support offered by Cloudbees.
- It can handle all parts of the GitOps process, it's an all-in-one solution.
- [What is Jenkins X? | Jenkins X - Cloud Native CI/CD Built On Kubernetes \(jenkins-x.io\)](https://jenkins-x.io)

# GitOps Operators

**Gitlab-EE**



- GitLab is a single application for the entire DevOps lifecycle and serves as a collaboration platform that empowers stakeholders to weigh in on the code production process.
- GitOps isn't just about the code, it's about the collaboration, and GitLab enables every team to work in a single platform
- It's the first attempt to integrate GitOps CD in Gitlab CI, implemented as Kubernetes operator running inside the cluster.
- It's based on Argo GitOps Engine.
- [What is GitOps? | GitLab](#)

# GitOps Operators

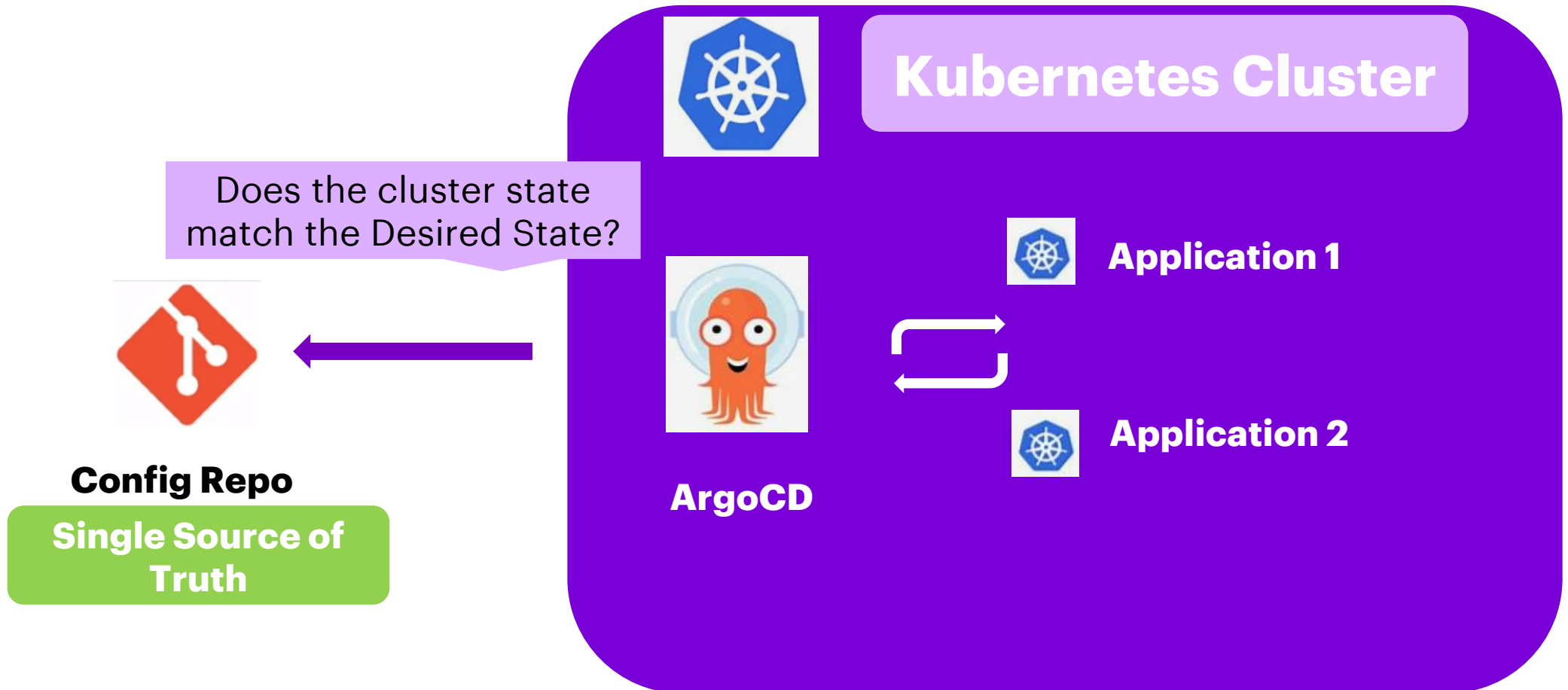
## Argo CD



- Argo CD is a declarative GitOps continuous delivery tool for Kubernetes and one of the 4 Argo Projects ( a set of Kubernetes-native tools for deploying and running jobs and applications).
- Includes a command line tool and an excellent graphical user interface.
- Focused on enterprise use-cases ( security, RBAC, SSO, centralized multi-tenancy multi-cluster management)
- [Argo CD - Declarative GitOps CD for Kubernetes \(argo-cd.readthedocs.io\)](https://argo-cd.readthedocs.io)

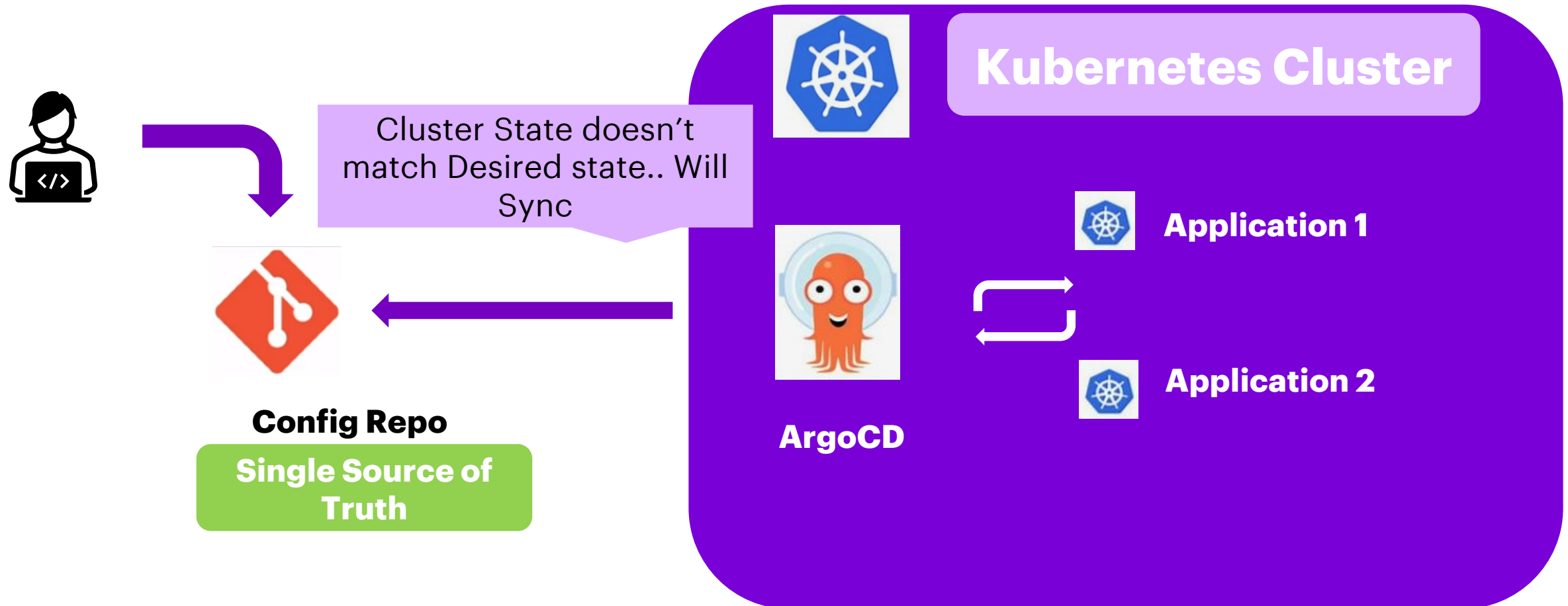
# ArgoCD

- ✓ **Declarative GitOps Tool**
- ✓ **Kubernetes Native Continuous Deployment**



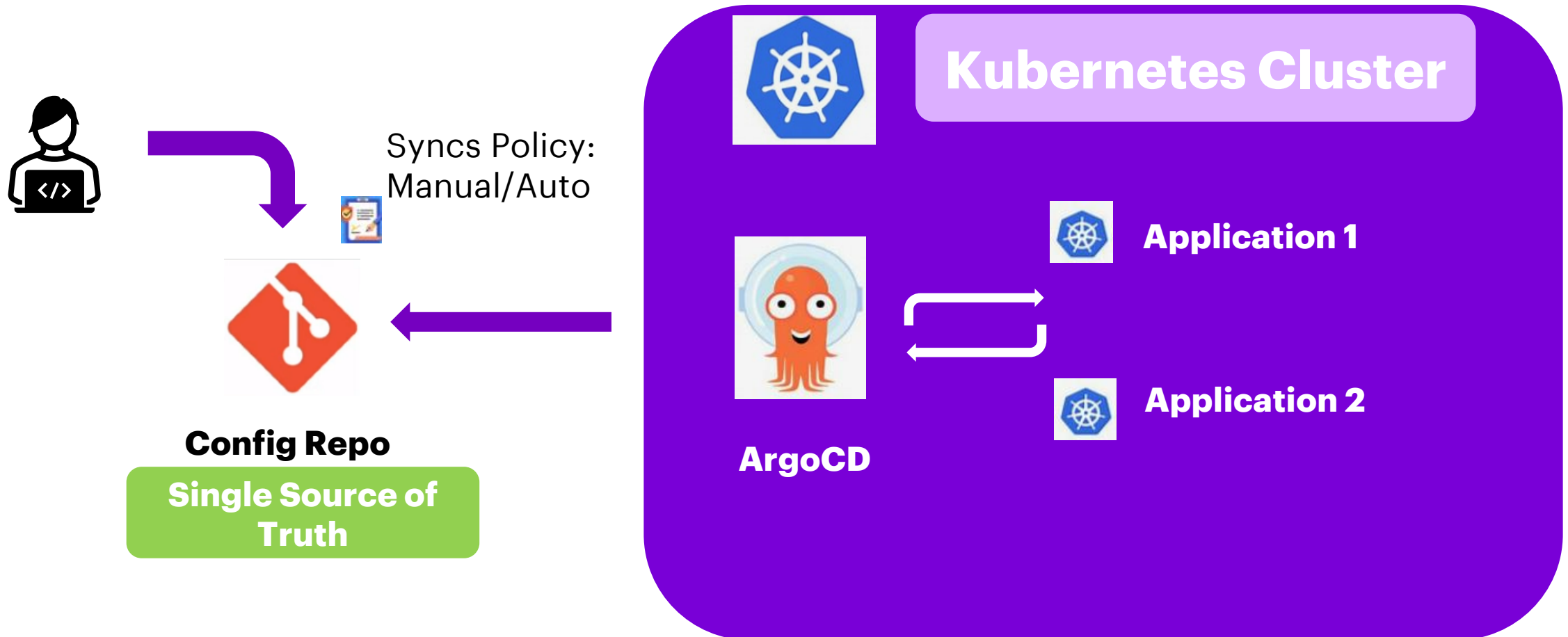
# ArgoCD

- ✓ **Declarative GitOps Tool**
- ✓ **Kubernetes Native Continuous Deployment**

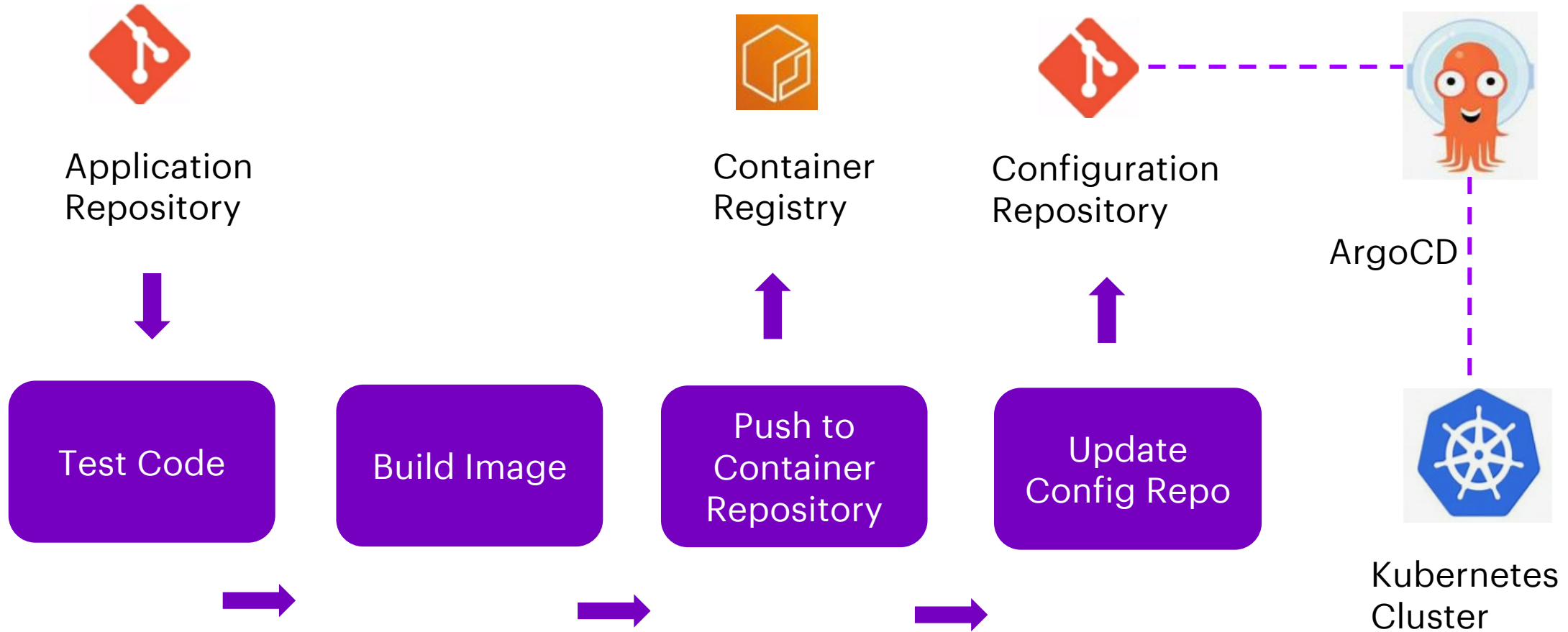


# ArgoCD

## ✓ ArgoCD Syncs Configs in Git to Kubernetes



# Working with ArgoCD to achieve GitOps



# Video – Jenkins and Kubernetes

Click on the below link to understand features of Jenkins and Kubernetes

[Jenkins Introduction](#)

[Introduction to Containers and Kubernetes](#)



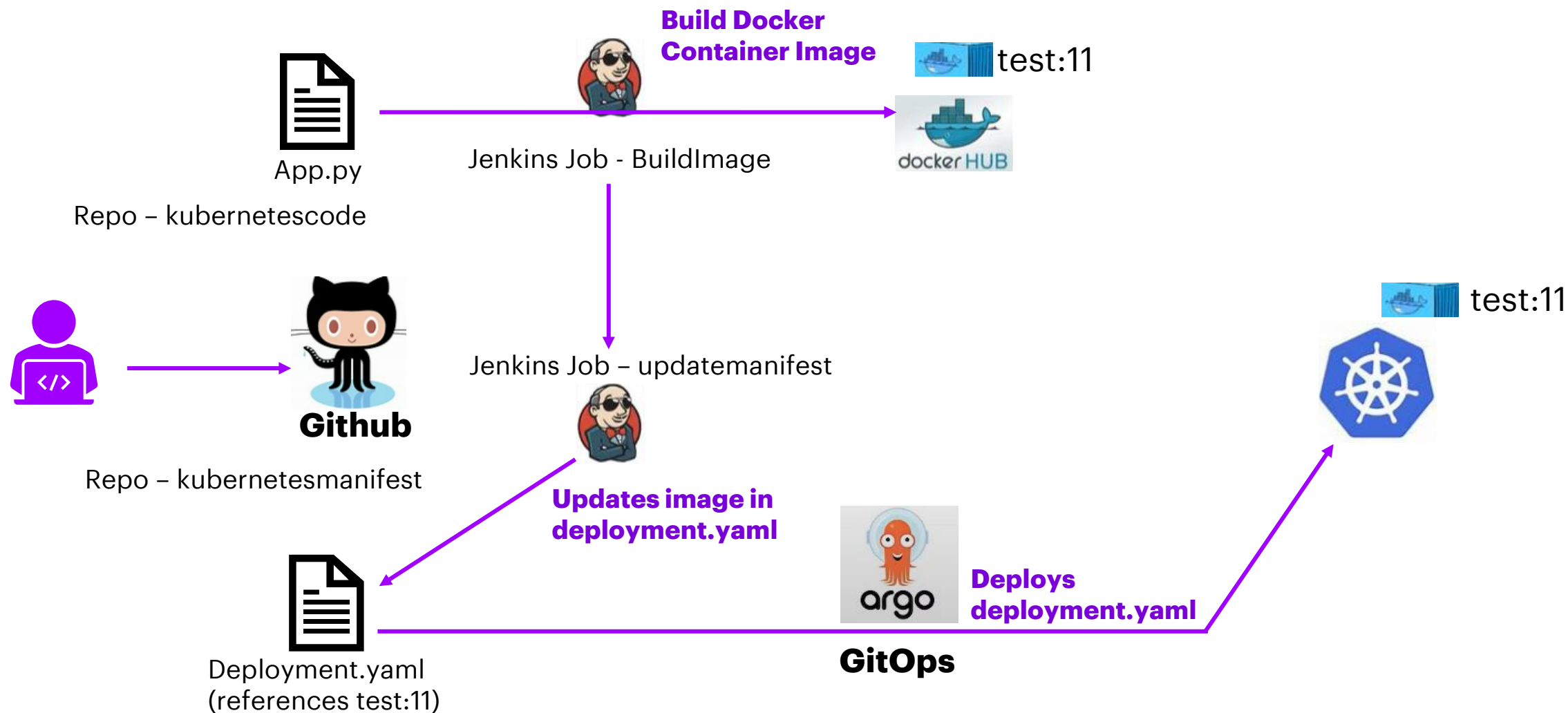
# CASE STUDY – GOAPPSFLYER

William just joined the team for his client GoAppsFlyer . His team develops many microservices and deploys them on cloud instances as well Kubernetes clusters. They are looking for solution for improvement towards manual and time-consuming build and deployment process, address operational as well security challenges for their services.

William has developed python based application for his client. To deploy his application he written kubernetesmanifest deployment (YAML) code as well and kept in github repos. Let's assist him in implementing GitOps using Jenkins, Docker, ArgoCD.



# GitOps Scenario Flow



# Lab Activity 3



# Thank you