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

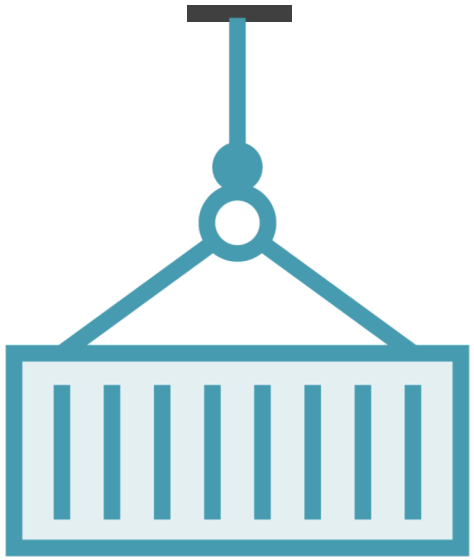
History and Overview of Argo CD

Understanding Argo CD core concepts and architecture

Understanding Supported Tooling with Argo CD

Understanding Containers and Kubernetes

Understanding Containers

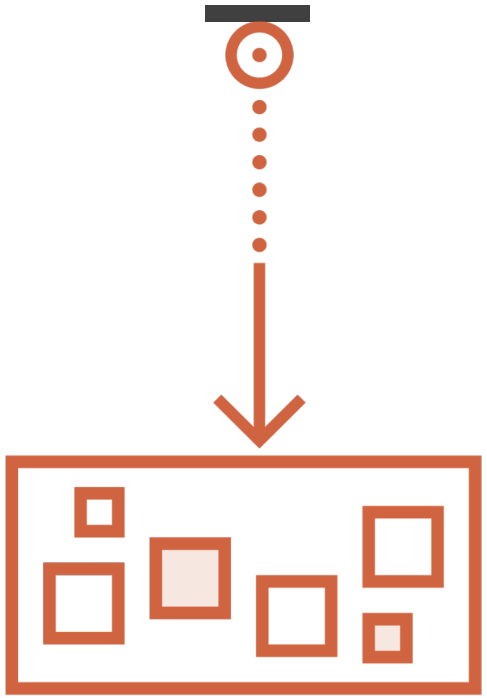


Containers are an abstraction of the application layer in an isolated user-space instance

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

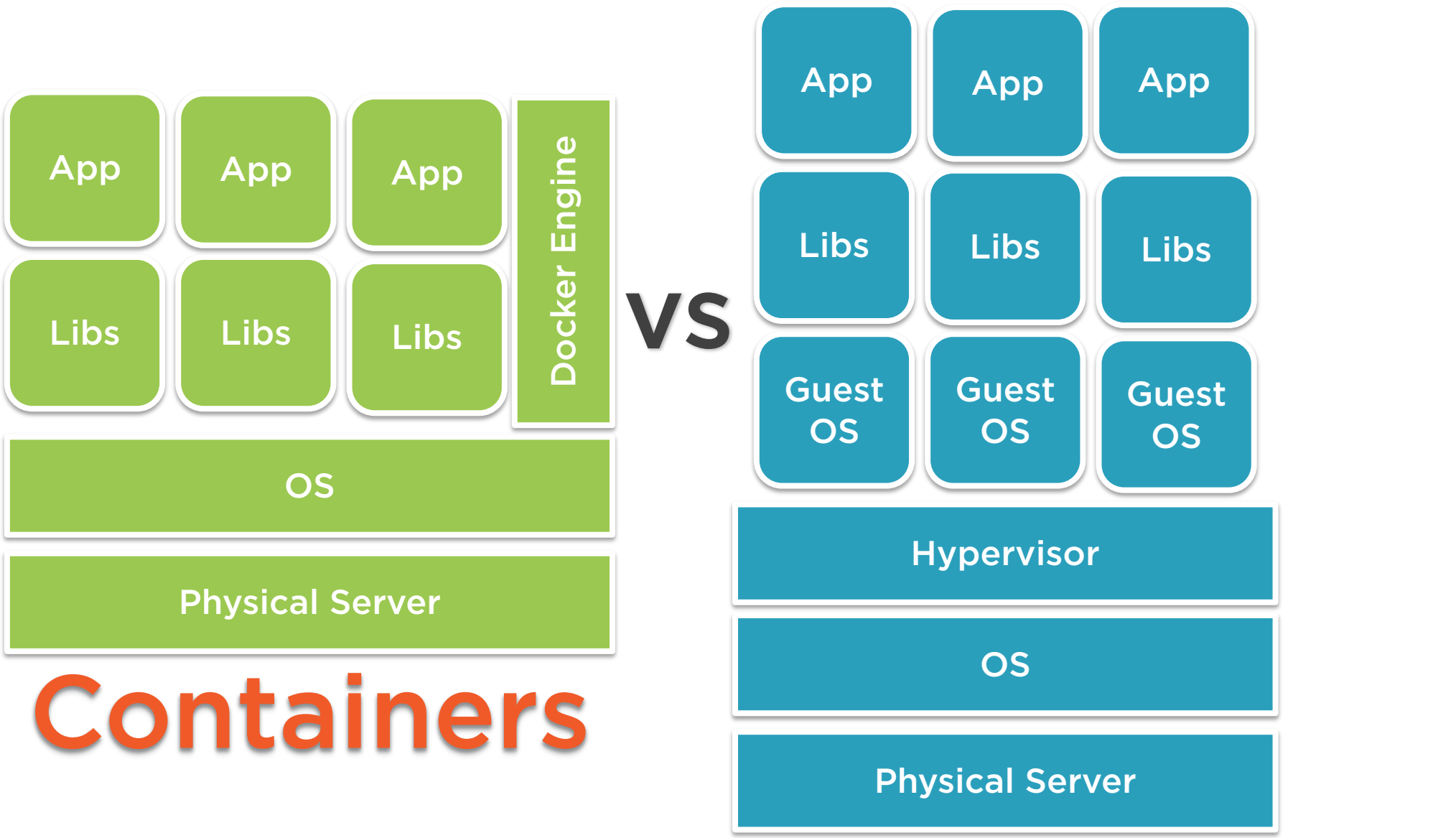
Understanding Containers

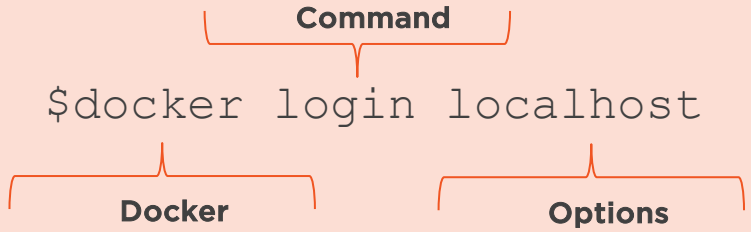


Containers allow the packaging of an application and its dependencies running in an instance

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


App
App
App
Libs
Libs
Libs
Docker Engine
OS
Physical Server
Containers
VS
App
App
App
Libs
Libs
Libs
Guest OS
Guest OS
Guest OS
Hypervisor
OS
Physical Server
Virtual Machines


Command
$docker login localhost
Docker
Options


command
$docker run hello_world
Docker
container

What is Kubernetes?

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

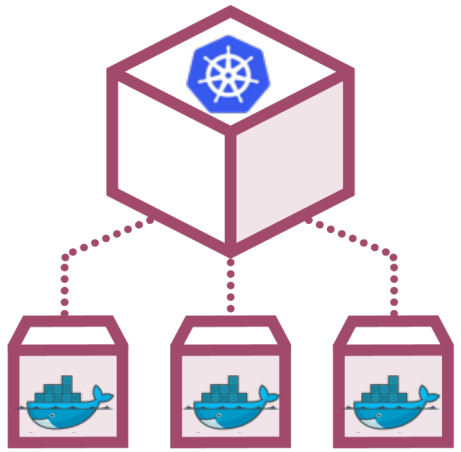
K8s automates deploying, managing, and scaling containerized workloads

Understanding Kubernetes

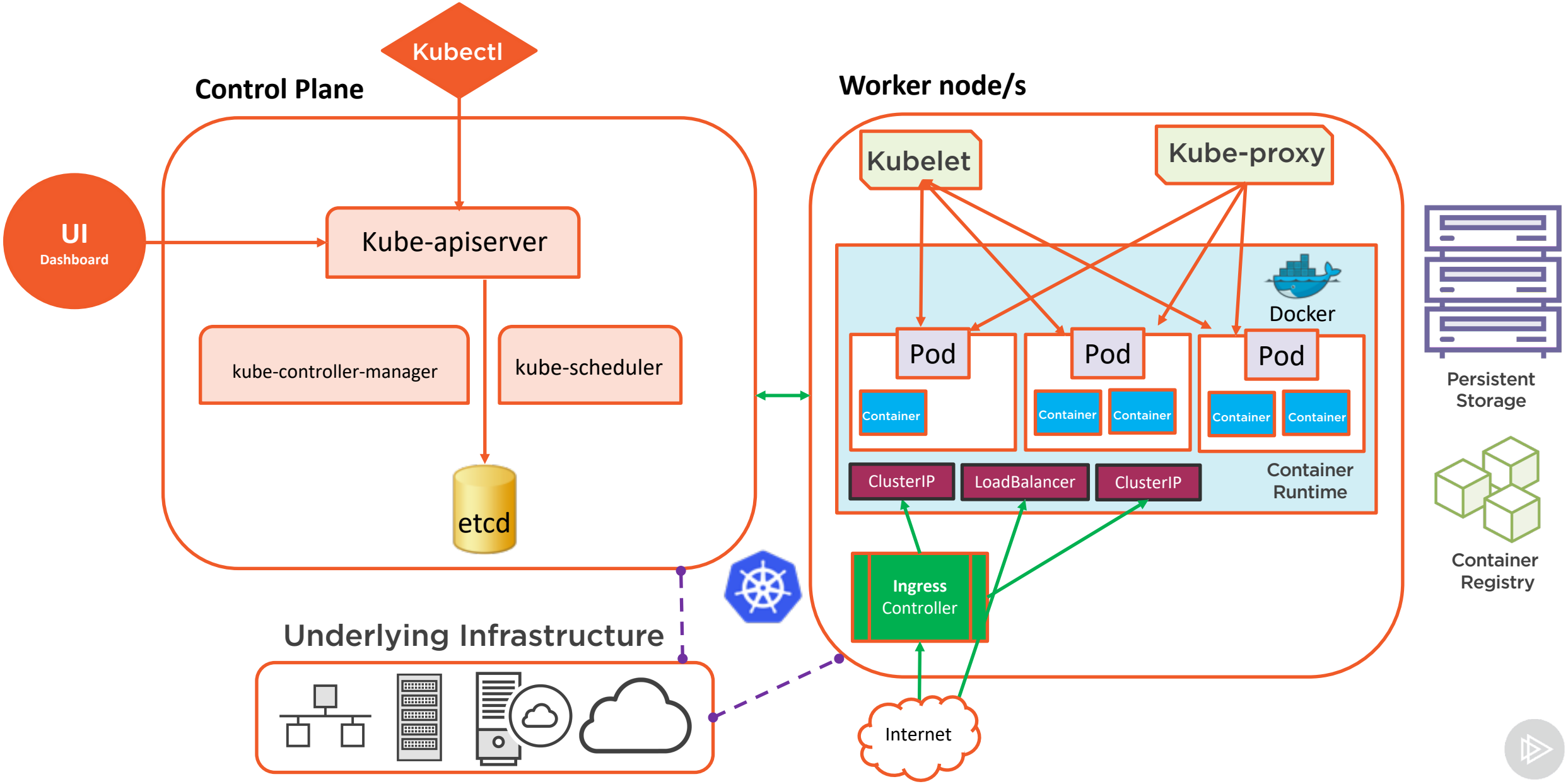
Kubernetes is an orchestration solution abstracts away the complexity of a multi-container environment

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

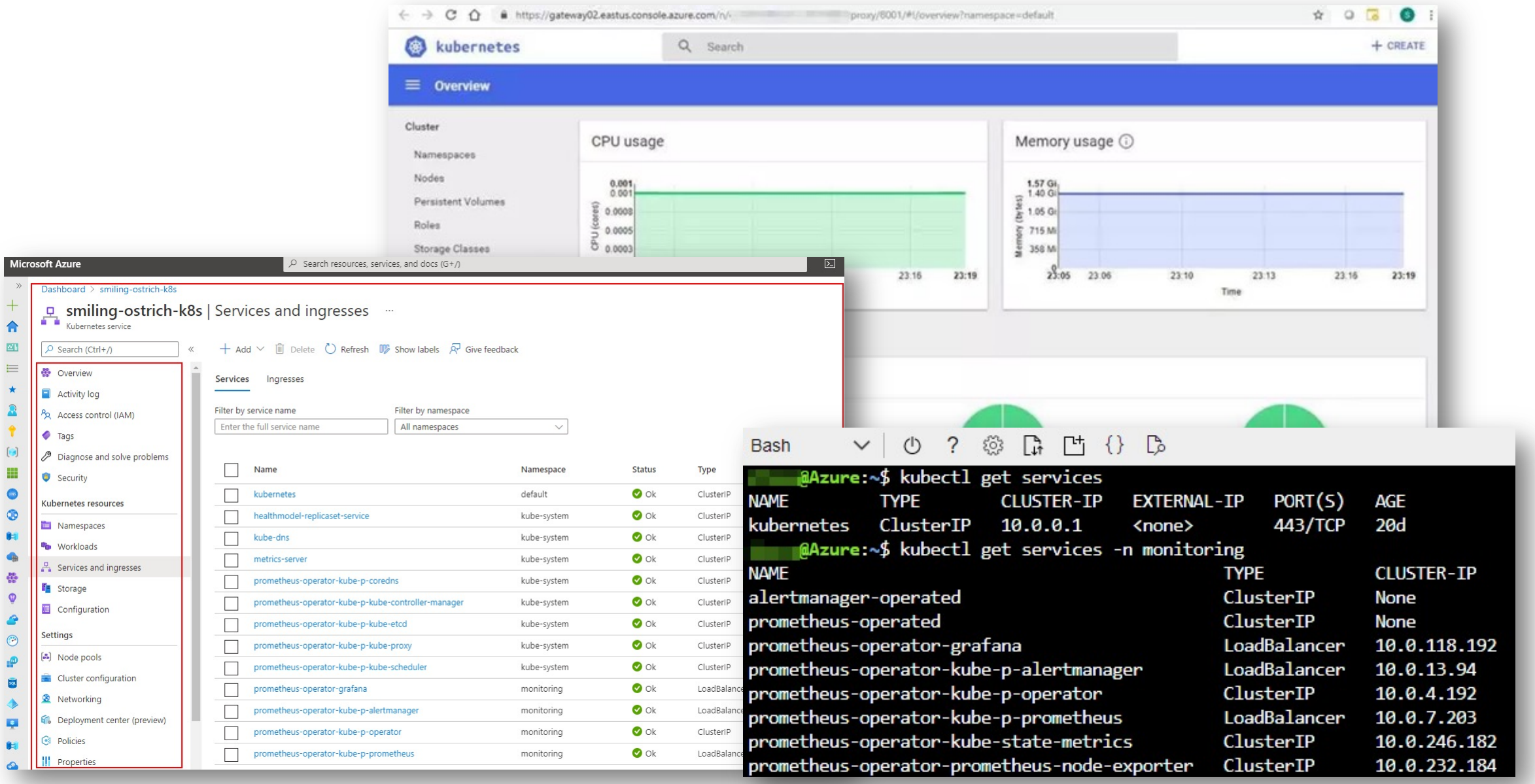
It offers a declarative management model, in which you describe the target desired configuration



Kubernetes Architecture



Access to Kubernetes



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

Base - ldap

```yaml
apiVersion: v1beta2
kind: Deployment
metadata:
  name: ldap
  labels:
    app: ldap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ldap
  template:
    metadata:
      labels:
        app: ldap
    spec:
      containers:
      - image: osixia/openlda
        name: ldap
        volumeMounts:
          - name: ldap-data
            mountPath: /var/l
          - name: ldap-config
            mountPath: /etc/l
          - name: ldap-certs
            mountPath: /conta
          - name: configmap-v
            mountPath: /conta
          - name: container-r
            mountPath: /conta
        ports:
          - containerPOrt: 389
          - name: openldap
      volumes:
      - name: ldap-data
```

Kustomization.yaml

```yaml
bases:
  - ldap
patches
  - patch.yaml
…
```

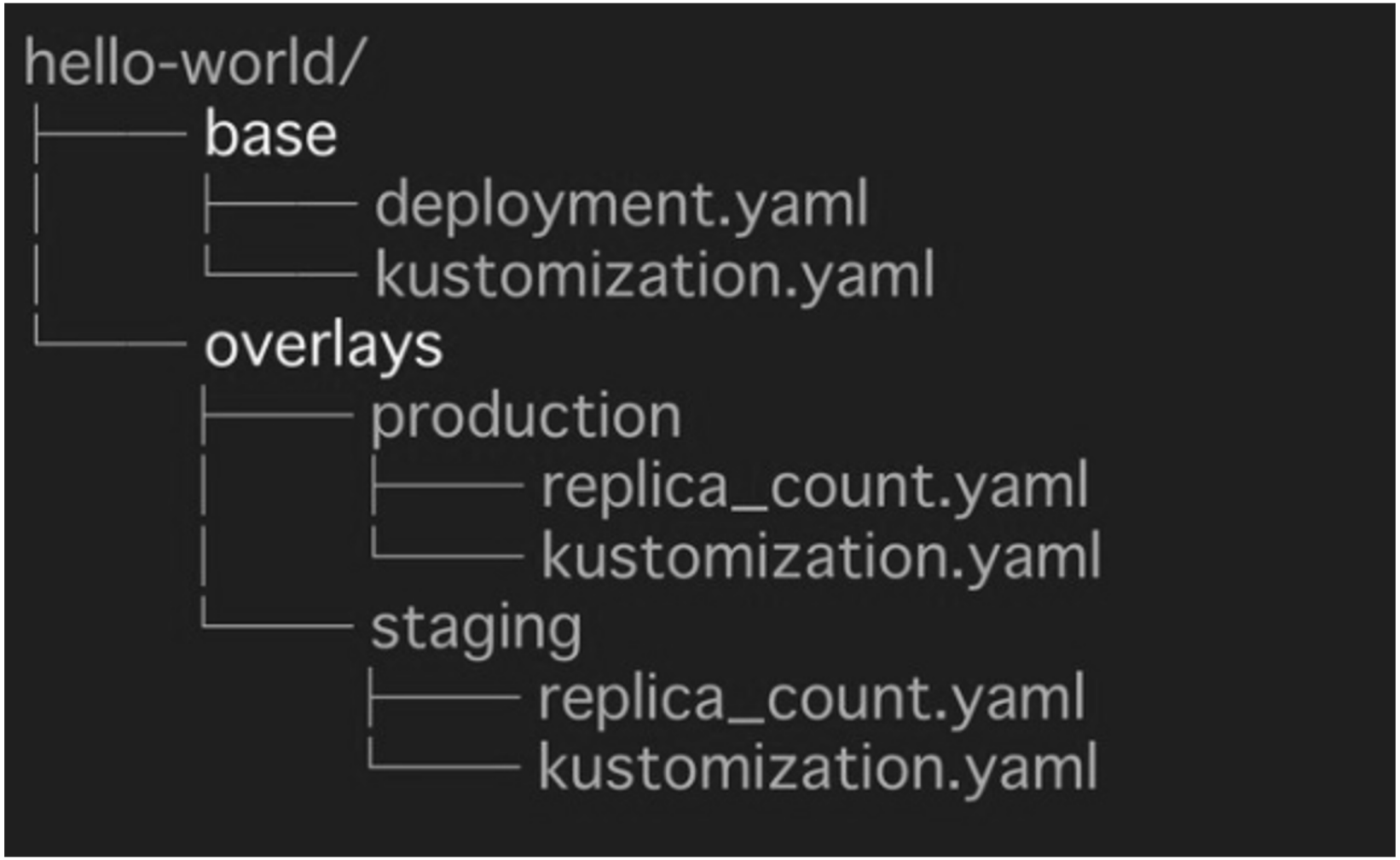
patch.yaml - Staging

```yaml
# Staging Deployment

apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: ldap
spec:
  replicas: 2
```

patch.yaml - Prod

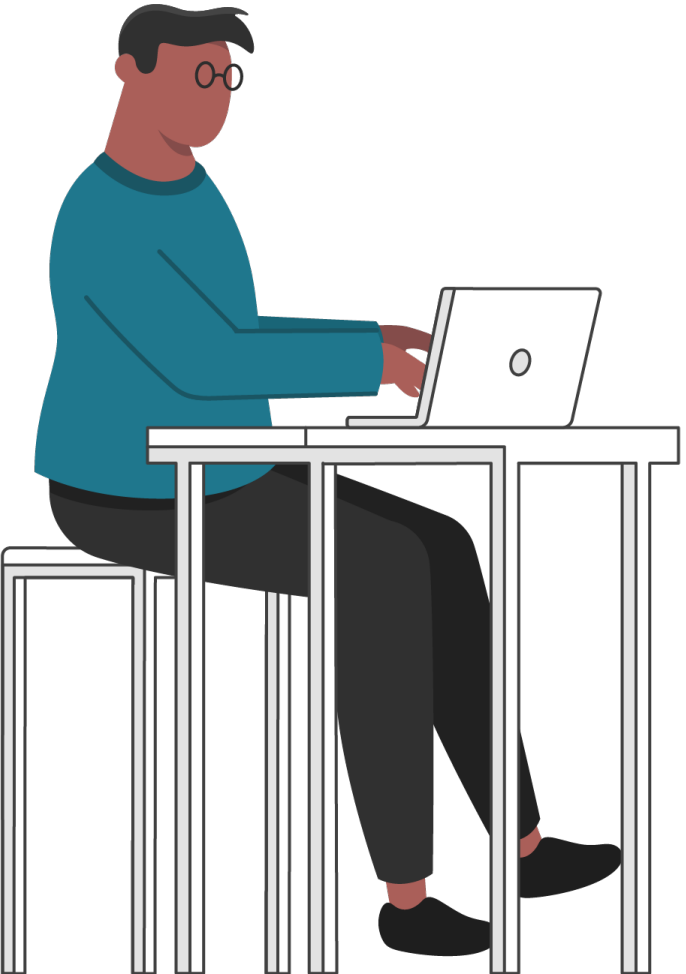
```yaml
# Production Deployment

apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: ldap
spec:
  replicas: 6
  template:
    spec:
      volumes:
        - name: ldap-data
          emptyDir: null
```

Kustomize Structure



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

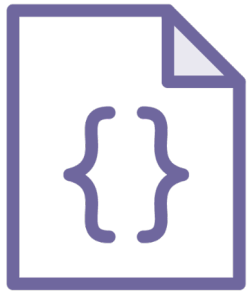
2018 GitOps adopted by major cloud providers

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

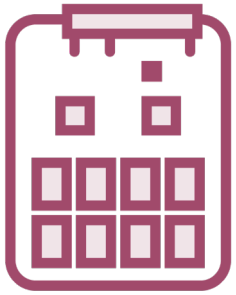
2021 1st GitOpsCon launched

GitOps Principles

Git is the source of truth for entire system

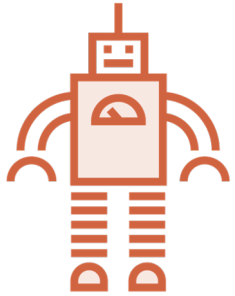


Desired system state is versioned in Git



System state described declaratively

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



Autonomous Agents enforce desired state and alert on drift

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

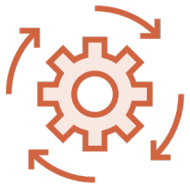
(i.e. 1 K8s cluster multiple namespaces or 1 K8s cluster per environment i.e. dev, stage, prod)

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

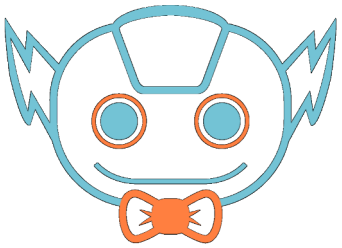
- Terraform GitOps Framework for building Kubernetes on any platform
- www.kubestack.com

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



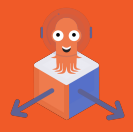
History and Overview of Argo CD

The Argo Project

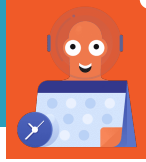
Group of four core solutions for orchestrating container workflows

Workflow engine for orchestrating parallel jobs on Kubernetes

Declarative, continuous delivery GitOps Operator for Kubernetes

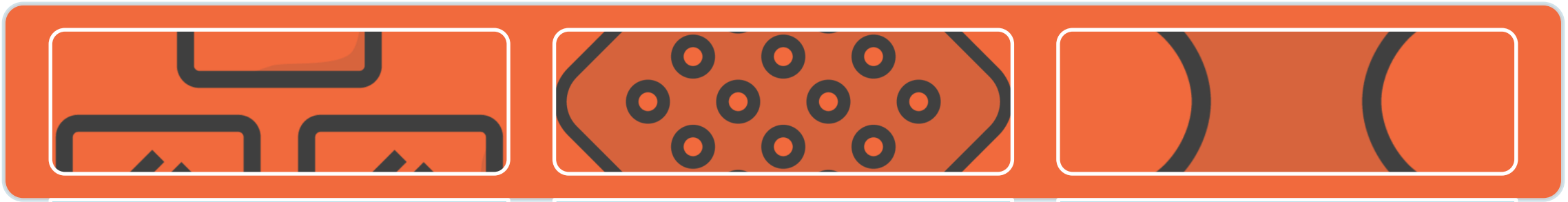


Advanced deployment capabilities such as blue-green, canary, & canary analysis



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

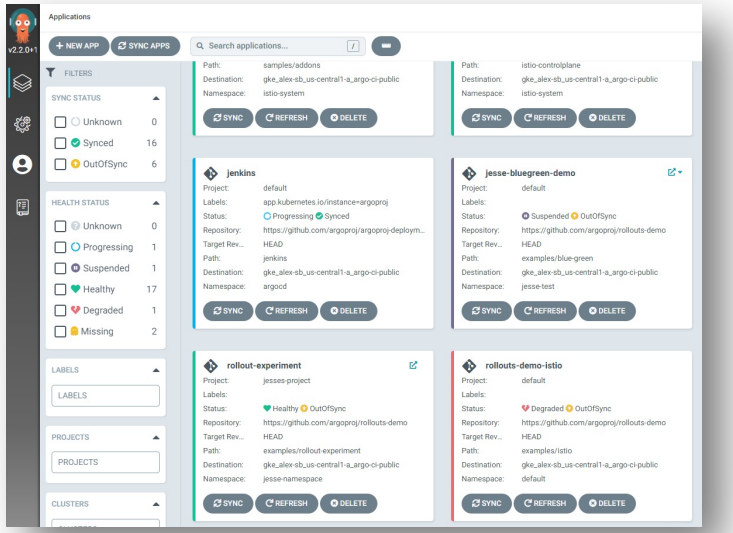
August 31st, 2017
Argo Workflows launched

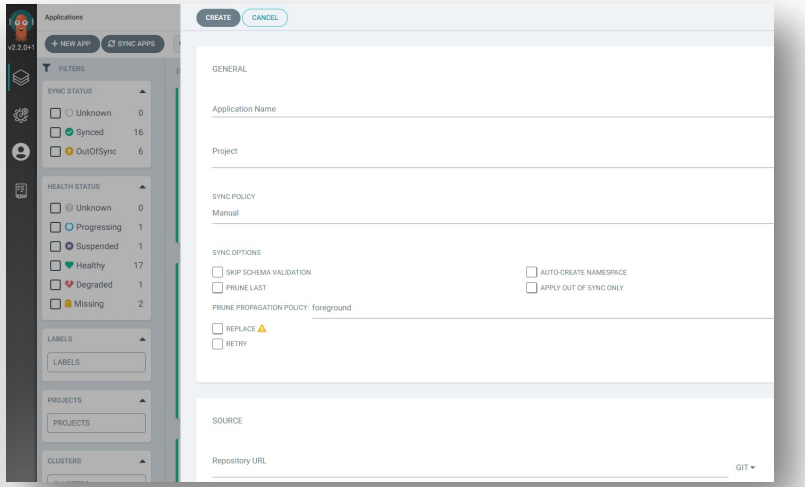
January 22nd 2018
Intuit acquired Applatix

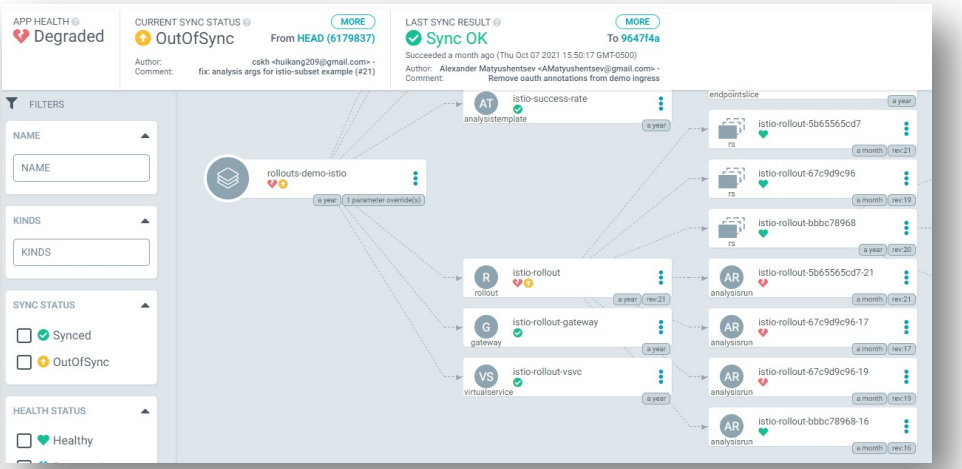
- The former Applatix team was tasked with building a CD product based on cloud native & GitOps best practices & principles

August 30th, 2018 Argo CD officially announced

Top Argo CD Features





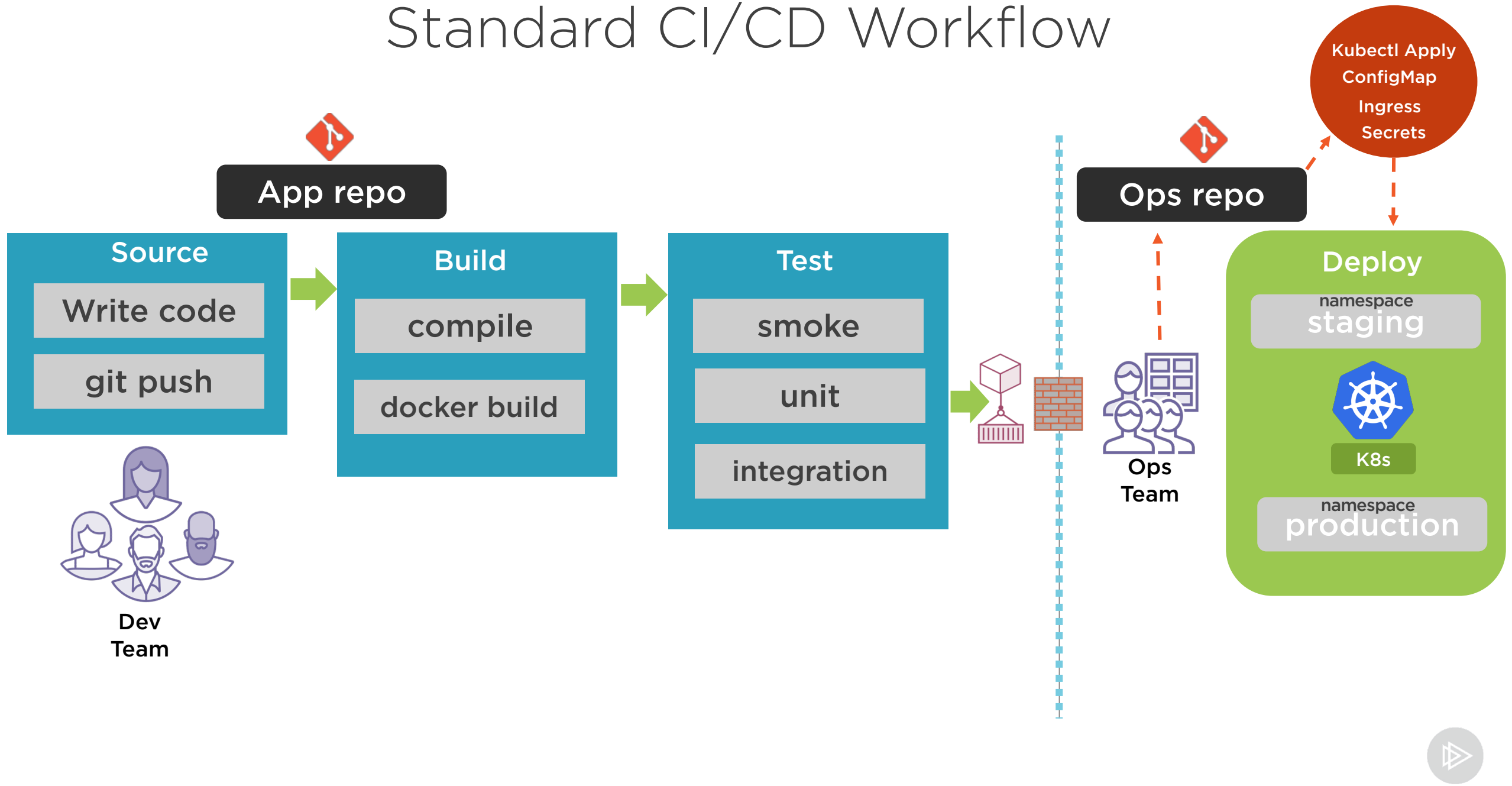


A Web User Interface

Automated Deployment of Apps

Health status of app & its resources

Standard CI/CD Workflow



CD Workflow with Argo CD

Argo CD operator pulls app configurations from a Git repo & deploys the app in a Kubernetes clusters

New app feature code committed & pull request submitted to the Git repo to modify app deployment

Pull request is merged with new code into main repo

Argo CD operator does a pull to the Git repo, sees changes, and updates app in Kubernetes cluster

Understanding Argo CD Core Concepts and Architecture

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 Architecture

API Server

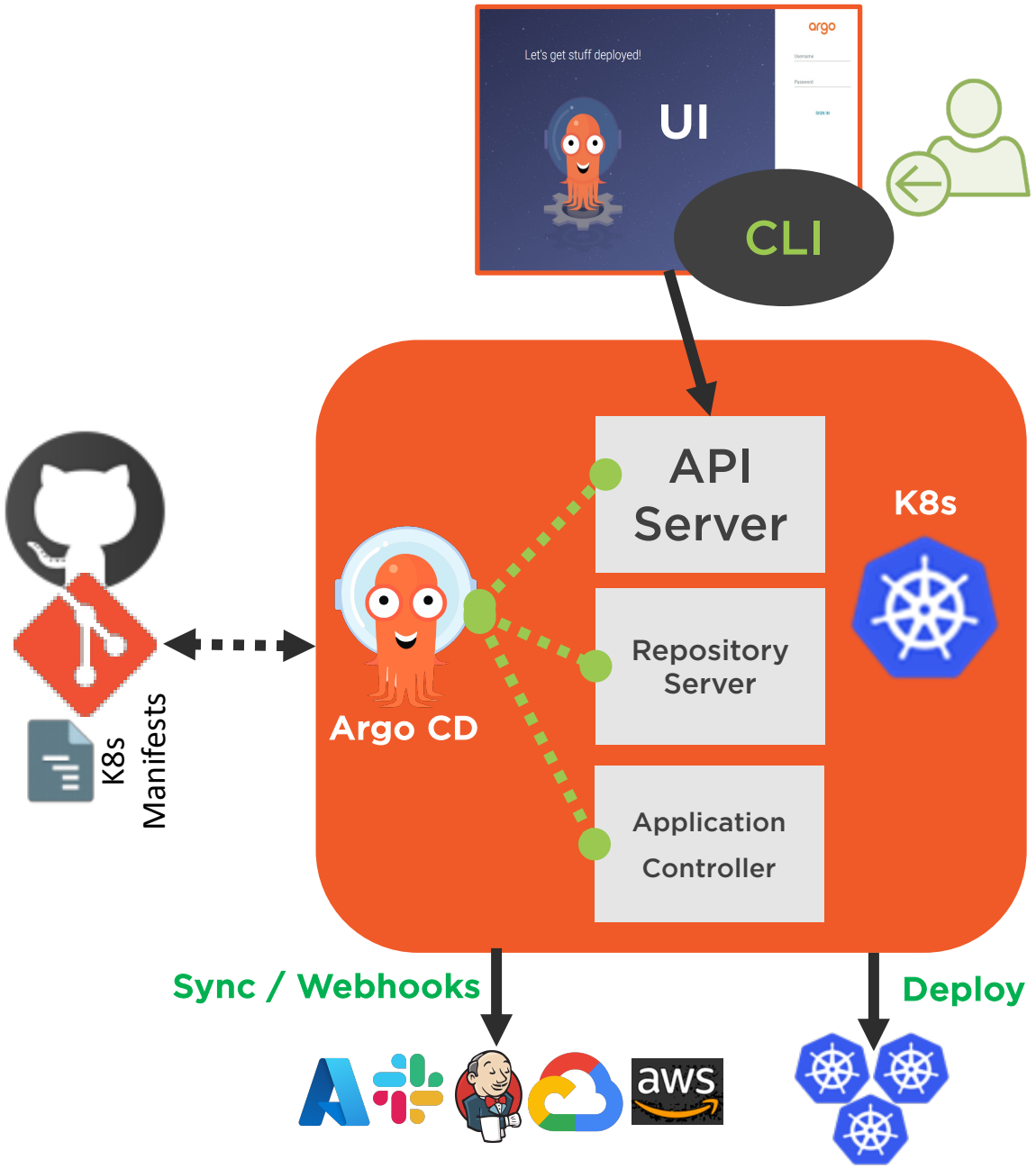
a gRPC/REST server that exposes the API that is consumed by the Web UI, CLI, and CI/CD systems

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 Run 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