

Getting Started with Argo CD

WHAT IS ARGO CD



Steve Buchanan

CONTAINER / CLOUD ARCHITECT

@buchatech | www.buchatech.com



Overview



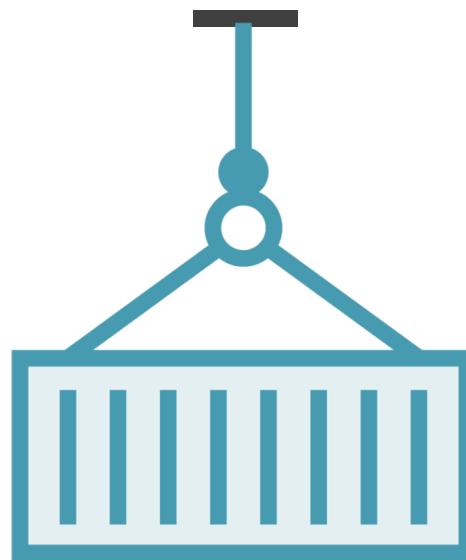
- Understanding Containers and Kubernetes**
- Understanding Helm and Kustomize**
- Understanding GitOps core concepts**
- History and Overview of Argo CD**
- Understanding Argo CD core concepts and architecture**
- Understanding Supported Tooling with Argo CD**



Understanding Containers and Kubernetes



Understanding Containers



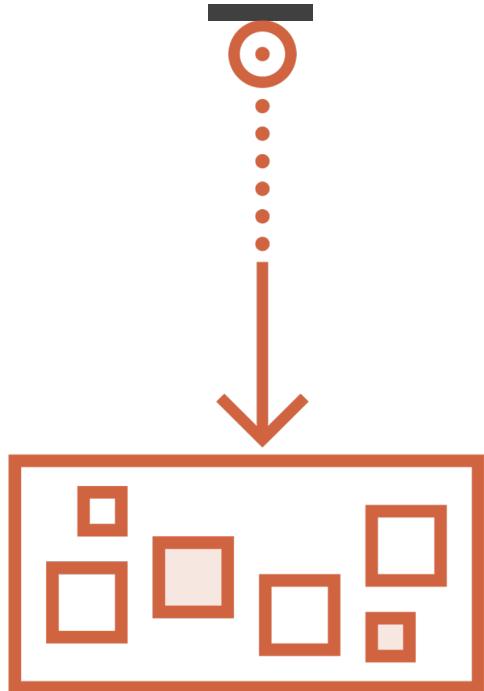
Containers are an abstraction of the application layer in an isolated user-space instance

Containers share the OS kernel storage and networking from the host they run on

Containers can be thought of like the core components needed for an application running as a process



Understanding Containers

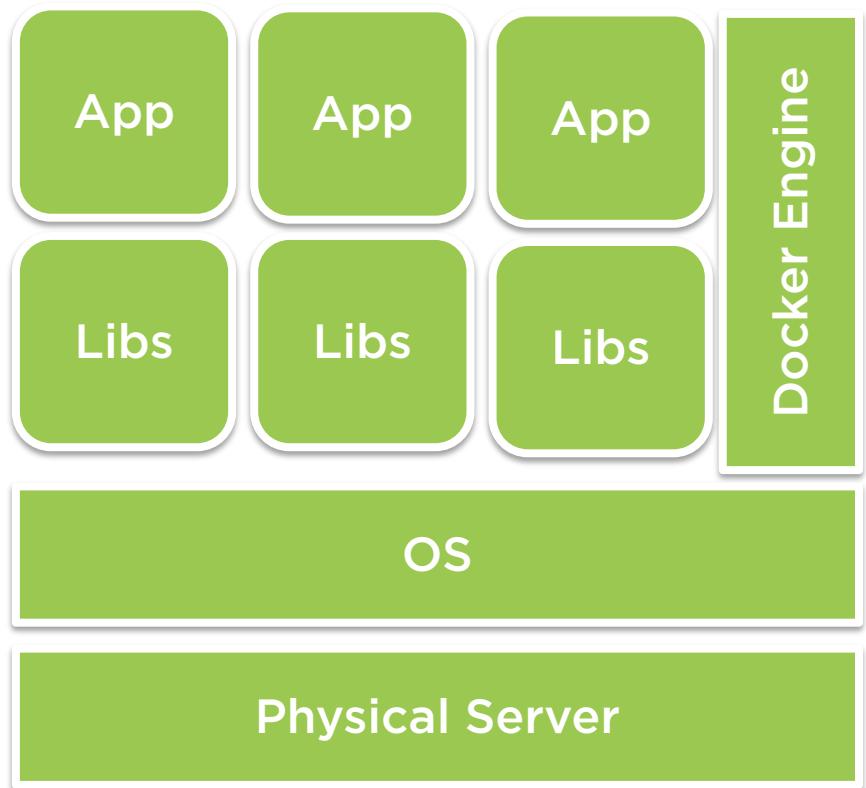


Containers allow the packaging of an application and its dependencies running in an instance

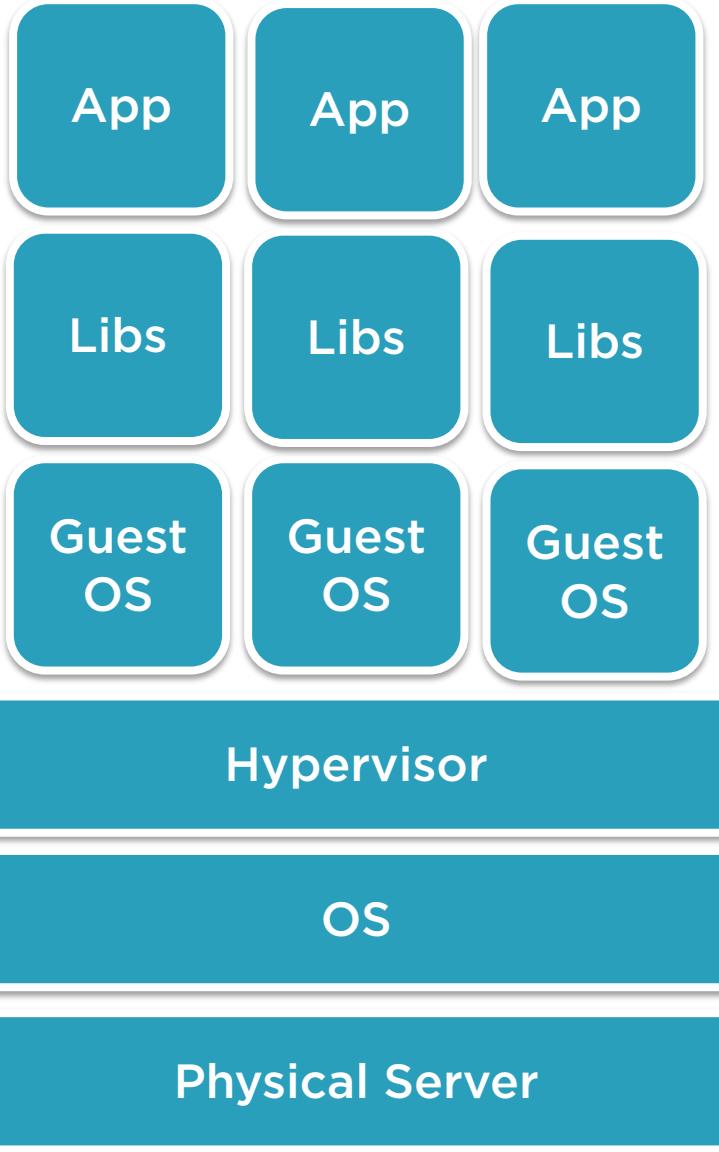
Containers allow software engineers to develop their applications and replicate across environments such as dev, stage, and prod in a consistent manner

Containers can move through Continuous Integration and Continuous Deployment pipelines in a manner that keeps the OS, dependencies and application unchanged providing ultimate flexibility and agility





VS



Containers

Virtual Machines



The diagram illustrates the structure of the command \$docker login localhost. A red bracket labeled "Command" covers the entire line. Another red bracket labeled "Docker" covers the prefix "\$docker". A third red bracket labeled "Options" covers the suffix "localhost".

```
$docker login localhost
```

Command

Docker

Options



A diagram illustrating the structure of a Docker command. The command is shown as `$docker run hello_world`. Red curly braces are used to group parts of the command. One brace groups the word `docker`, labeled **Docker**. Another brace groups the entire command line, labeled **command**. A third brace groups the argument `hello_world`, labeled **container**.

```
$docker run hello_world
```

command

Docker

container



What is Kubernetes?



Kubernetes (aka k8s) is an open source orchestration platform for containers

**K8s automates deploying,
managing, and scaling
containerized workloads**

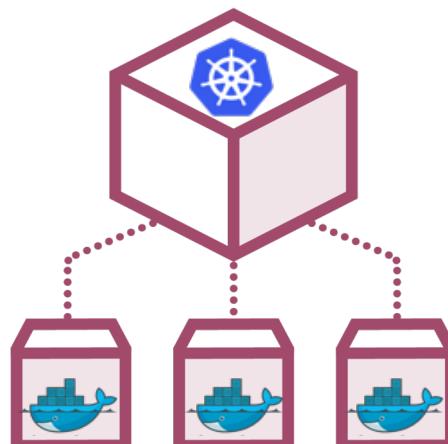


Understanding Kubernetes

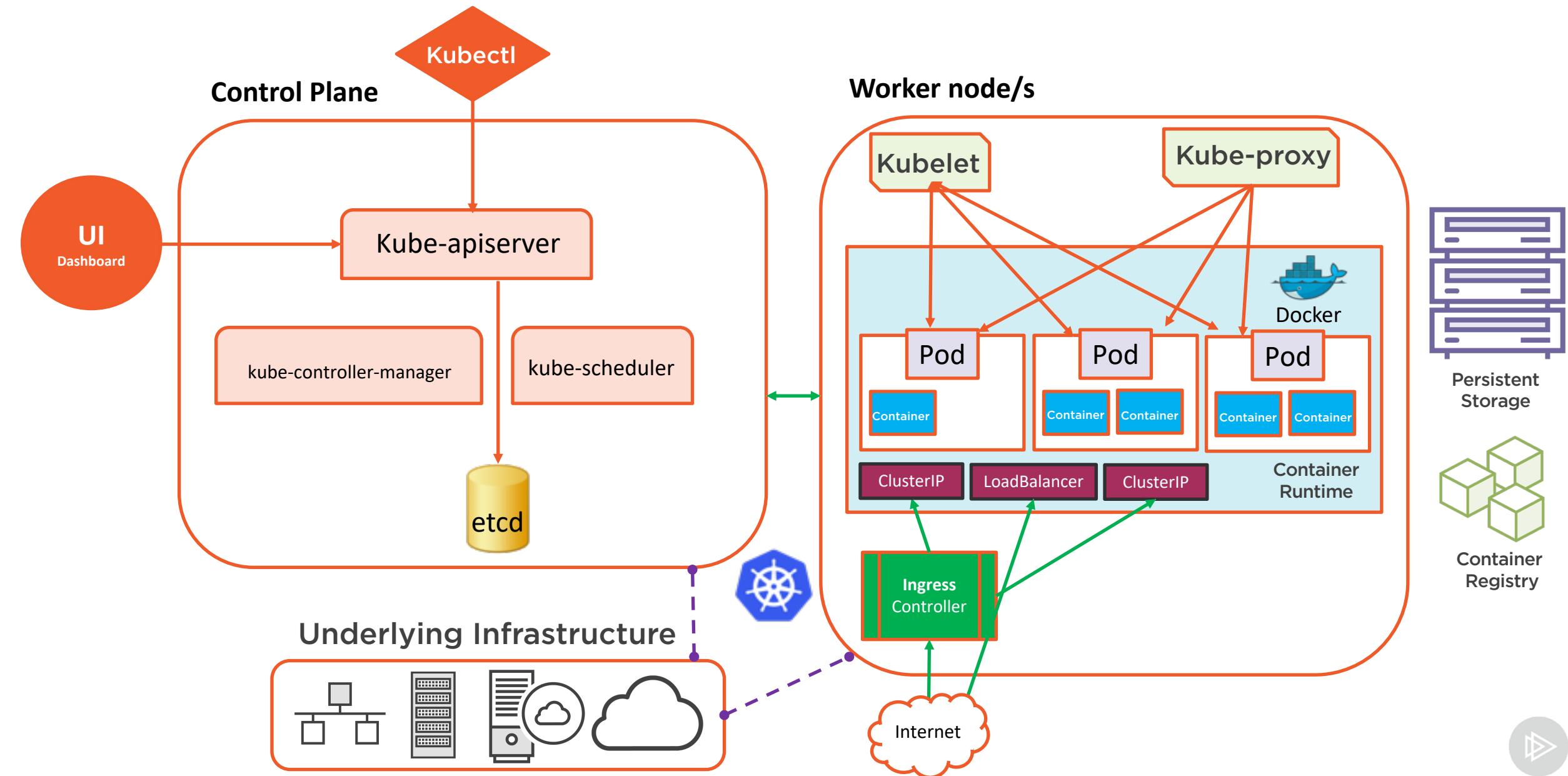
Kubernetes is an orchestration solution abstracts away the complexity of a multi-container environment

It combines compute, networking, & storage components that hundreds or thousands of containers rely on

It offers a declarative management model, in which you describe the target desired configuration



Kubernetes Architecture



Access to Kubernetes

The screenshot displays the Microsoft Azure portal interface for managing a Kubernetes cluster named "smiling-ostrich-k8s".

Microsoft Azure Navigation Bar: Shows the current dashboard and the selected "smiling-ostrich-k8s" cluster.

Kubernetes Overview: A top-level dashboard showing CPU and Memory usage over time. The CPU usage chart shows values between 0.0003 and 0.001 cores, while the Memory usage chart shows values between 356 MiB and 1.57 GiB.

Services and Ingresses: A detailed view of the cluster's services. The table lists the following services:

Name	Namespace	Status	Type
kubernetes	default	Ok	ClusterIP
healthmodel-replicaset-service	kube-system	Ok	ClusterIP
kube-dns	kube-system	Ok	ClusterIP
metrics-server	kube-system	Ok	ClusterIP
prometheus-operator-kube-p-coredns	kube-system	Ok	ClusterIP
prometheus-operator-kube-p-kube-controller-manager	kube-system	Ok	ClusterIP
prometheus-operator-kube-p-kube-etcd	kube-system	Ok	ClusterIP
prometheus-operator-kube-p-kube-proxy	kube-system	Ok	ClusterIP
prometheus-operator-kube-p-kube-scheduler	kube-system	Ok	ClusterIP
prometheus-operator-grafana	monitoring	Ok	LoadBalancer
prometheus-operator-kube-p-alertmanager	monitoring	Ok	LoadBalancer
prometheus-operator-kube-p-operator	monitoring	Ok	ClusterIP
prometheus-operator-kube-p-prometheus	monitoring	Ok	LoadBalancer
prometheus-operator-kube-state-metrics	monitoring	Ok	ClusterIP
prometheus-operator-prometheus-node-exporter	monitoring	Ok	ClusterIP

Bash Terminal: A terminal window showing the output of two `kubectl get services` commands. The first command shows services in the default namespace, and the second shows services in the monitoring namespace.

```
@Azure:~$ kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes     ClusterIP  10.0.0.1    <none>        443/TCP   20d

@Azure:~$ kubectl get services -n monitoring
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
alertmanager-operated   ClusterIP  None         None
prometheus-operated   ClusterIP  None         None
prometheus-operator-grafana   LoadBalancer  10.0.118.192
prometheus-operator-kube-p-alertmanager   LoadBalancer  10.0.13.94
prometheus-operator-kube-p-operator   ClusterIP  10.0.4.192
prometheus-operator-kube-p-prometheus   LoadBalancer  10.0.7.203
prometheus-operator-kube-state-metrics   ClusterIP  10.0.246.182
prometheus-operator-prometheus-node-exporter   ClusterIP  10.0.232.184
```

Understanding Helm and Kustomize





What Is Helm?

Think of Helm as a package manager for Kubernetes

It is a Kubernetes deployment tool for automating creation, packaging, configuration, & deployment of apps & configurations to Kubernetes clusters

Official CNCF project



What Is Kustomize?



Kustomize is a standalone tool to customize the creation of Kubernetes objects through a file called `kustomization.yaml`

Kustomize is a template-free way to customize application configuration that is built into kubectl

It traverses a Kubernetes manifest to add, remove or update configuration options without forking or actual YAML files



Kustomize Structure



Kustomization.yaml

```
1 bases:  
2 - ldap  
3 patches  
4 - patch.yaml  
5 ...
```

Base - Idap

```
1 apiVersion: v1beta2  
2 kind: Deployment  
3 metadata:  
4 name: ldap  
5 labels:  
6 app: ldap  
7 spec:  
8 replicas: 1  
9 selector:  
10 matchLabels:  
11 app: ldap  
12 template:  
13 metadata:  
14 labels:  
15 app: ldap  
spec:  
16 containers:  
17 - image: osixia/openldap  
18 name: ldap  
19 volumeMounts:  
20 - name: ldap-data  
21 mountPath: /var/l  
22 - name: ldap-config  
23 mountPath: /etc/l  
24 - name: ldap-certs  
25 mountPath: /conta  
26 - name: configmap-v  
27 mountPath: /conta  
28 - name: container-r  
29 mountPath: /conta  
30 ports:  
31 - containerPort: 389  
32 - name: openldap  
33 volumes:  
34 - name: ldap-data
```

patch.yaml - Staging

```
1 # Staging Deployment  
2  
3 apiVersion: apps/v1beta2  
4 kind: Deployment  
5 metadata:  
6 name: ldap  
7 spec:  
8 replicas: 2
```

patch.yaml - Prod

```
1 # Production Deployment  
2  
3 apiVersion: apps/v1beta2  
4 kind: Deployment  
5 metadata:  
6 name: ldap  
7 spec:  
8 replicas: 6  
9 template:  
10 spec:  
11 volumes:  
12 - name: ldap-data  
13 emptyDir: null
```



Kustomize Structure

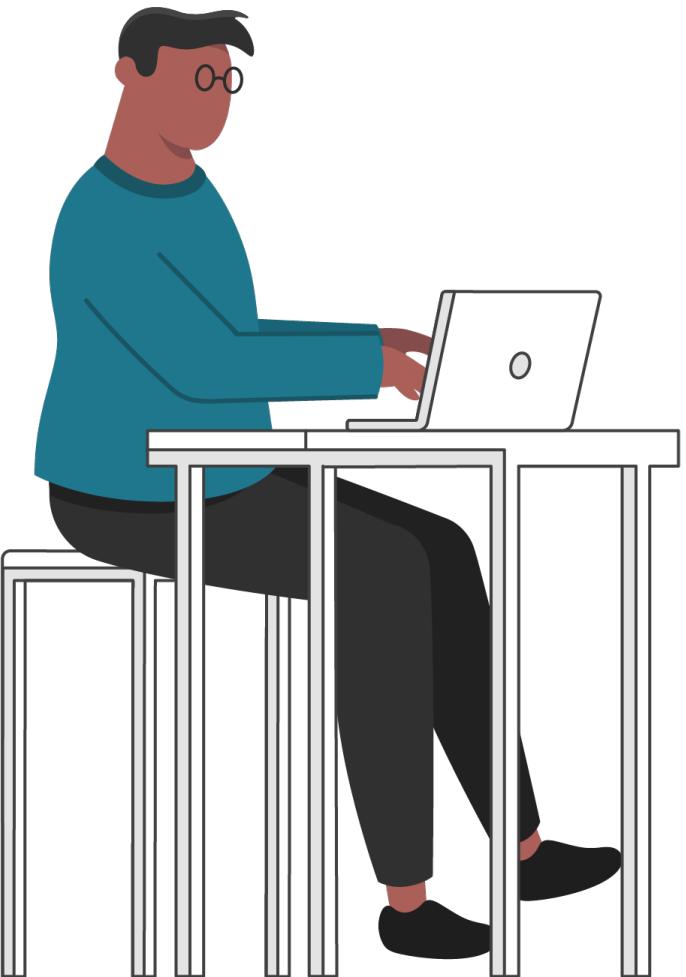
```
hello-world/
  └── base
      ├── deployment.yaml
      └── kustomization.yaml
  └── overlays
      ├── production
          ├── replica_count.yaml
          └── kustomization.yaml
      └── staging
          ├── replica_count.yaml
          └── kustomization.yaml
```



Understanding GitOps Core Concepts



What Is Gitops?



“GitOps is an operating model pattern for cloud native applications & Kubernetes storing application & declarative infrastructure code in Git as the source of truth used for automated continuous delivery.”



GitOps History

2017 the term GitOps was introduced by **Alexis Richardson** of Weaveworks in the '*Operations by Pull Request*' blog



2018 “GitOps is the best thing since configuration as code”
Tweet by Kelsey Hightower



2018 GitOps adopted by major cloud providers



December 10th, 2020 the GitOps Working Group hosted its first meeting as a step towards governance, documentation, guidelines, & a clear definition for GitOps



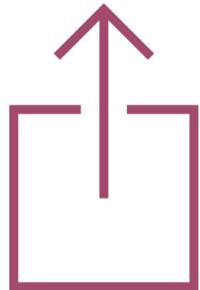
2021 1st GitOpsCon launched



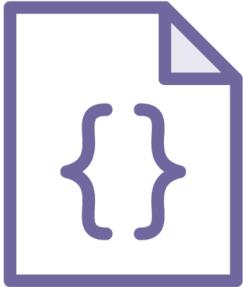
GitOps Principles



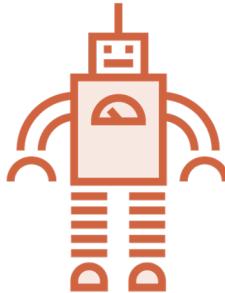
Git is the source of truth
for entire system



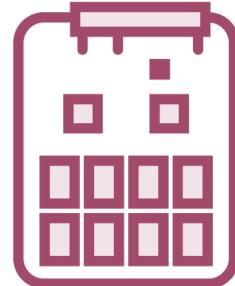
Git as the single place
for operations (create,
change, delete)



Desired system state is
versioned in Git



Autonomous Agents
enforce desired state
and alert on drift



System state described
declaratively



Automated delivery of
Approved system state
changes



GitOps Architecture Components

Source Control System

(i.e. ADO, GitHub, GitLab, Bit Bucket)

Git Repository

Container/Helm Registry

Operator

(i.e. Flux, ArgoCD, Kubectl apply, Terraform K8s provider etc)

Runtime Environment

(i.e. 1 K8s cluster multiple namespaces or 1 K8s cluster per environment i.e. dev, stage, prod)

Namespaces

(namespace per environment, per app, service, per engineer, per build ect)



Use Cases for GitOps



Cloud Native App Management i.e. “CD” in CI/CD



Service Rollouts



Infrastructure management i.e. K8s clusters, fleets, microservices



GitOps Operators

Flux

- Kubernetes GitOps Operator
- fluxcd.io

Argo CD

- Kubernetes GitOps Operator with visual approach
- argoproj.github.io/argo-cd

Kubestack

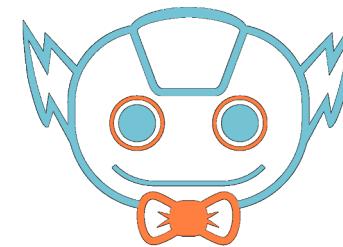
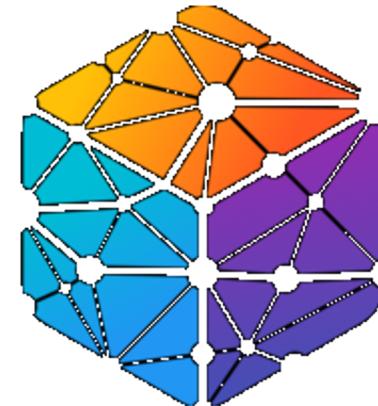
- Terraform GitOps Framework for building Kubernetes on any platform
- www.kubestack.com

Jenkins X

- Full Kubernetes CI/CD with built-in GitOps
- jenkins-x.io

Atlantis

- GitOps for cloud via Terraform Pull Request Automation
- www.runatlantis.io



History and Overview of Argo CD



The Argo Project

Group of four core solutions for orchestrating container workflows



**Workflow engine for
orchestrating parallel jobs on
Kubernetes**



**Declarative, continuous
delivery GitOps Operator for
Kubernetes**



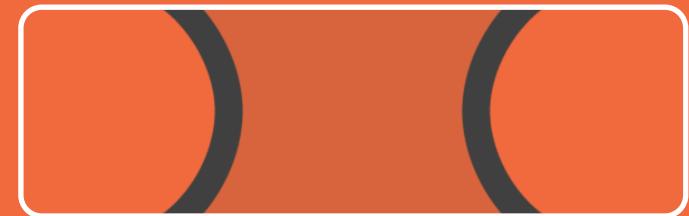
**Advanced deployment
capabilities such as blue-green,
canary, & canary analysis**



**Event-driven workflow
automation framework for
Kubernetes**



Argo CD



Argo CD is the core component of the Argo Project

Argo CD is a GitOps Operator that provides continuous delivery for Kubernetes

Has Application Controller to continuously monitor apps running on K8s cluster/s comparing the live app state against desired state defined in a Git repository



History of Argo CD

2016
Applatix a new startup is launched

August 31st, 2017
Argo Workflows launched

January 22nd 2018
Intuit acquired Applatix

- The former Applatix team was tasked with building a CD product based on cloud native & GitOps best practices & principles

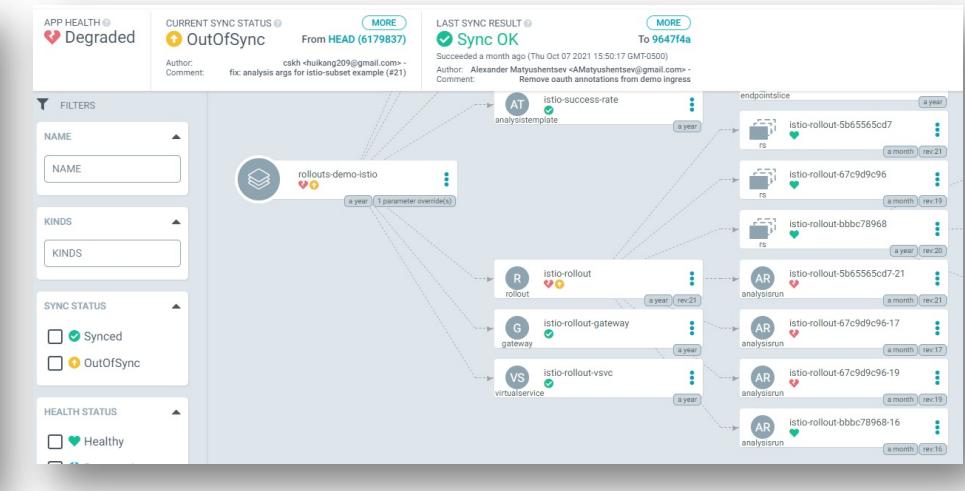
August 30th, 2018 Argo CD officially announced



Top Argo CD Features

The screenshot shows the Argo CD web interface's main dashboard. It features a sidebar on the left with sections for Applications, Labels, Projects, and Clusters. The main area displays several application cards, each with details like Project, Labels, Status, Repository, Target Rev., Path, Destination, and Namespace. Buttons for SYNC, REFRESH, and DELETE are at the bottom of each card.

This screenshot shows the 'CREATE' dialog for a new application. It includes fields for Application Name, Project, SYNC POLICY (set to Manual), SYNC OPTIONS (with checkboxes for SKIP SCHEMA VALIDATION, PRUNE LAST, PRUNE PROPAGATION POLICY: foreground, REPLACE, and RETRY), and SOURCE (Repository URL). A 'FILTERS' sidebar on the left is visible.



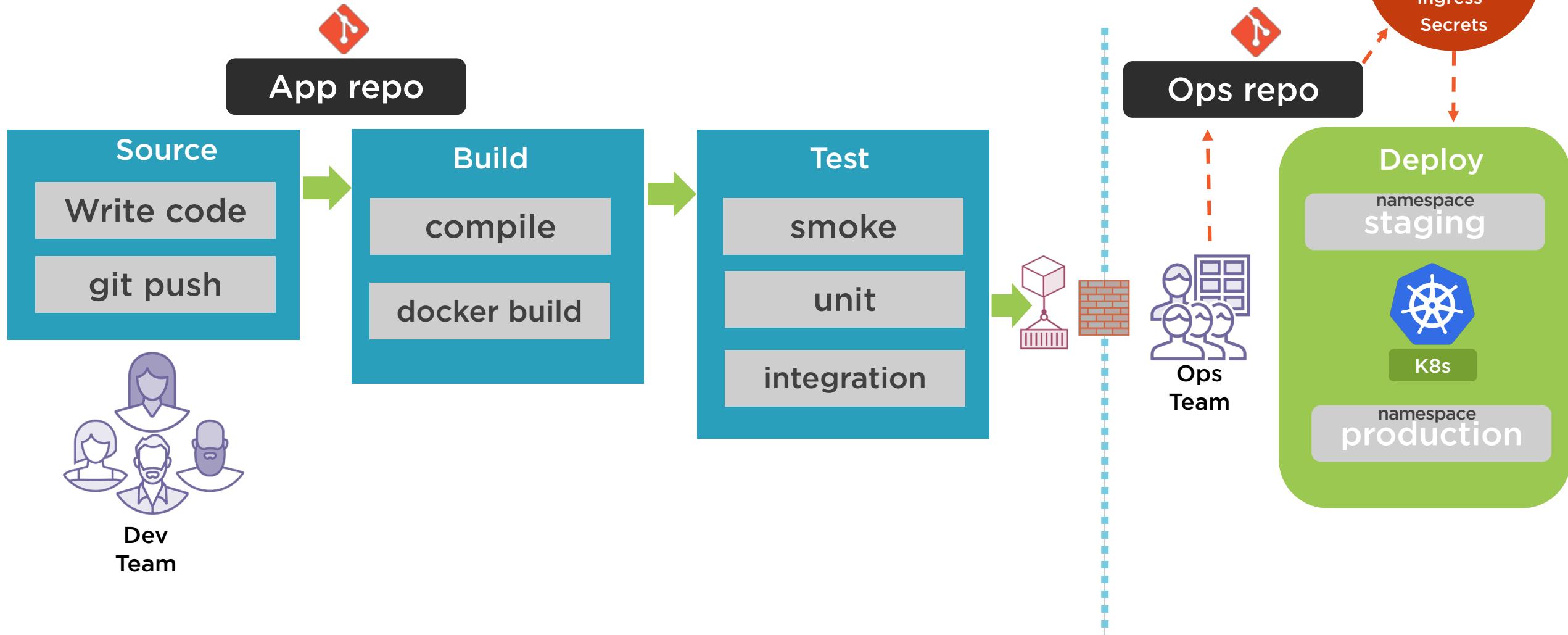
A Web User Interface

Automated Deployment
of Apps

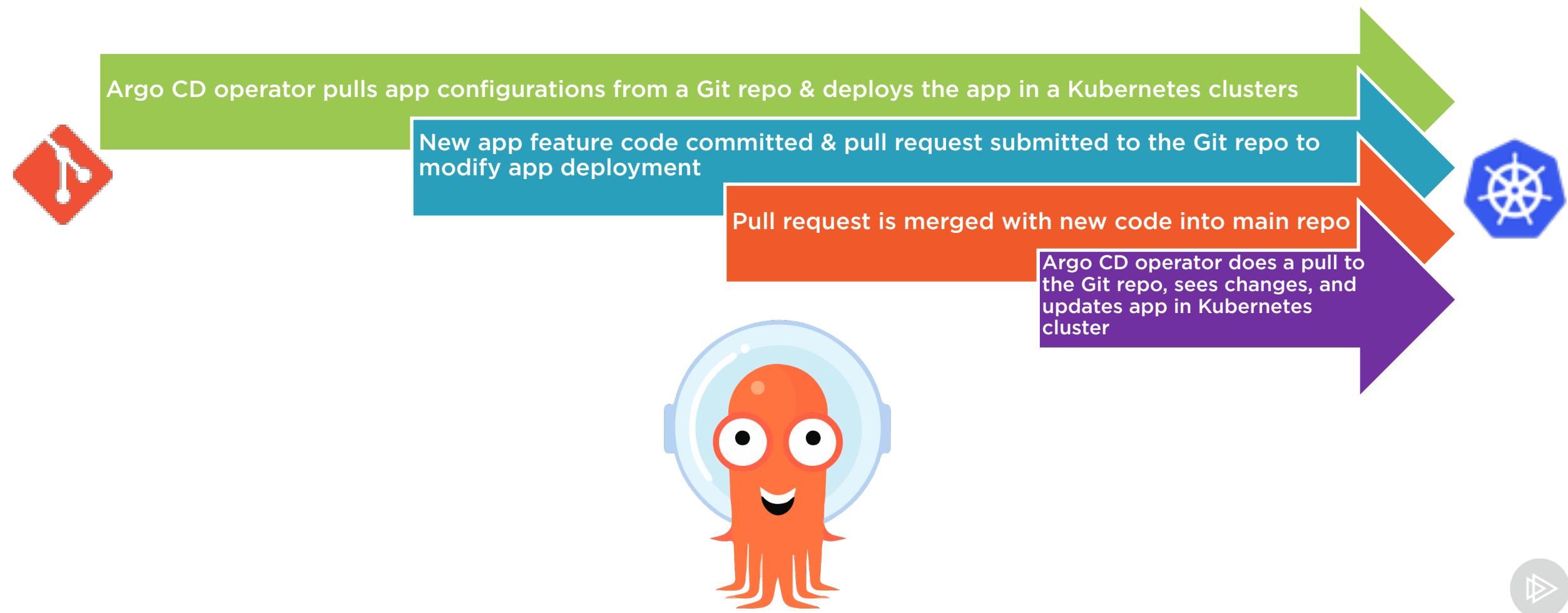
Health status of app &
its resources



Standard CI/CD Workflow



CD Workflow with Argo CD



Understanding Argo CD Core Concepts and Architecture



API Server

a gRPC/REST server that exposes the API that is consumed by the Web UI, CLI, and CI/CD systems

Repository Server

an internal service that maintains a local cache of the Git repo holding the app manifests

Application Controller

a Kubernetes controller that continuously monitors running apps & compares the current, live state against the desired state in the Git repo



Argo CD Architecture

API Server

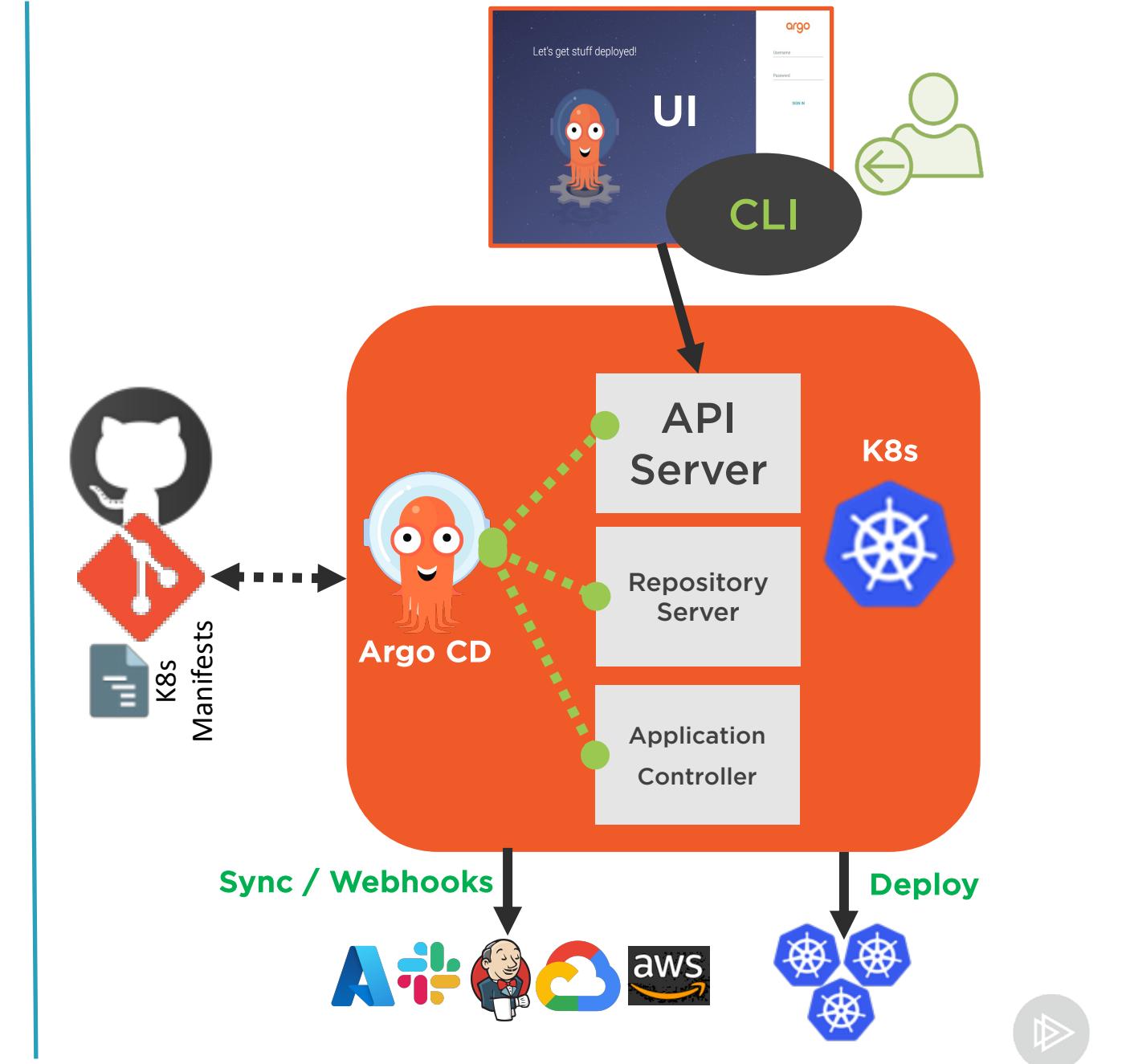
a gRPC/REST server that exposes the API that is consumed by the Web UI, CLI, and CI/CD systems

Repository Server

an internal service that maintains a local cache of the Git repo holding the app manifests

Application Controller

a Kubernetes controller that continuously monitors running apps & compares the current, live state against the desired state in the Git repo



Argo CD Components

Components (Deployments / Services):

argocd-dex-server

- Argo CD embeds and bundles Dex as part of its installation, for the purpose of delegating authentication to an external identity provider and handling SSO

argocd-metrics / argocd-server-metrics

- Argo CD exposes Application metrics and API Server metrics for Prometheus

argocd-redis

- Redis used as caching technology working with the Argo CD repository server

argocd-repo-server

- Clones the Git repository, keeping it up to date and generating manifests using the appropriate tool

argocd-server

- Runs the ArgoCD API server



Understanding Supported Tooling with Argo CD



Argo CD Supported Integrated Tooling

Argo CD supports several different ways in which Kubernetes manifests can be defined:

Kustomize

Helm

Ksonnet

Jenkins

Jsonnet



Argo CD Tool Detection

When a new app is created in Argo CD it is able to detect the tooling used to create that app

The tool is detected as follows:



Jsonnet/Ksonnet if there are two files, one named `app.yaml` and one named `components/params.libsonnet` or any file matching `*.jsonnet` in a directory `app`



Helm if there's a file matching `Chart.yaml`



Kustomize if there's a `kustomization.yaml`, `kustomization.yml`, or `Kustomization`



Summary



In this module we covered:

- We gained an understanding of Containers, Kubernetes, as well as Helm & Kustomize
- Brief understanding of GitOps including its history, core concepts, and even GitOps Operators
- Overview of the Argo Project & its core solutions with a focus on Argo CD
- We then explored Argo CD, what it is, what it does including top features, its history, architecture & its supported tooling

Why this is important:?

- Its useful to understand core things like Containers, kubernetes, GitOps, Helm, Kustomize & topics we covered as these are used with Argo CD
- Its good to have knowledge of Argo & Argo CD all up to build on as you continue with this course

