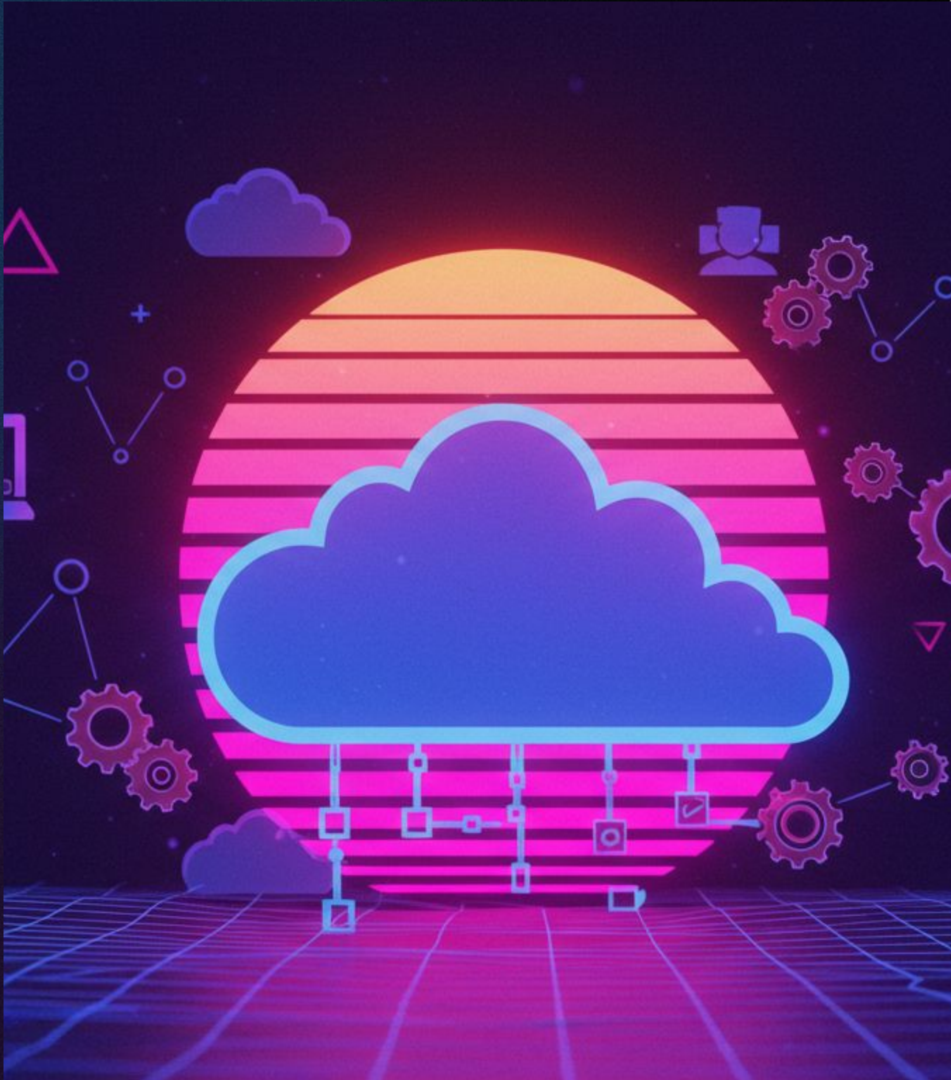
Bring Joy to your Deploy[ments]

Murriel McCabe

Elizabeth Ponce

SCALE 22x: March 7 2025
https://linktr.ee/deploywithjoy



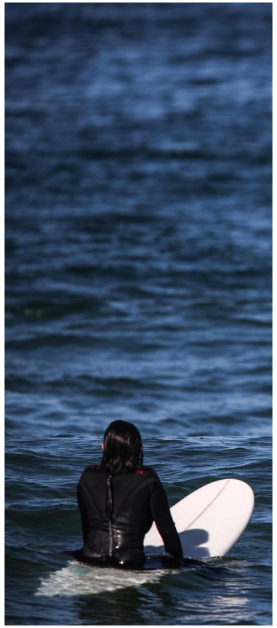
Elizabeth Ponce

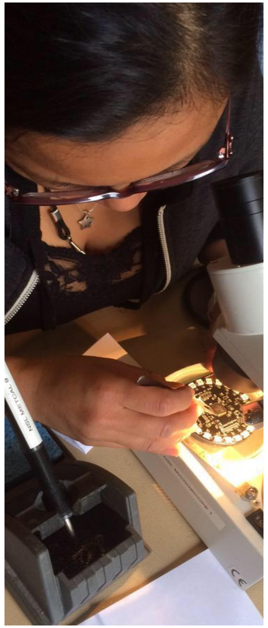
```yaml
apiVersion: scale/25
kind: Bio
metadata:
  name: elizabeth
  labels:
    job: swe search infrastructure
    location: portland
    company: airbnb
spec:
  replicas: 3
    hobbies:
      - name: running
      - name: singing
      - name: community
```

All opinions in this talk are mine!

Murriel Grace McCabe

opinions mine!





```
apiVersion: scale/v25
kind: Bio
metadata:
  name: murriel
  labels:
    job: cloud
    location: long beach
    company: google
spec:
  replicas: 4
    hobbies:
       - name: makingthings
       - name: gardening
       - name: community
       - name: goingoutside
---
apiVersion: russianblues.cat/v1
kind: Cat
metadata:
  name: orion
  ...
  name: andromeda
```

www.murrielgrace.com

Today's Focus

Deploying containerized applications to Kubernetes

Get Ready For 45 Minutes About:

- Fundamentals of CI/CD
- Overview of Open Source Tools
- Making Decisions about Tools
- Deployment Demo

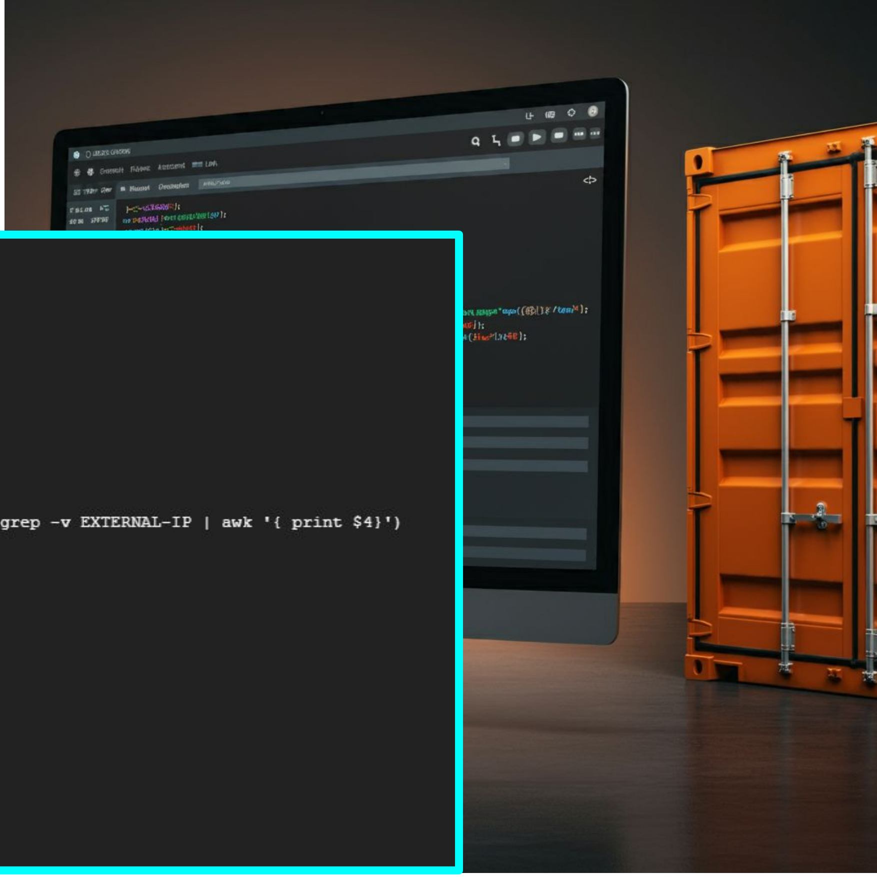
this is a

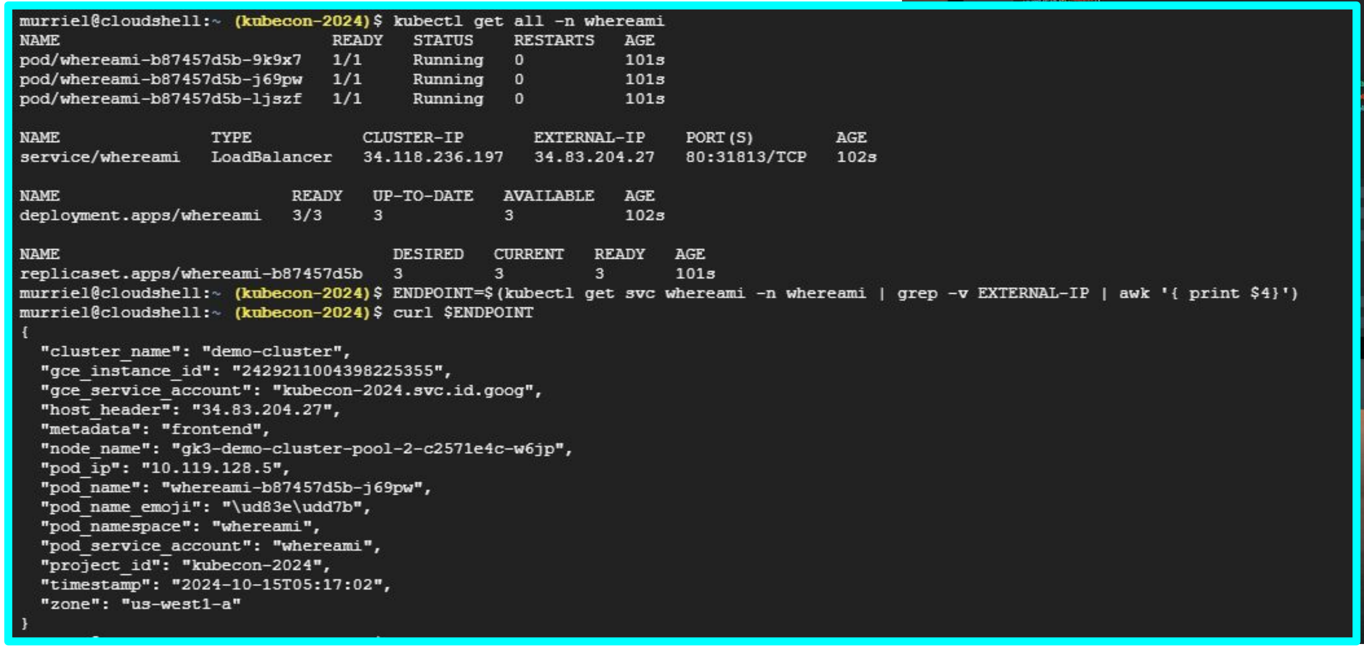
BEGINNER FRIENDLY

talk!

Demo App

whereami





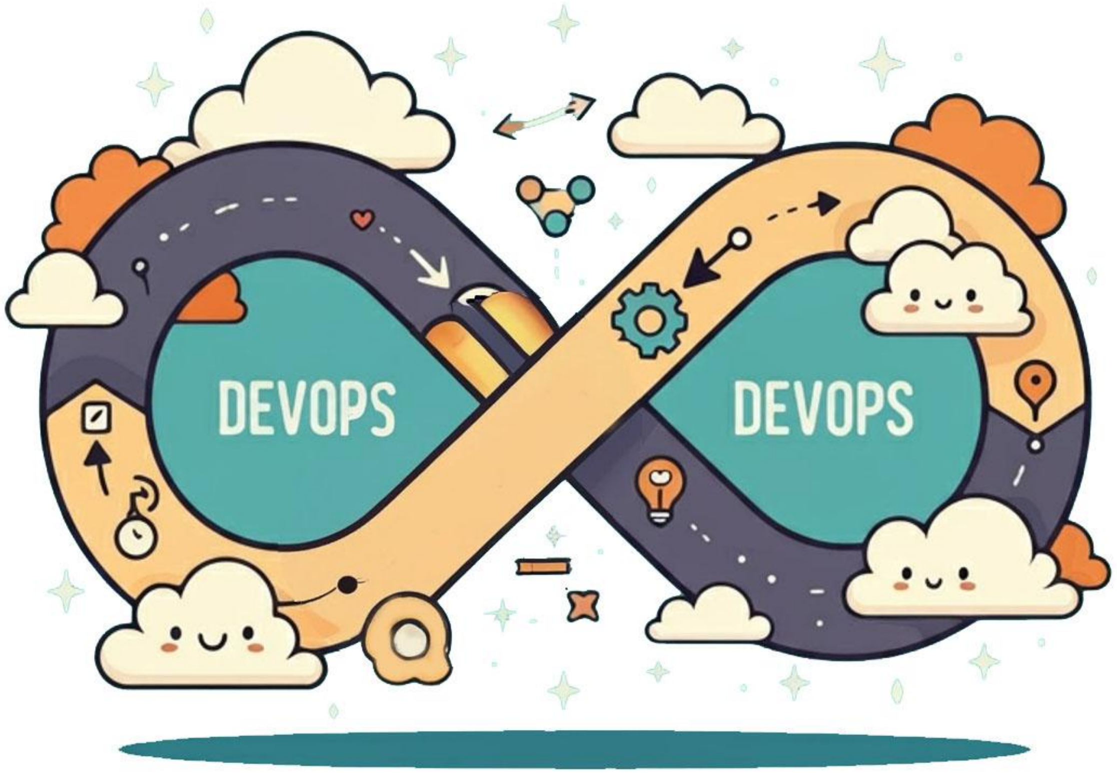
```
murriel@cloudshell:~ (kubecon-2024)$ kubectl get all -n whereami
NAME                           READY   STATUS    RESTARTS   AGE
pod/whereami-b87457d5b-9k9x7   1/1     Running   0          101s
pod/whereami-b87457d5b-j69pw   1/1     Running   0          101s
pod/whereami-b87457d5b-ljszf   1/1     Running   0          101s

NAME               TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
service/whereami   LoadBalancer   34.118.236.197   34.83.204.27   80:31813/TCP   102s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/whereami   3/3     3            3           102s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/whereami-b87457d5b   3         3         3       101s
murriel@cloudshell:~ (kubecon-2024)$ ENDPOINT=$(kubectl get svc whereami -n whereami | grep -v EXTERNAL-IP | awk '{ print $4}')
murriel@cloudshell:~ (kubecon-2024)$ curl $ENDPOINT
{
  "cluster_name": "demo-cluster",
  "gce_instance_id": "2429211004398225355",
  "gce_service_account": "kubecon-2024.svc.id.goog",
  "host_header": "34.83.204.27",
  "metadata": "frontend",
  "node_name": "gk3-demo-cluster-pool-2-c2571e4c-w6jp",
  "pod_ip": "10.119.128.5",
  "pod_name": "whereami-b87457d5b-j69pw",
  "pod_name_emoji": "\ud83e\udd7b",
  "pod_namespace": "whereami",
  "pod_service_account": "whereami",
  "project_id": "kubecon-2024",
  "timestamp": "2024-10-15T05:17:02",
  "zone": "us-west1-a"
}
```

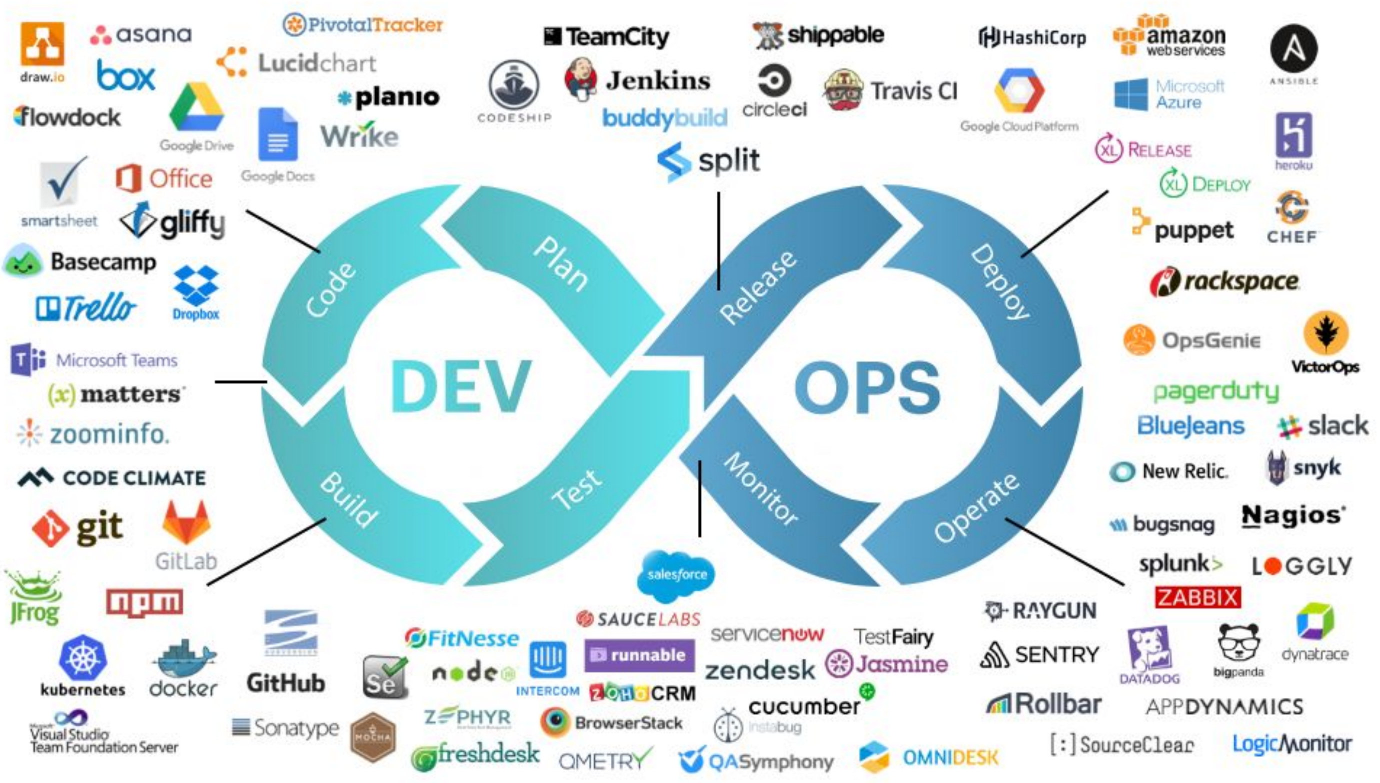
https://github.com/deploywithjoy/cicd-demos

>> *forked from https://github.com/GoogleCloudPlatform/kubernetes-engine-samples/tree/main/quickstarts/whereami*

Inner Loop ⇄ Outer Loop

Plan	Deploy
Code	Release
Build	Operate
Test	Monitor





RELEASE ENGINEERING

PLATFORM ENGINEERING

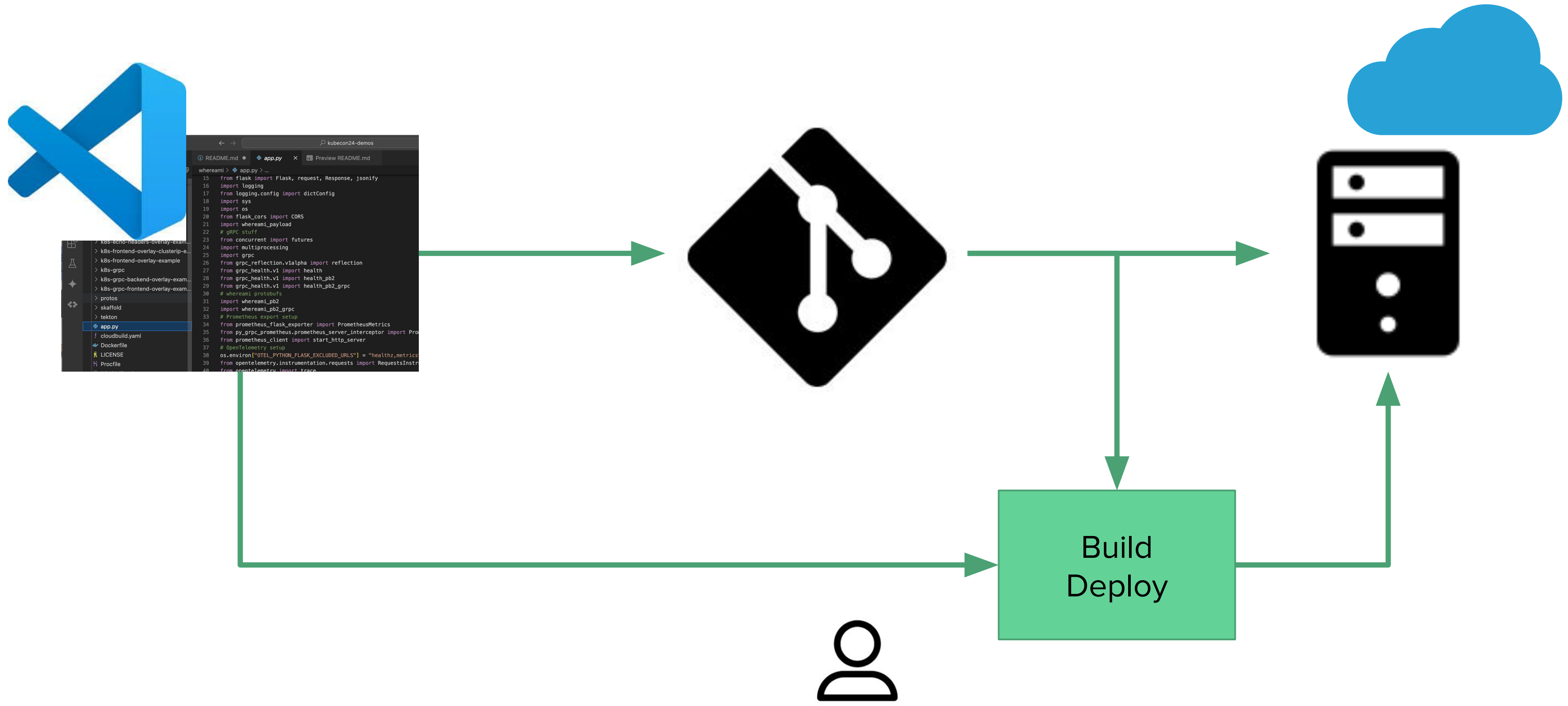
SRE

Some Development and Platform Concerns:

Hosting Environment | Version Control System | Application Code and Dependencies | Database Information | Domain and DNS | Security Information | Environment Variables | Secrets Management | Observability, Monitoring and Logging | Secure Software Supply Chain

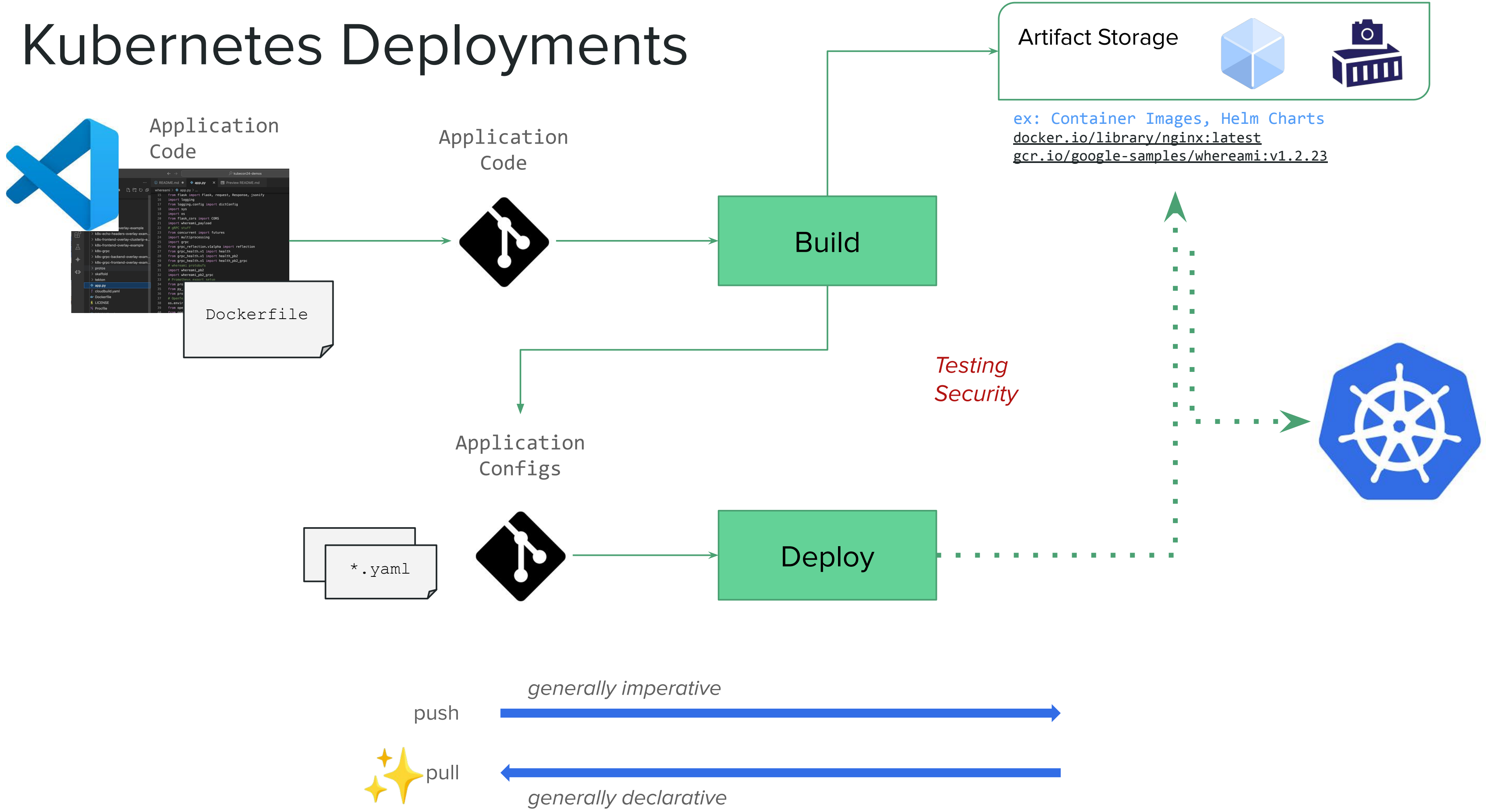
CI/CD Systems!

Basic Deployment


kubecon24-demos
README.md
app.py
Preview README.md
whereami > app.py > ...
k8s-echo-headers-overlay-exam...
k8s-frontend-overlay-clusterip-e...
k8s-frontend-overlay-example
k8s-grpc
k8s-grpc-backend-overlay-exam...
k8s-grpc-frontend-overlay-exam...
protos
skaffold
tekton
app.py
cloudbuild.yaml
Dockerfile
LICENSE
Procfile
Build
Deploy

```python
from flask import Flask, request, Response, jsonify
import logging
from logging.config import dictConfig
import sys
import os
from flask_cors import CORS
import whereami_payload
# gRPC stuff
from concurrent import futures
import multiprocessing
import grpc
from grpc_reflection.v1alpha import reflection
from grpc_health.v1 import health
from grpc_health.v1 import health_pb2
from grpc_health.v1 import health_pb2_grpc
# whereami protobufs
import whereami_pb2
import whereami_pb2_grpc
# Prometheus export setup
from prometheus_flask_exporter import PrometheusMetrics
from py_grpc_prometheus.prometheus_server_interceptor import Pro
from prometheus_client import start_http_server
# OpenTelemetry setup
os.environ["OTEL_PYTHON_FLASK_EXCLUDED_URLS"] = "healthz,metrics
from opentelemetry.instrumentation.requests import RequestsInstr
from opentelemetry import trace
```

Kubernetes Deployments



```
from flask import Flask, request, Response, jsonify
import logging
from logging.config import dictConfig
import sys
import os
from flask_cors import CORS
import whereami_payload
# gRPC stuff
from concurrent import futures
import multiprocessing
import grpc
from grpc_reflection.v1alpha import reflection
from grpc_health.v1 import health
from grpc_health.v1 import health_pb2
from grpc_health.v1 import health_pb2_grpc
# whereami protobufs
import whereami_pb2
import whereami_pb2_grpc
# Prometheus export setup
from pro
from py_
from pro
# OpenTe
os.envir
from ope
```

ex: Container Images, Helm Charts
docker.io/library/nginx:latest
gcr.io/google-samples/whereami:v1.2.23

SKAFFOLD

Developer Workflow

Kustomize.io

HELM

Templating | Packaging | Configuration

TEKTON

argo

JENKINS X

Jenkins

CI/CD Tools

Manual Deploy

imperative

```
$ cat << EOF | kubectl create -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-scale
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-scale
  template:
    metadata:
      labels:
        app: hello-scale
    spec:
      containers:
      - name: hello-scale
        image: gcr.io/google-samples/whereami:v1.2.22
        ports:
        - containerPort: 8080
EOF
```

Manual Deploy

declarative

```
$ kubectl apply -f whereami.yaml

$ kubectl apply -f whereami/
```

imperative

```
$ kubectl run \
--image=us-docker.pkg.dev/google-samples/containers/gke/whereami:v1.2.22
\ --expose --port 8080 whereami
```

Skaffold

Skaffold is an open-source tool developed by Google that simplifies and streamlines the development workflow for containerized applications

- Lightweight and Client-Side
- Cross-Environment Compatibility
- Optimized Development Loop

```
$ skaffold dev

$ docker build

$ docker tag

$ docker push

# edit Kubernetes manifests

$ kubectl apply -f

$ kubectl logs
```

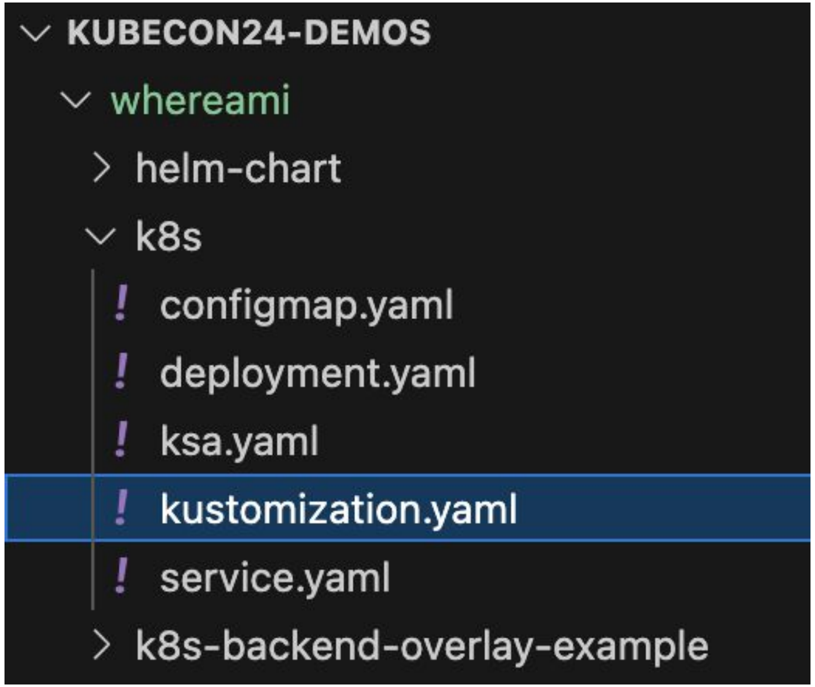
https://skaffold.dev/docs/init/

Kustomize

- Manages Kubernetes configuration without templates
- Uses base and overlay system for easy customization
- Streamlines management of different environments
- Generates resources like ConfigMaps and Secrets
- Organizes and composes resource collections

Kustomize - Key Concepts

- Layering: Base, Overlays, Patches
- Base: Defines common settings.
- Overlays: Modify base for specific environments.
- Patches: Make changes without altering originals.



```
KUBECON24-DEMOS
  whereami
    helm-chart
    k8s
      configmap.yaml
      deployment.yaml
      ksa.yaml
      kustomization.yaml
      service.yaml
    k8s-backend-overlay-example
```

Using Kustomize

```
~/whereami
├── k8s
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
├── k8s-backend-overlay-example
│   ├── cm-flag.yaml
│   ├── kustomization.yaml
│   └── service-type.yaml
└── k8s-frontend-overlay-clusterip-example
       ├── cm-flag.yaml
       ├── kustomization.yaml
       └── service-type.yaml
```

```
kubectl apply -k k8s
serviceaccount/whereami created
configmap/whereami created
service/whereami created
deployment.apps/whereami created
```

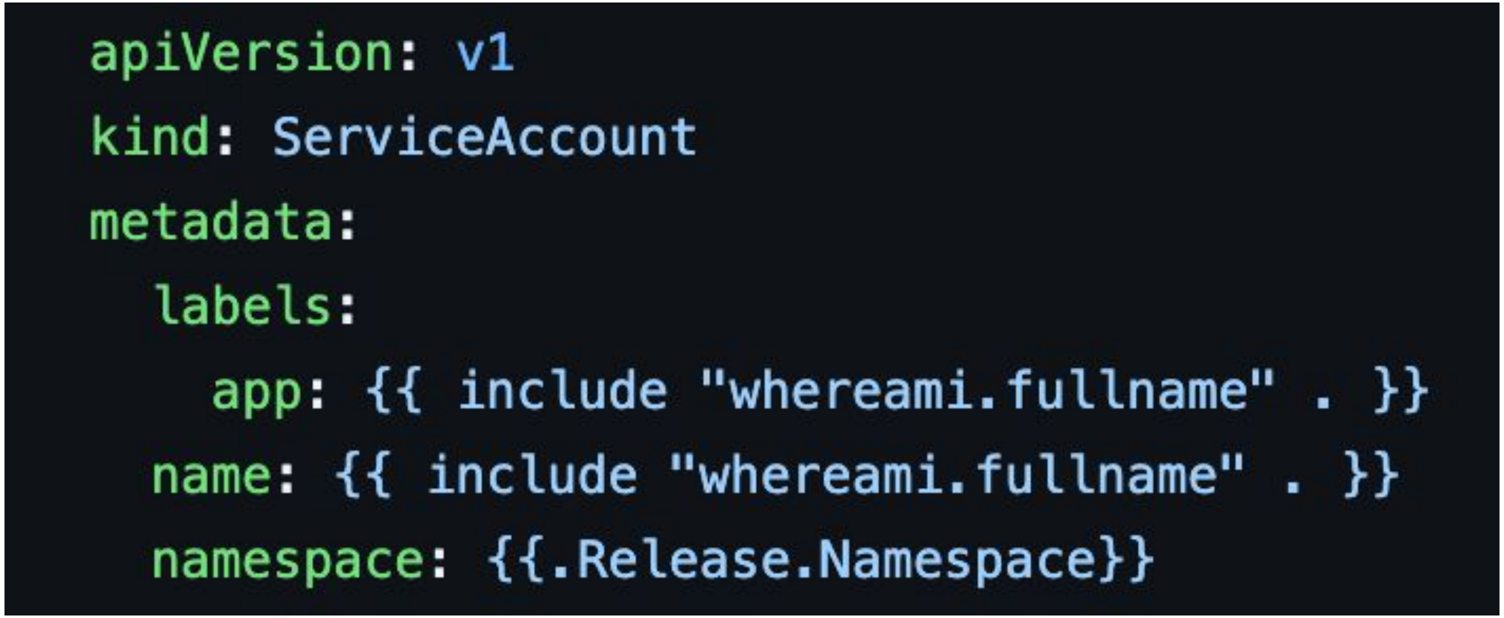
```
resources:
 -   ksa.yaml
 -   deployment.yaml
 -   service.yaml
 -   configmap.yaml
```

base

```
nameSuffix: "-backend"
commonLabels:
  app: whereami-backend
resources:
- ../k8s
patches:
- path: cm-flag.yaml
  target:
    kind: ConfigMap
- path: service-type.yaml
  target:
    kind: Service
```

overlay

Visual example of a templating pattern



```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app: {{ include "whereami.fullname" . }}
  name: {{ include "whereami.fullname" . }}
  namespace: {{.Release.Namespace}}
```

Helm - Key Concepts

HELM

The package manager for Kubernetes

Package manager like apt or yum
Around since 2015

Helps manage Kubernetes applications

Helm charts help define, install, and upgrade kube apps

Helm manages dependencies within different components in an application

image from https://helm.sh/

Helm - Charts!

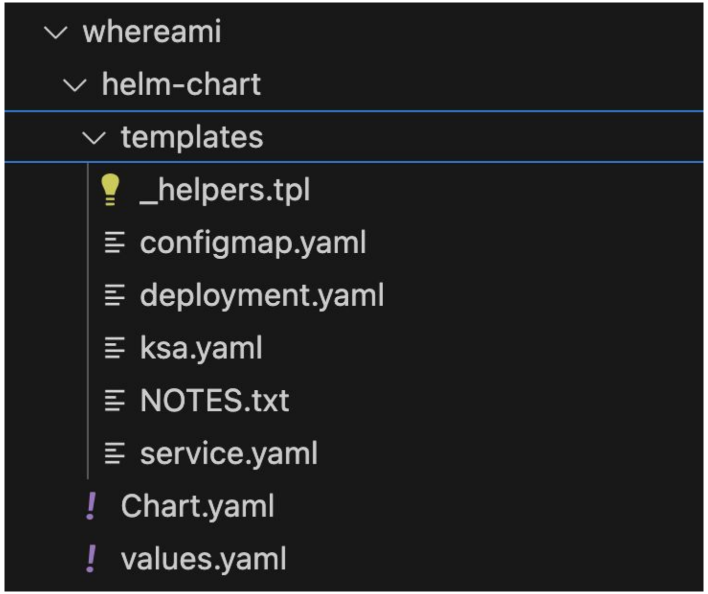
A Helm chart is a collection of files that describe a related set of Kubernetes resources and make it easy to version and share applications.

Charts are easy to create, version, share and publish.

Charts typically include:

- YAML manifests for Kubernetes resources
- Templates for generating Kubernetes manifest files
- Values files for configuring the templates
- Metadata about the chart itself

helm/chartmuseum



```
whereami
  helm-chart
    templates
      _helpers.tpl
      configmap.yaml
      deployment.yaml
      ksa.yaml
      NOTES.txt
      service.yaml
    Chart.yaml
    values.yaml
```

Set up Helm and install a Chart

```
$ brew install helm
$ helm search hub
```

```
URL                                                 CHART VERSION   APP VERSION        DESCRIPTION
https://artifacthub.io/packages/helm/romanow-he...     1.5.0          8.3.4              Grafana allows you to query, visualize, alert o...
https://artifacthub.io/packages/helm/romanow-he...     1.5.0          1.8.4              The Time Series Data Platform where developers ...
```

```
$ helm repo add myrepo https://path.to.repo

$ helm install whereami \
oci://us-docker.pkg.dev/google-samples/charts/whereami \
-version 1.2.23
```

Pipelines

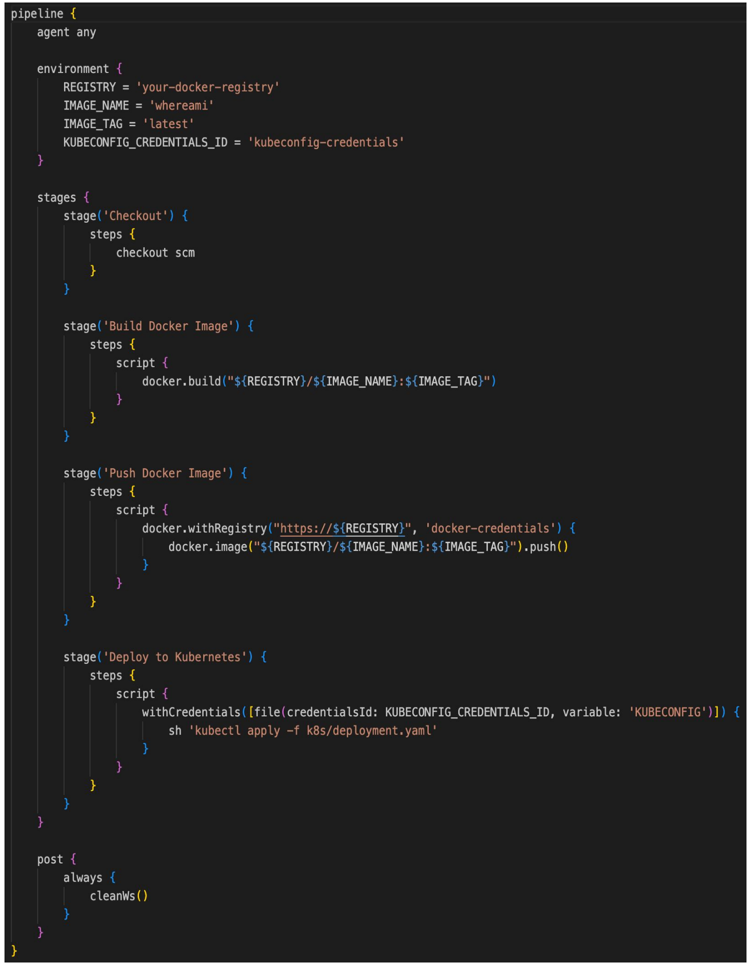
- Set of Tasks executed in defined order
- Built-in support for common CI/CD tasks (build, test, deploy)
- Conditional execution and parallel tasks
- Pipelines create an audit trail in code
- Generally cover source control, build, test, staging, and finally deploy

Jenkins - Key Concepts

- Step: a single task!
- Stage: defines a conceptually distinct subset of tasks performed through the entire Pipeline
- Node: a machine which is part of the Jenkins environment and is capable of executing a Pipeline.

Jenkins - Pipelines

- Scripted Pipelines offer flexibility and control
- Declarative Pipelines use simpler, predefined structure
- Jenkinsfiles, using Groovy syntax, define Pipelines



```
pipeline {
    agent any

    environment {
        REGISTRY = 'your-docker-registry'
        IMAGE_NAME = 'whereami'
        IMAGE_TAG = 'latest'
        KUBECONFIG_CREDENTIALS_ID = 'kubeconfig-credentials'
    }

    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    docker.build("${REGISTRY}/${IMAGE_NAME}:${IMAGE_TAG}")
                }
            }
        }

        stage('Push Docker Image') {
            steps {
                script {
                    docker.withRegistry("https://${REGISTRY}", 'docker-credentials') {
                        docker.image("${REGISTRY}/${IMAGE_NAME}:${IMAGE_TAG}").push()
                    }
                }
            }
        }

        stage('Deploy to Kubernetes') {
            steps {
                script {
                    withCredentials([file(credentialsId: KUBECONFIG_CREDENTIALS_ID, variable: 'KUBECONFIG')]) {
                        sh 'kubectl apply -f k8s/deployment.yaml'
                    }
                }
            }
        }
    }

    post {
        always {
            cleanWs()
        }
    }
}
```

Tekton

Launched as part of Google’s Knative project

Spun off in 2019 to the Continuous Delivery Foundation

Runs as an extension on clusters

Provides a set of building blocks for creating CI/CD pipelines that can build, test, and deploy across multiple cloud providers or on-prem systems.

CLI - tkn

https://raw.githubusercontent.com/cdfoundation/artwork/main/tekton/horizontal/color/tekton-horizontal-color.svg

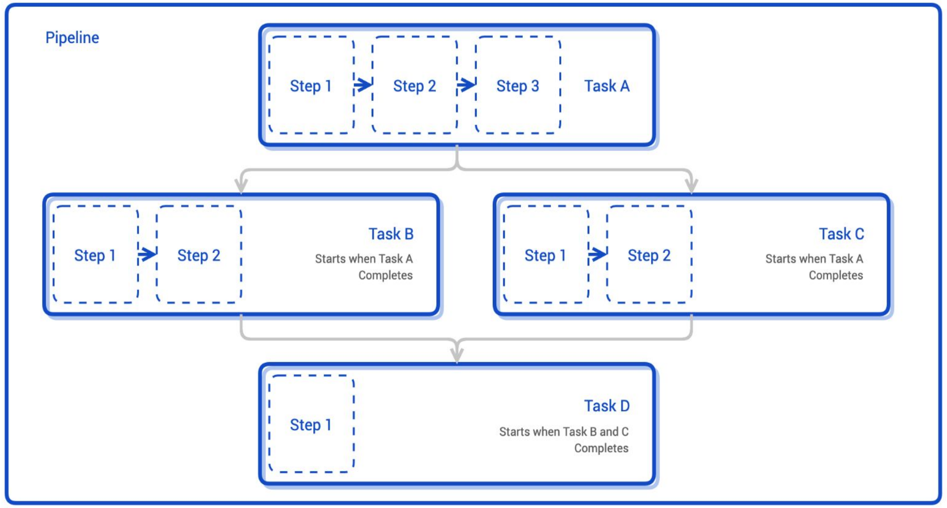
Tekton - Key Concepts

Steps - most basic unit and represents a specific operation in a CI/CD workflow

Tasks - collection of ordered steps

Pipelines - collection of tasks

Triggers - event-based pipelines


Pipeline
Step 1
Step 2
Step 3
Task A
Step 1
Step 2
Task B
Starts when Task A Completes
Step 1
Step 2
Task C
Starts when Task A Completes
Step 1
Task D
Starts when Task B and C Completes

https://tekton.dev/docs/concepts/overview/

Tekton - Set Up a TaskRun

```
kubectl apply —f hello-world.yaml
task.tekton.dev/hello created
```

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: hello
spec:
  steps:
    - name: echo
      image: alpine
      script: |
        #!/bin/sh
        echo "Hello World"
```

```
kubectl apply —f hello-world-run.yaml
taskrun.tekton.dev/hello-task-run created
```

```
apiVersion:
tekton.dev/v1beta1
kind: TaskRun
metadata:
  name:
hello-task-run
spec:
  taskRef:
    name: hello
```

```
kubectl get taskrun hello-task-run
NAME             SUCCEEDED   REASON      STARTTIME   COMPLETIONTIME
hello-task-run   True        Succeeded   15s         1s
```

https://tekton.dev/docs/concepts/overview/

Argo CD

Declarative, GitOps continuous delivery tool for Kubernetes

Argo Events

Event based dependency management for Kubernetes

Argo Workflows

Kubernetes-native workflow engine supporting DAG and step-based workflows

Argo Rollouts

Advanced Kubernetes deployment strategies such as Canary and Blue-Green

https://github.com/deploywithjoy/cicd-demos/tree/develop/whereami/argo

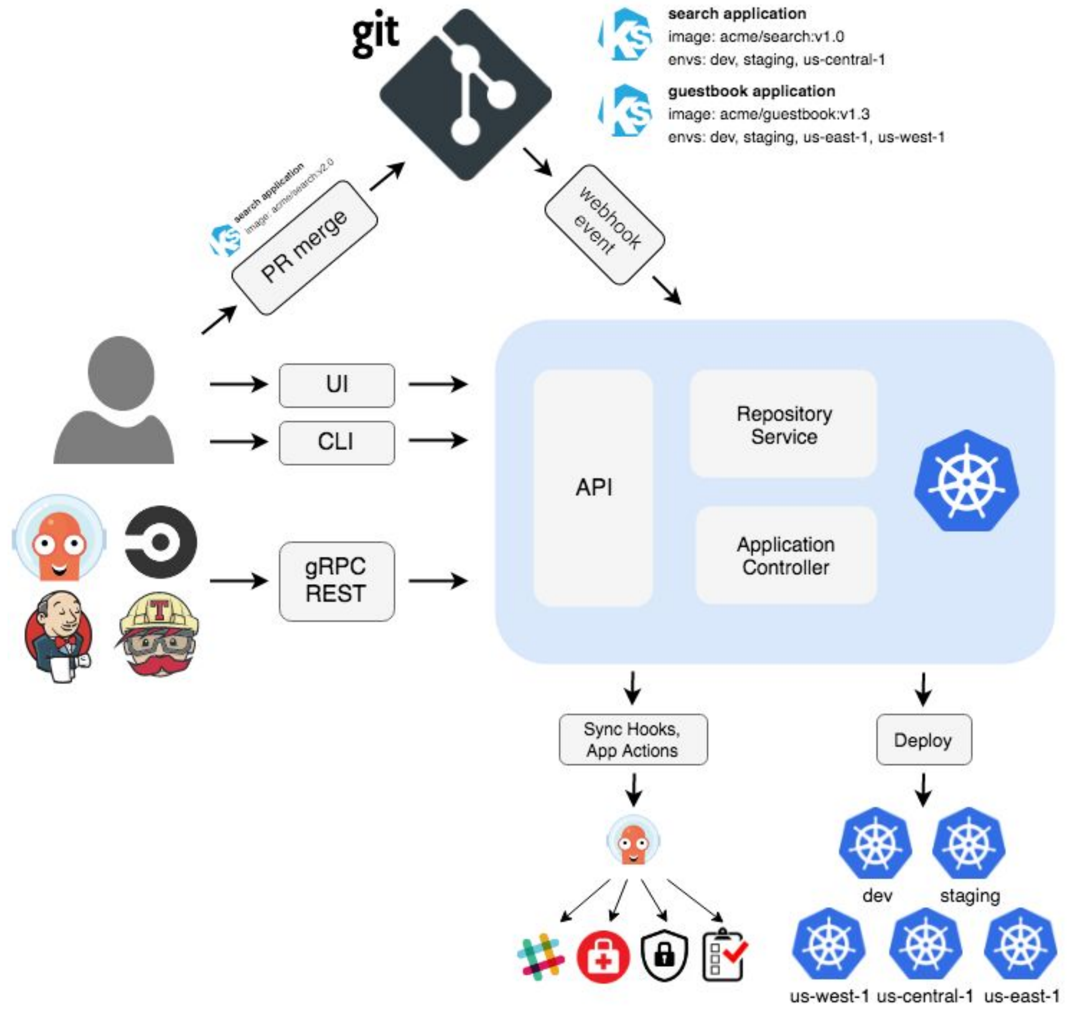
Argo CD

Focus on the “CD” part of CI/CD

Application definitions, configurations, and environments should be declarative and version controlled.

Application deployment and lifecycle management should be automated, auditable, and easy to understand.

https://github.com/argoproj/argo-cd



Key Resources

Application

An instance of an application defined by git source and destination cluster/namespace

Project

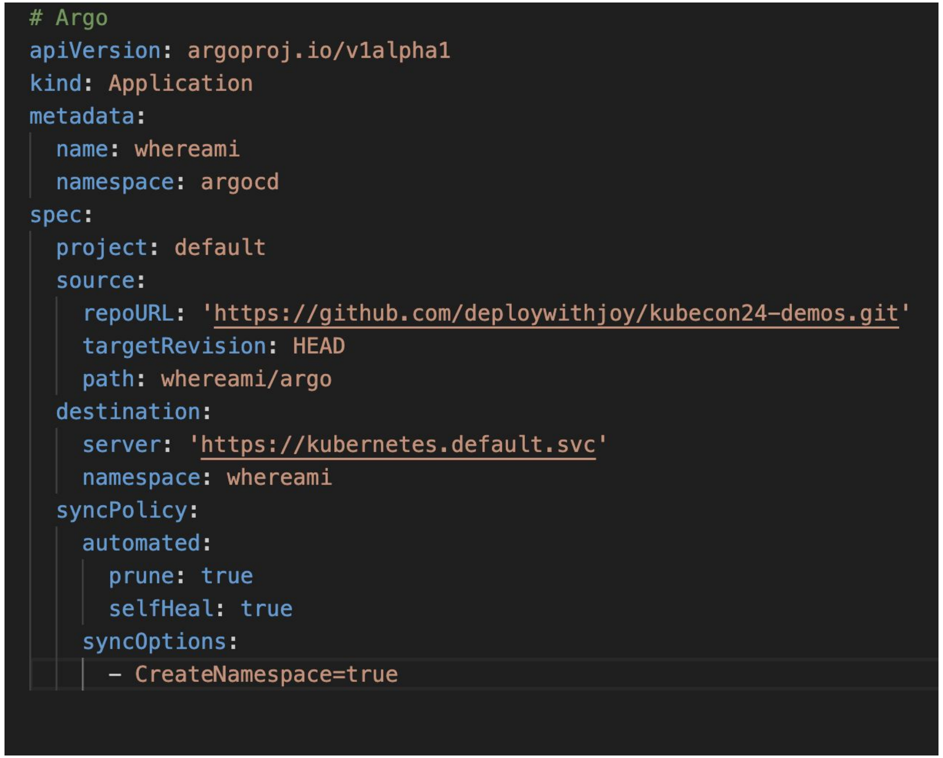
A group of Applications

Repo

Repository as secret

Kubernetes Manifests

kustomize | helm | yaml | jsonnet



```yaml
# Argo
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: whereami
  namespace: argocd
spec:
  project: default
  source:
    repoURL: 'https://github.com/deploywithjoy/kubecon24-demos.git'
    targetRevision: HEAD
    path: whereami/argo
  destination:
    server: 'https://kubernetes.default.svc'
    namespace: whereami
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true
```

Setup Argo

```
$ kubectl create namespace argocd

$ kubectl apply -n argocd -f
    \ https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

Use Argo

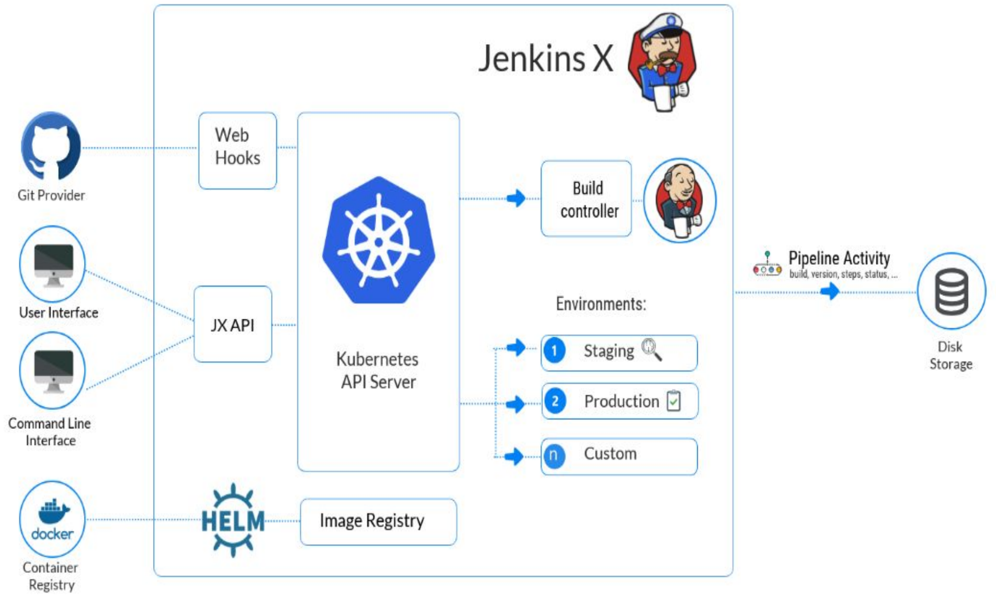
```
$ argocd app list
```

…or Argo Dashboard or API

Jenkins X

- GitOps
- Tekton Pipeline Orchestration
- ChatOps Integration
- Developer Experience and CLI
- Integration with Open-Source Projects

Jenkins X - Key Concepts



https://jenkins-x.io/about/concepts/

Source Repositories: Creates file structure and sets up tools.

Environments: Handles builds between environments using GitOps.

Pipelines!

Kubernetes Integration: Creates and configures Kubernetes cluster.

Single Command Operations: Create Git repos, webhooks, CD pipelines.

Jenkins X - ChatOps commands examples

`/lgtm`	This PR looks good to me - this command can be from anyone with access to the repo who is in the `OWNERS` file
`/test this`	Run the default test pipeline context for this PR
`/test (context)`	Run a specific test pipeline context by name
`/retest`	Rerun any failed test pipeline contexts for this PR
`/override (context)`	Override a failed pipeline context
`/hold`	Set this PR to not automerge even if it has been set `lgtm` and approved
`/hold cancel`	remove the `hold` label from the PR, allowing automerge
`/assign (user)`	assign the PR to the given (`user`)
`/cc (user)`	add the given `user` as a reviewer for the PR

So what tool should I use?

It depends!

Summary of CI/CD Tools

Tool	Focus	GitOps	Push or Pull	Foundation	K8S Native	CI	CD
ArgoCD	Declarative GitOps continuous delivery	x	Pull	CNCF Graduated	x	Via Argo events	X
Tekton	Cloud Native CI/CD pipelines; CI/CD framework	some	**Pull**	CNCF, CDF*	x	Building blocks	Building blocks
Jenkins	Traditional CI/CD	some	Both*	CNCF, CDF		x	x
Jenkins X	All In One CI/CD for Kubernetes (including ChatOps)	x	Pull	CNCF, CDF	x	x	x

**Continuous Delivery Foundation*

Working Together

skaffold

 deploy with kubectl

 deploy or render with helm

 render with kustomize

argo

 App delivery (and self install!) using kustomize, helm, yaml

jenkins

 kubernetes plugin: kubectl (kustomize, helm, yaml)

jenkins-x 💙 tekton (see catalog of tasks)

everyone gets a helm chart

Resources

kubectl

Skaffold

Kustomize

Helm

cd.foundation

ArgoCD

Tekton

Jenkins

Jenkins X

https://github.com/deploywithjoy/cicd-demos

Network and Find Your Pals at SCaLE!

KubeCon | CloudNativeCon
North America 2023

KubeCon | CloudNativeCon
North America 2024

Actual Footage of Murriel and Elizabeth

KubeCon + CloudNativeCon | 2023 - Chicago | 2024 - Salt Lake City

Demo!

Thank you!

https://linktr.ee/deploywithjoy

/in/emcponce

/in/murrielperez/

Talk Feedback