# Developing, Testing and Optimizing Microservice Applications on Kubernetes


Develop a passion for learning.

TEKsystems Global Services

# WORKFORCE DEVELOPMENT

PARTICIPANT GUIDE

# Content Usage Parameters

**Content** refers to material including instructor guides, student guides, lab guides, lab or hands-on activities, computer programs, etc. designed for use in a training program

1

Content is subject to copyright protection

2

Content may only be leveraged by students enrolled in the training program

3

Students agree not to reproduce, make derivative works of, distribute, publicly perform and publicly display content in any form or medium outside of the training program

4

Content is intended as reference material only to supplement the instructor-led training

# Overview

Microservice development on Kubernetes requires a specific methodology and tool set to make it easy to produce resilient self-healing applications. A good developer does not automatically make a good application on Kubernetes but with some foundational skills it is possible to greatly improve development velocity and the quality of the end product. Students will learn about application development, debugging, and testing with Kubernetes. Monitoring, tracing, and service mesh implementations with Kubernetes.

# Logistics

## Class Hours:

- Instructor will provide start and end times
- Breaks throughout class.

## Lunch:

- Yes, 1 hour and 15 minutes
- Extra time for email, phone calls, or simply a walk.

## Telecommunication:

- Turn off or set electronic devices to vibrate
- Reading or attending to devices can be distracting to other students

## Miscellaneous

- Courseware
- Labs

# What you'll learn

## Day 1 - Local k3d Development & Core Tools

- Set up Windows VM
- K3d Getting Started
- K3d and Persistent Volumes
- Building a Helm Chart
- Working with Operator SDK

## Day 2 - Advanced Development & Cloud Integration

- Skaffold and k3d
- Prometheus Monitoring
- Tilt and k3d
- Remote Development with Telepresence on EKS
- Skaffold Microservices

# Introduce yourself

Time to introduce yourself:

- Name you prefer
- Current role
- Familiarity with:
  - Kubernetes
  - Microservices
  - KiND/K3d
- Expectations and goals for class

# Meet The Instructor

Dwayne Pugh

DevOps and Site Reliability Engineer

Experience with public and private sector, from legacy systems to cutting edge cloud native technologies.


LinkedIn
https://www.linkedin.com/in/bpugh143/

Email
dwayne@innovationinsoftware.com


Expertise

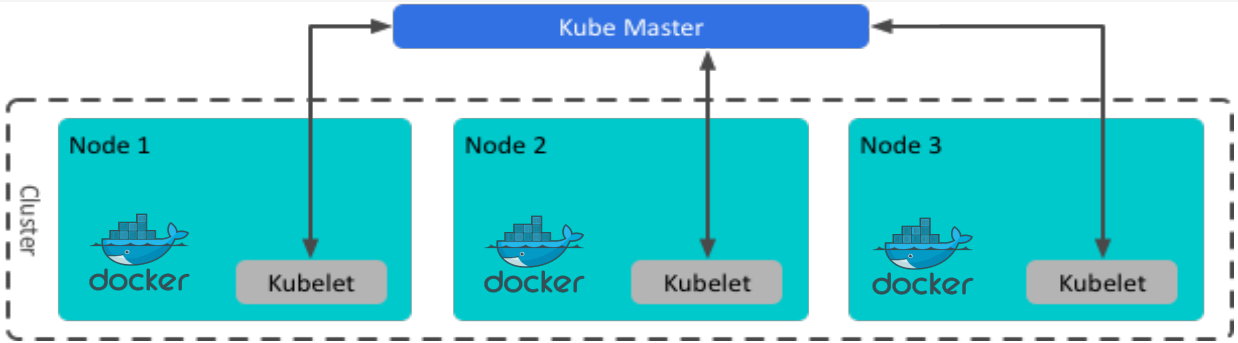
- OpenShift | AWS
- Terraform | Terragrunt | Atlantis
- Jenkins | Groovy | Ansible
- Java | Linux
- Docker | Kubernetes
- Kustomize | Helm | Ansible

# Kubernetes

The name Kubernetes originates from Greek, meaning helmsman or pilot. Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

# Kubernetes

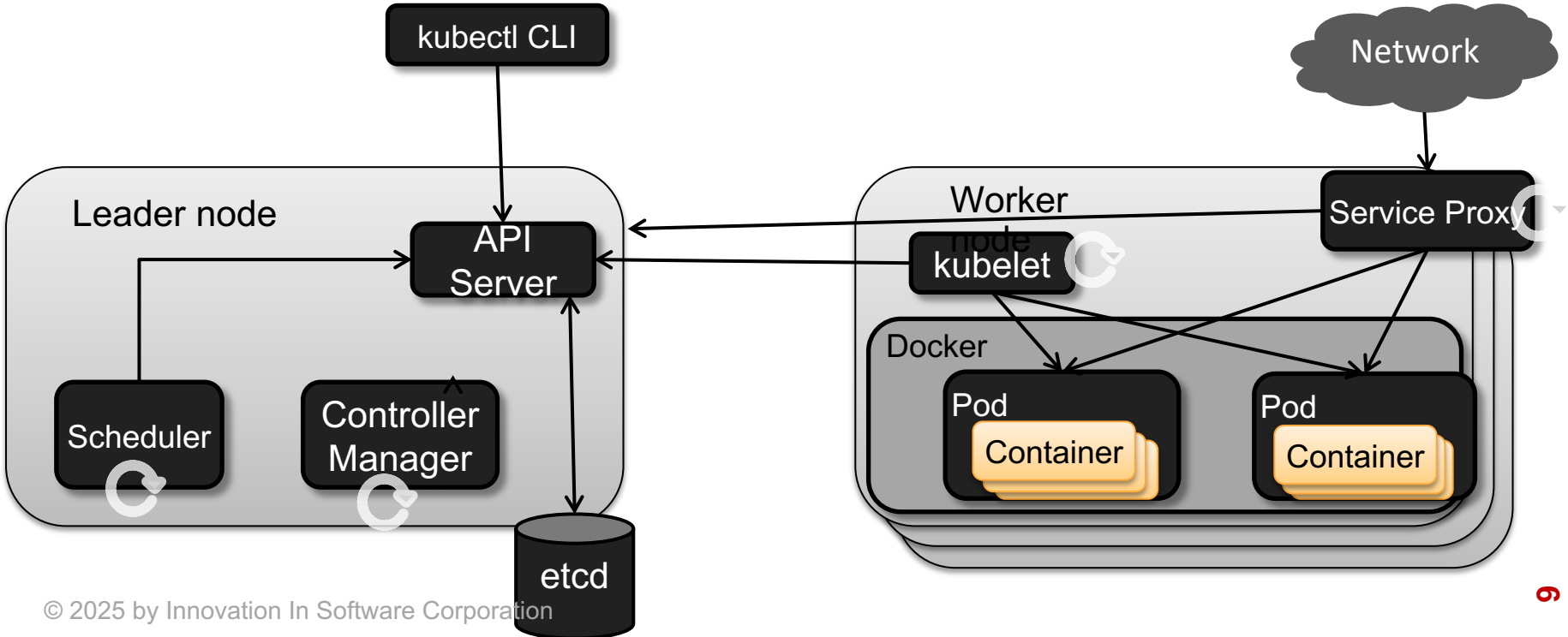
*Kubernetes provides the infrastructure for container-centric deployment and operation of applications*




- Kubernetes resource objects provide key application management features, including
  - App elasticity and self-healing
  - Naming and discovery
  - Rolling updates
  - Request load balancing
  - Application health checking
  - Log access and resource monitoring

# Kubernetes

- Kubernetes nodes can be physical hosts or VM's running a container-friendly Linux (kernel > 3.10)



# Terms and Definitions

- Pod
- Service
- Volume
- Namespace
- ReplicaSet (RS)
- Deployment
- StatefulSet
- DaemonSet
- Job

# Local Development, Testing, and Debugging with **K3X**

# What about all these K3’s

1. K3s - Lightweight Kubernetes Distribution
2. K3c - Container runtime for k3s
3. K3d - Fast spin-up of local k3s clusters

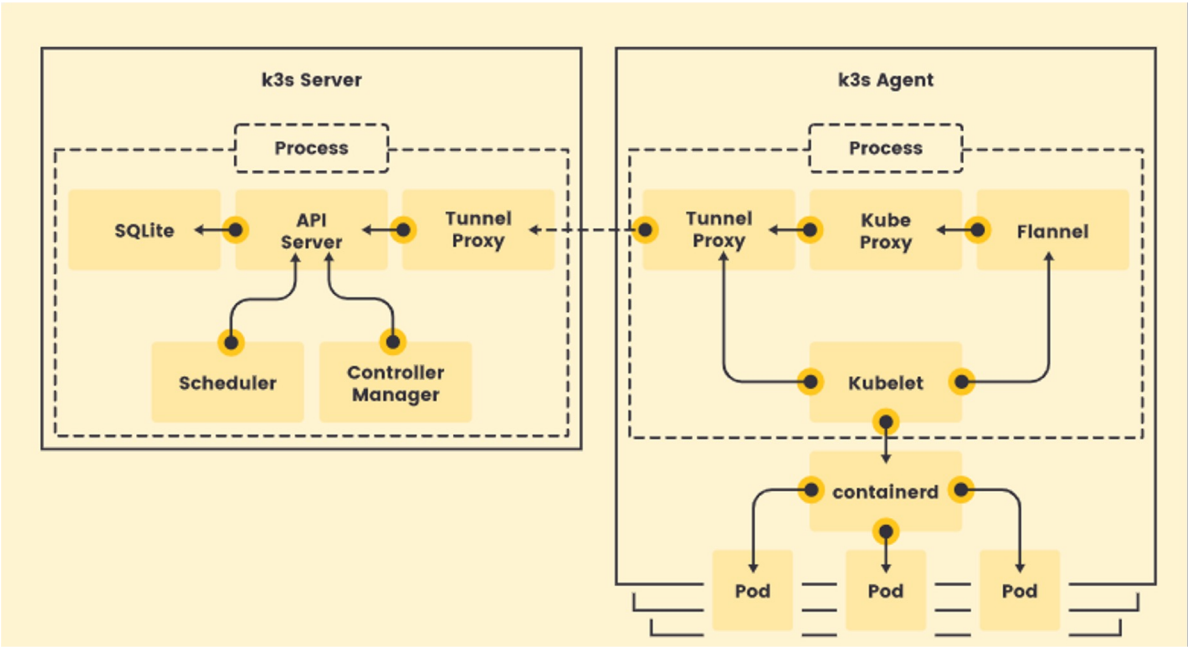
# What is k3s

Lightweight Kubernetes Distribution

# k3s Process Architecture

k3s is to get a very efficient and lightweight fully compliant Kubernetes distribution.
K3s can be install either through a simple script that will download and configure a linux binary

K3s.io



# Why k3s?

## Important distinctions

**Lightweight Kubernetes**
- Single Binary Side (-50mb)
- MemorySize (-300mb)
- Low Cognitive Load
- Perfect for Edge
- Used Just about everywhere

**Designed for production**
- Fully CNCF
- Secure by default
- Best practice defaults

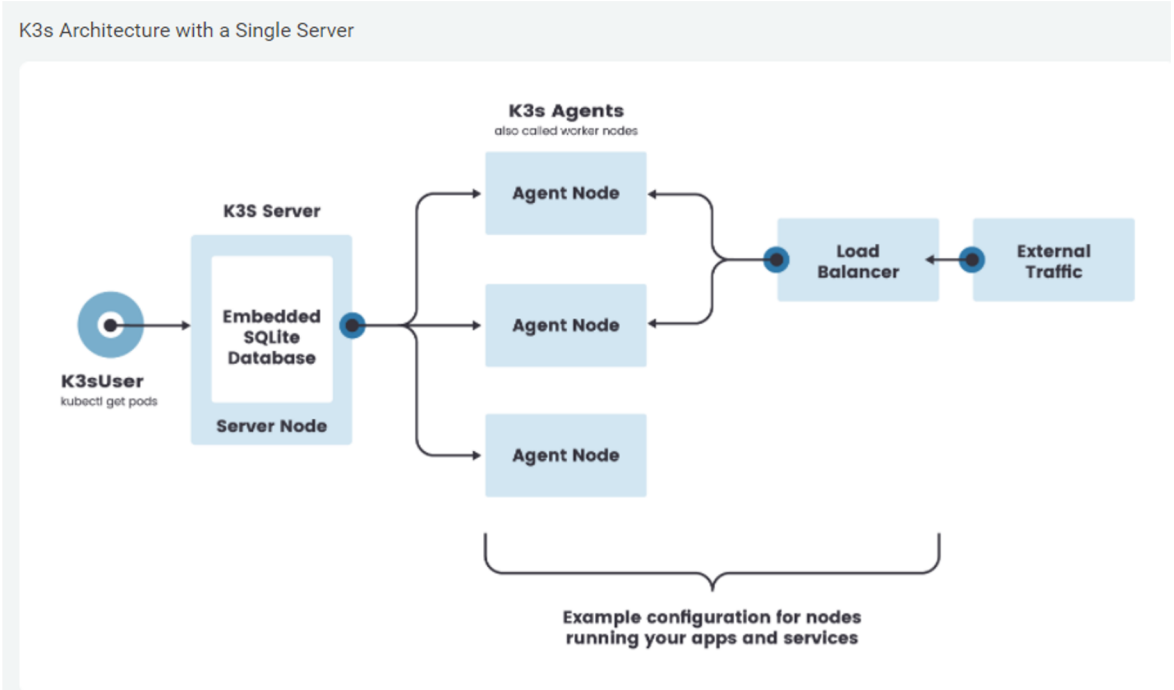
**Kubernetes Distribution**
- Opinionated
- Complete
- Strong focus on simplicity

# K3S Single Server

a cluster that has a single-node K3s server with an embedded SQLite database.

each agent node is registered to the same server node.

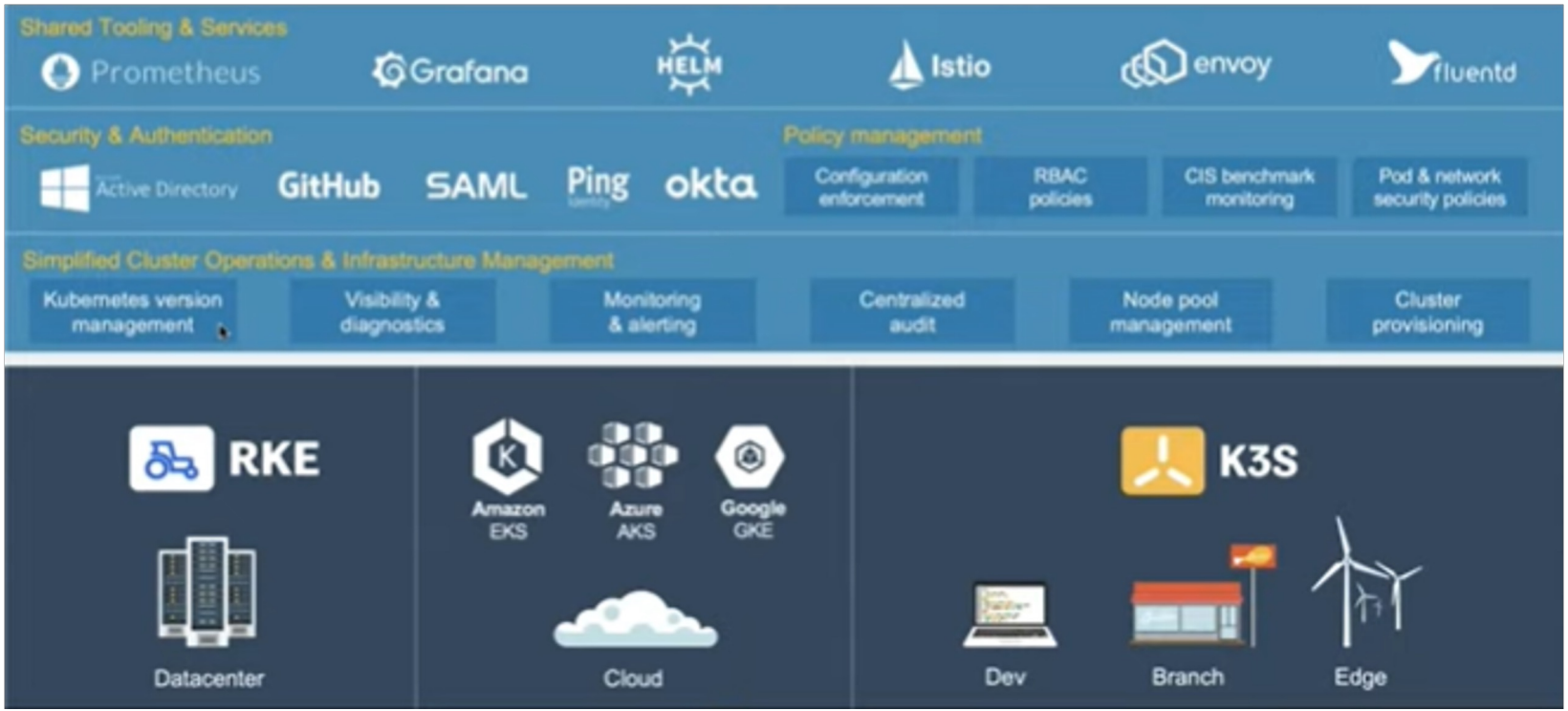
A K3s user can manipulate Kubernetes resources by calling the K3s API on the server node.



# What is k3d?

k3d is a binary that provisions a k3s kubernetes cluster on docker. Designed to easily run k3s in Docker, it provides a simple CLI to create, run, delete a full compliance Kubernetes cluster with 0 to n worker nodes.

# K3d CNCF Landscape



# Where can I find  k3d?

*https://k3d.io/*

use the install script to grab the latest release:

- wget: wget -q -O - https://raw.githubusercontent.com/rancher/k3d/main/install.sh | bash
- curl: curl -s https://raw.githubusercontent.com/rancher/k3d/main/install.sh | bash

# K3d

1. Runs k3s inside containers for development
2. Creates new cluster in seconds
3. Multiple clusters, multi node cluster
4. Runs on Docker (Docker for Desktop is the easiest route)

# K3d Creating a Cluster

Create the directory on the host where we will persist the data:

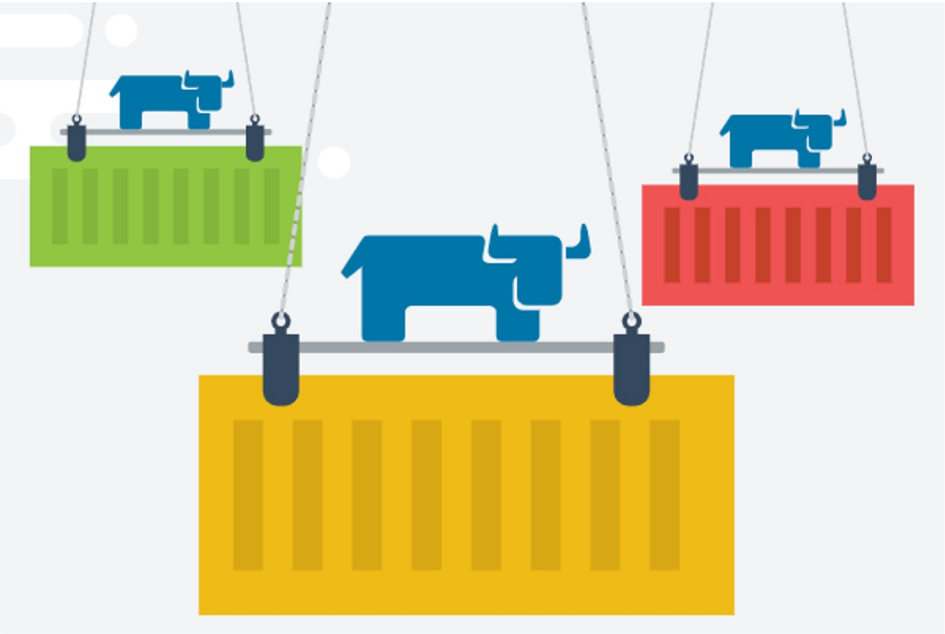
```
> mkdir -p /tmp/k3dvol
```

Create the cluster:

```
> k3d create --name "k3d-cluster" --volume /tmp/k3dvol:/tmp/k3dvol --publish "80:80" --workers 2
> export KUBECONFIG="$(k3d get-kubeconfig --name='k3d-cluster')"
```

# K3d Code Home

https://github.com/rancher/k3d/releases



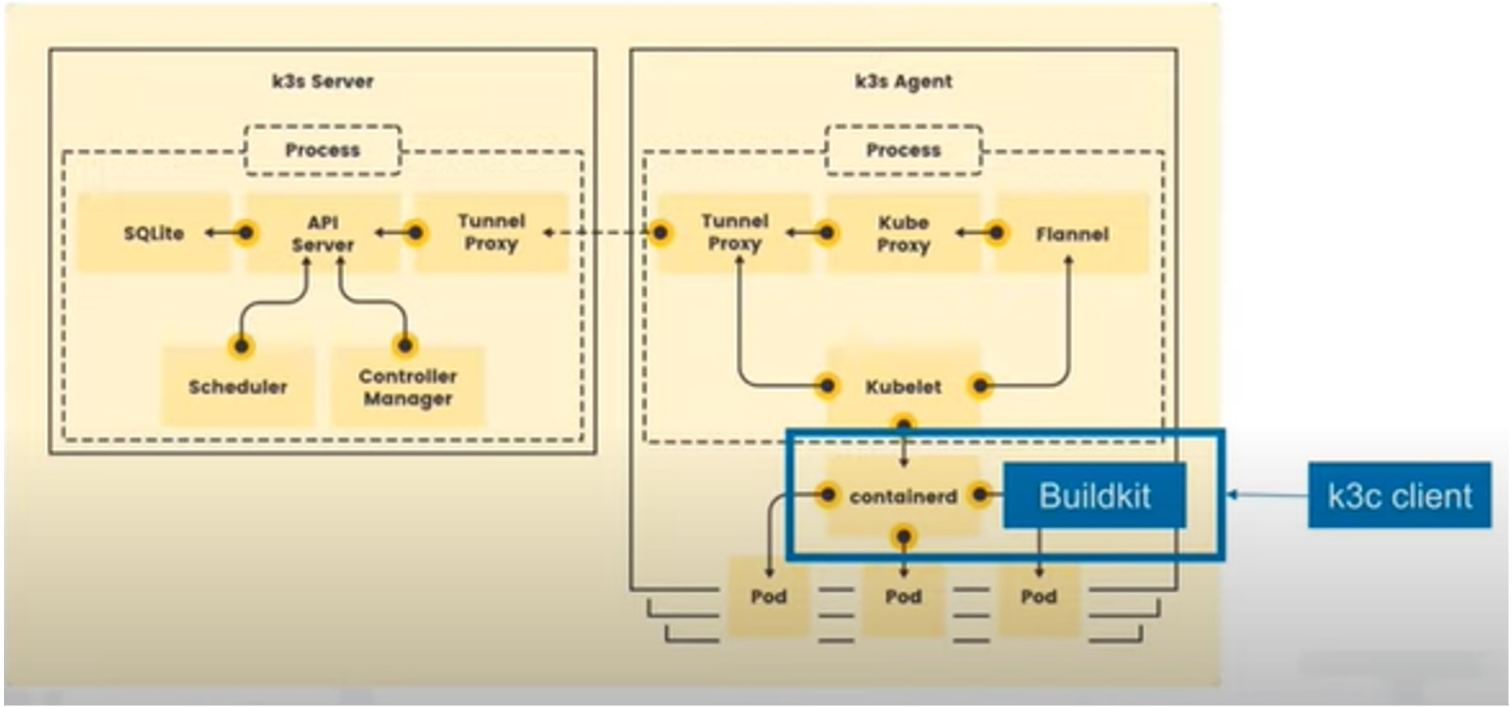
# K3d Command Tree

https://k3d.io/usage/commands/#command-tree

# What is k3c

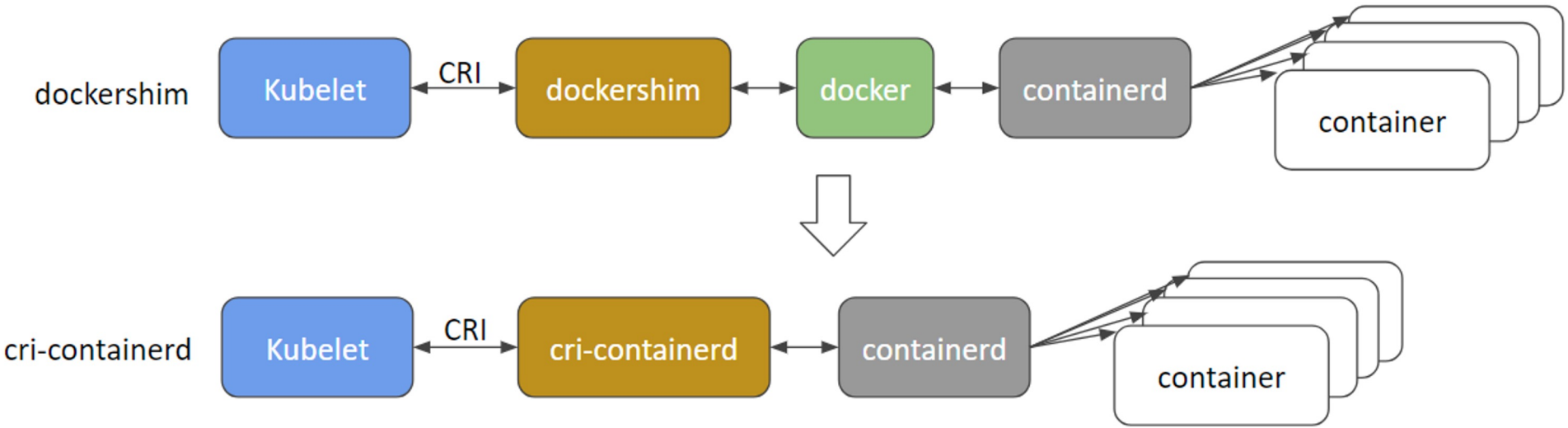
1. Enhances development and debugging flows
    a. Familiar Docker CLI Style UX
    b. Provide “build” functionality
    c. Run one off containers for development “run”
    d. Easy to debug nodes containers “ps, logs, exec”
2. New default container runtime for k3s
    a. Contanerd + buildkit +enhanced CRI

# K3c in a context



# Containerd

Containerd is an OCI compliant core container runtime designed to be embedded into larger systems. It provides the minimum set of functionality to execute containers and manages images on a node.



# K3c and Buildkit

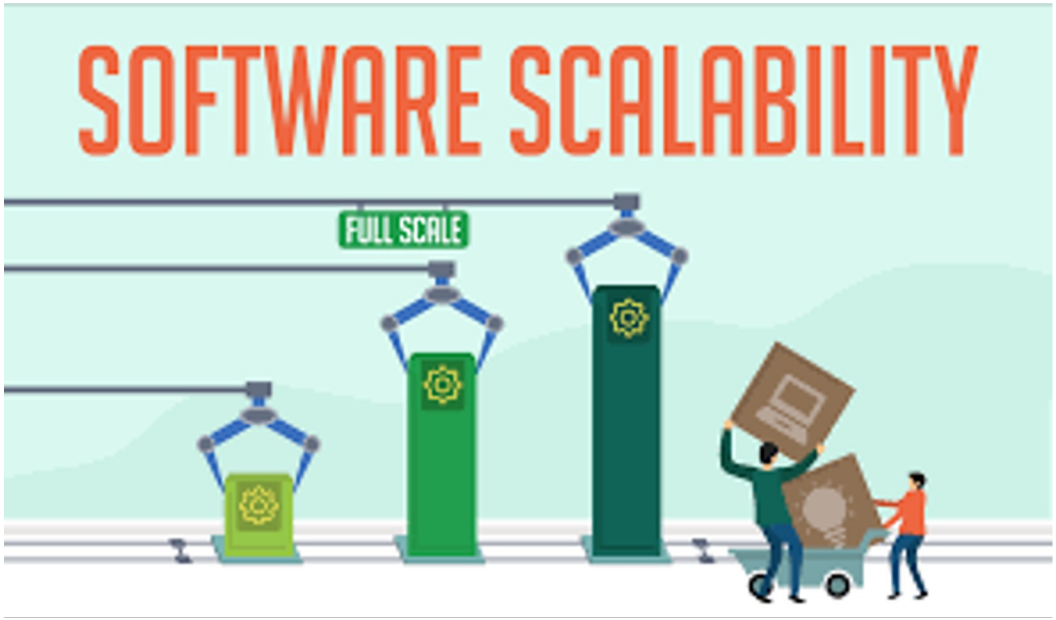
Docker Build introduced BuildKit, with improvements in performance, storage management, feature functionality, and security.

- Docker images created with BuildKit can be pushed to Docker Hub just like Docker images created with legacy build
- the Dockerfile format that works on legacy build will also work with BuildKit builds
- The new `--secret` command line option allows the user to pass secret information for building new images with a specified Dockerfile

# Scalability

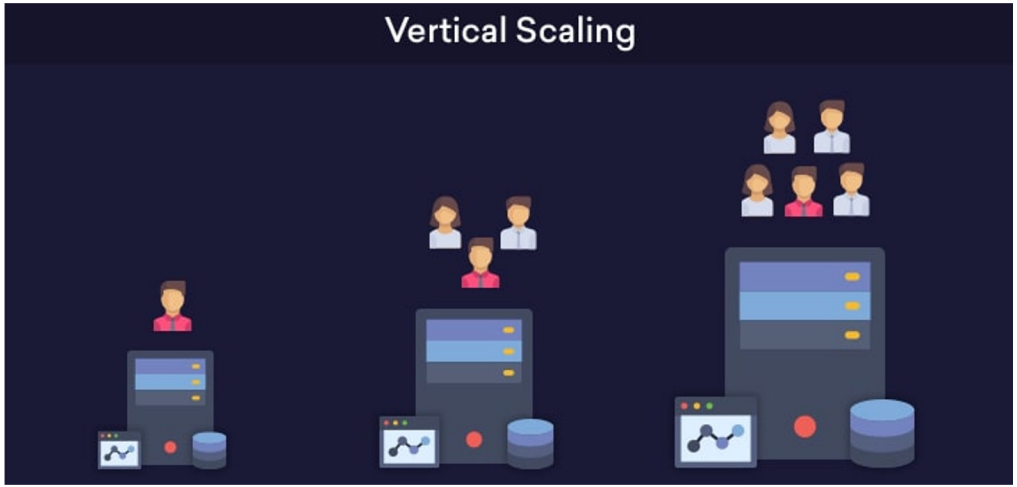
Our scalability definition is built on two concepts:

- Service Level Indicators
- Service Level Objectives

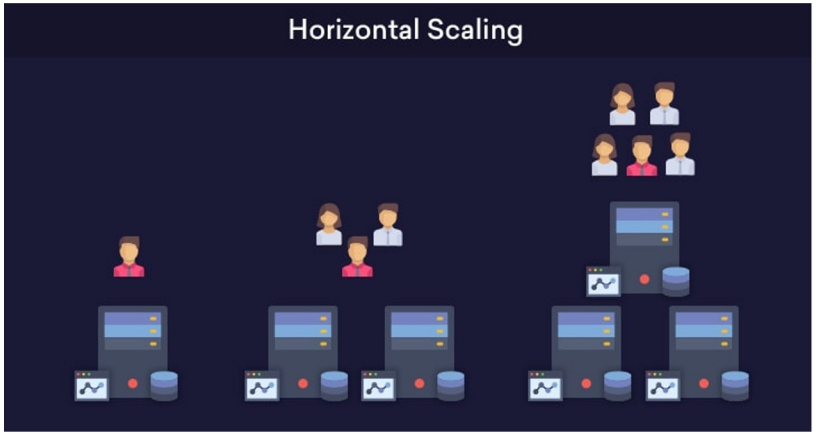


# Vertically Scalable Components

**vertical scaling** (a.k.a. “**scaling** up”), you're adding more power to your existing machine. In **horizontal scaling** (a.k.a. “**scaling** out”), you get the additional resources into your system by adding more machines to your network, sharing the processing and memory workload across multiple devices.

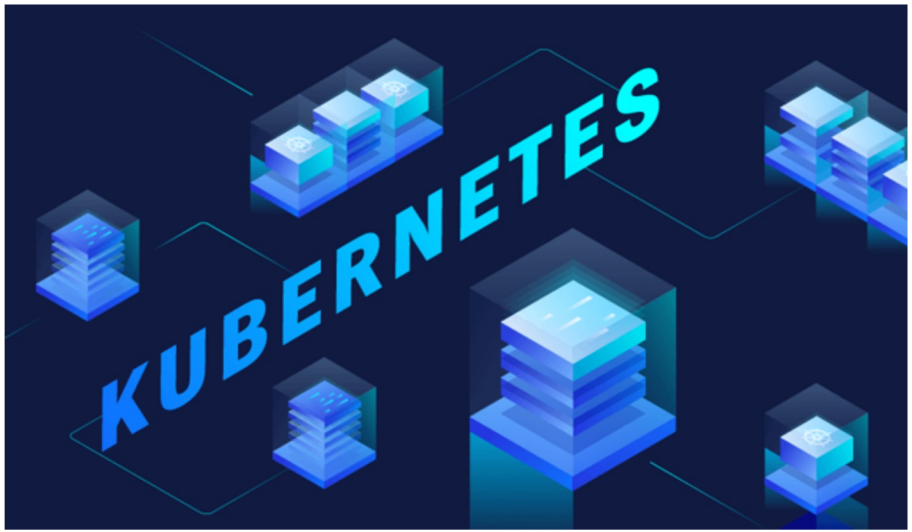


# Horizontally Scalable Components




**horizontally scalable** refers to systems whose capacity and throughput are increased by adding additional nodes. This is in distinction to **vertically** scaled systems, where adding capacity and throughput generally involves replacing smaller nodes with larger and more powerful ones.

# Building Applications Using Layers of Horizontally Scalable Components



# POP QUIZ:

## Kubernetes: Local Development, Test, and Debug

What is horizontal scalability?

A: adding capacity and throughput by replacing smaller nodes

B: Systems whose capacity and throughput are increased by adding additional nodes.

C: Adding more memory to a database.

# POP QUIZ:

## Kubernetes: Local Development, Test, and Debug

What is horizontal scalability?

A: adding capacity and throughput by replacing smaller nodes

B: Systems whose capacity and throughput are increased by adding additional nodes.

C: Adding more memory to a database.

# POP QUIZ:

## Kubernetes: Local Development, Test, and Debug

What is K3c?

A: A binary that provisions a kubernetes cluster on docker. Designed to easily run in Docker

B: Lightweight Kubernetes Distribution

C: A local container engine designed to fill the same gap Docker does in the Kubernetes ecosystem.

# POP QUIZ:

## Kubernetes: Local Development, Test, and Debug

What is K3c?

A: A binary that provisions a kubernetes cluster on docker. Designed to easily run in Docker

B: Lightweight Kubernetes Distribution

C: A local container engine designed to fill the same gap Docker does in the Kubernetes ecosystem.

# Experiment – k3d Getting Started


# Label Recommendations

| Label Example Key | Description | Label Example value |
|---|---|---|
| Application-ID/Application-name | The name of the application or its ID | my-awesome-app/app-nr-2345 |
| Version-nr | The version number | ver-0.9 |
| Owner | The team or individual the object belongs to | Team-kube/Josh |
| Stage/Phase | The stage or phase | Dev, staging, QA, Canary, Production |
| Release-nr | The release number | release-nr-2.0.1 |

# Label Recommendations

| Label Example Key | Description | Label Example value |
|---|---|---|
| Tier | Which tier the app belongs to | front-end/back-end |
| Customer-facing | Is the resource part of an app that is customer facing? | Yes/No |
| App-role | What roles does the app have | Cache/Web/Database/Auth |
| Project-ID | The associated project ID | my-project-276 |
| Customer-ID | The customer ID for the resource | customer-id-29 |

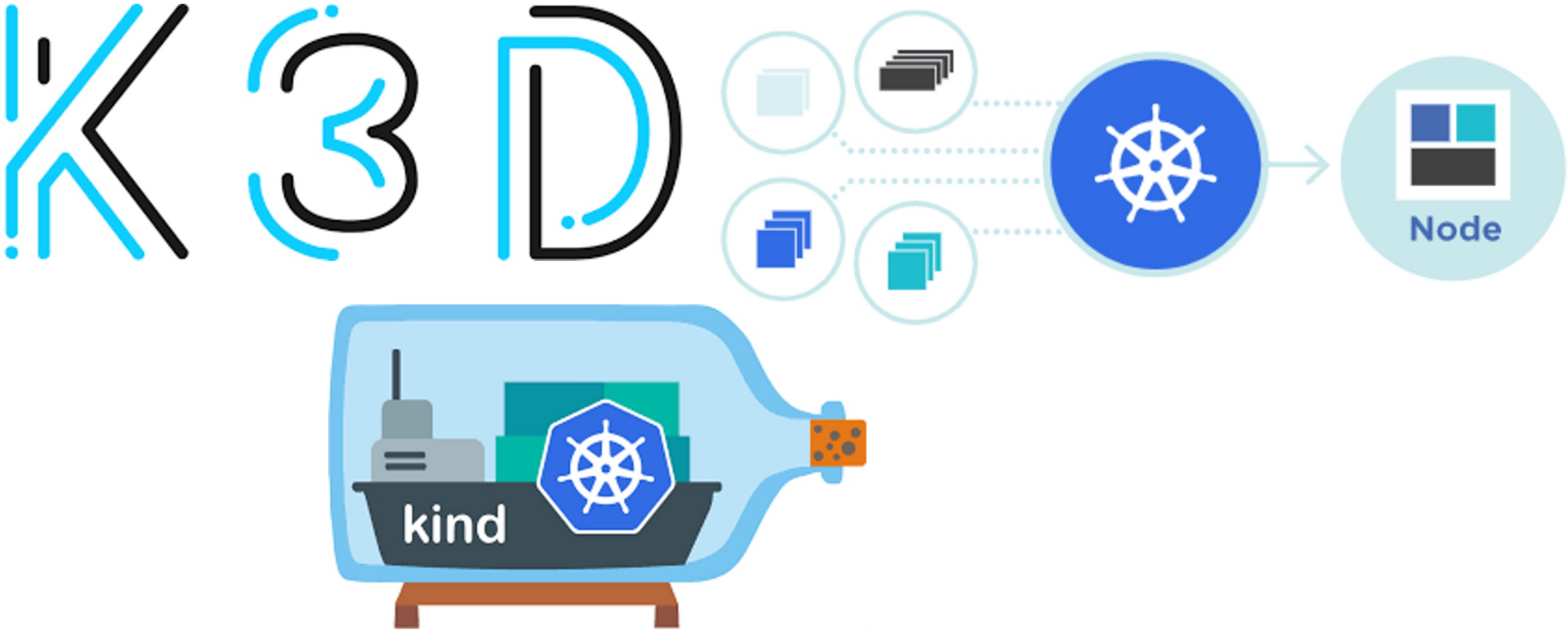
# Managing the Chaos : Annotations

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: webapp-rs
  labels:
    app: webapp
  annotations:
     buildversion: 1.34

spec:
  template:
    metadata:
      name: webapp-pods
      labels:
        app: webapp
        tier: front-end
    spec:
      containers:
        - name: httpd
          image: httpd
          imagePullPolicy: IfNotPresent
          ports:
          - containerPort: 80
```

# Microservices

- Services in a **microservice architecture (MSA)** communicate over a network **using technology-agnostic protocols** such as HTTP.
- **Organized around business capabilities**.
- **Implemented** using different programming languages, databases, and components, **depending on what fits best**.
- **Small in size**, messaging-enabled, bounded by contexts, **decentralized** and deployed **with automated processes.**

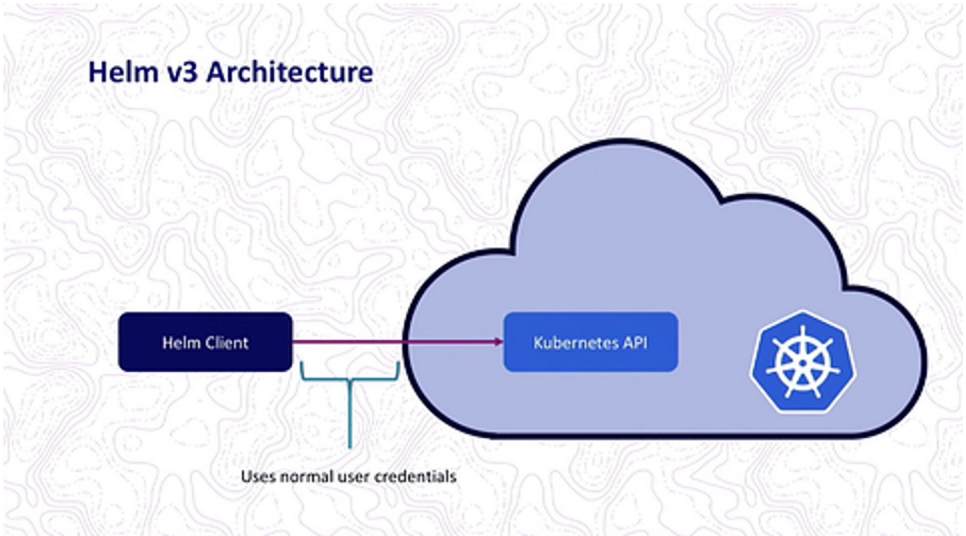
# Local development



# Helm3

Helm helps you manage Kubernetes applications — Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.

Charts are easy to create, version, share, and publish — so start using Helm and stop the copy-and-paste.



# Observability




# POP QUIZ:

## Kubernetes: Framing Our Journey

What is Kubernetes?

A: a set of platform as a service products

B: An open-source container orchestration tool

C: a tool that streamlines installing and managing applications

# POP QUIZ:

## Kubernetes: Framing Our Journey

What is Kubernetes?

A: a set of platform as a service products

B: An open-source container orchestration tool

C: a tool that streamlines installing and managing applications

# POP QUIZ:

## Kubernetes: Framing Our Journey

What is K8s?

A: A term for Kubernetes

B: A Kubernetes Service

C: A term for a local cluster

# POP QUIZ:

## Kubernetes: Framing Our Journey

What is K8s?

A: A term for Kubernetes

B: A Kubernetes Service

C: A term for a local cluster

# POP QUIZ:

## Kubernetes: Framing Our Journey

What is a node in Kubernetes?

A: high-level structures that wrap one or more containers

B: the smallest fundamental unit of computing hardware.

C: a performance monitoring and metrics collection system

# POP QUIZ:

## Kubernetes: Framing Our Journey

What is a node in Kubernetes?

A: high-level structures that wrap one or more containers

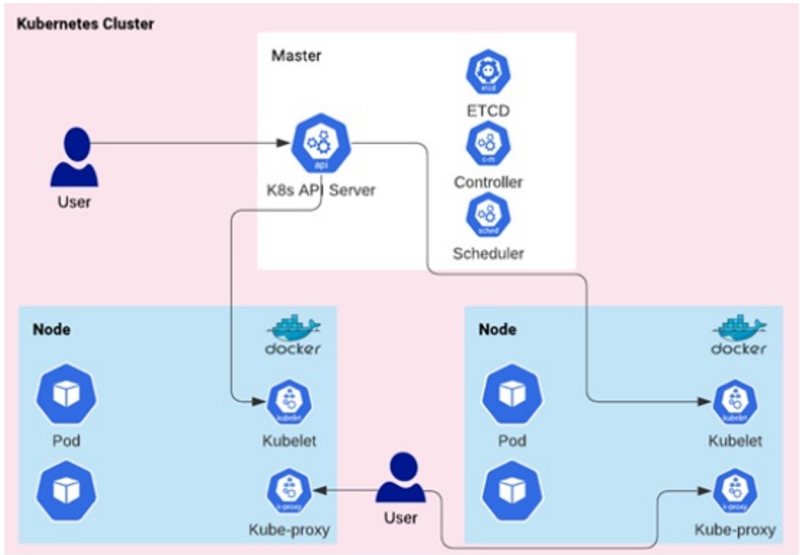
B: the smallest fundamental unit of computing hardware.

C: a performance monitoring and metrics collection system

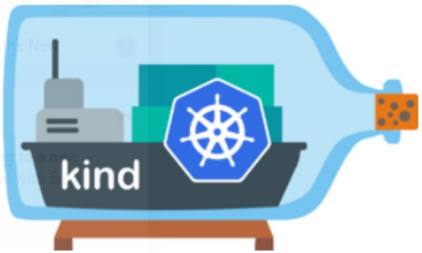
# Local Development, Testing, and Debugging with **KIND**

# What is the objective? Local Dev

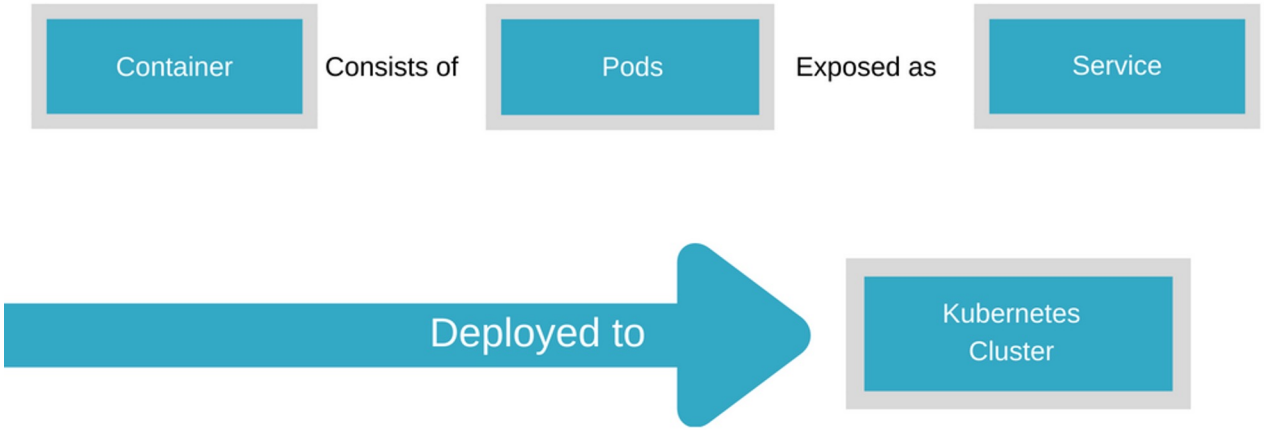
kind is a tool for running local Kubernetes clusters using Docker container "nodes".
kind was primarily designed for testing Kubernetes itself, but may be used for local development or CI



# What about Local Clusters

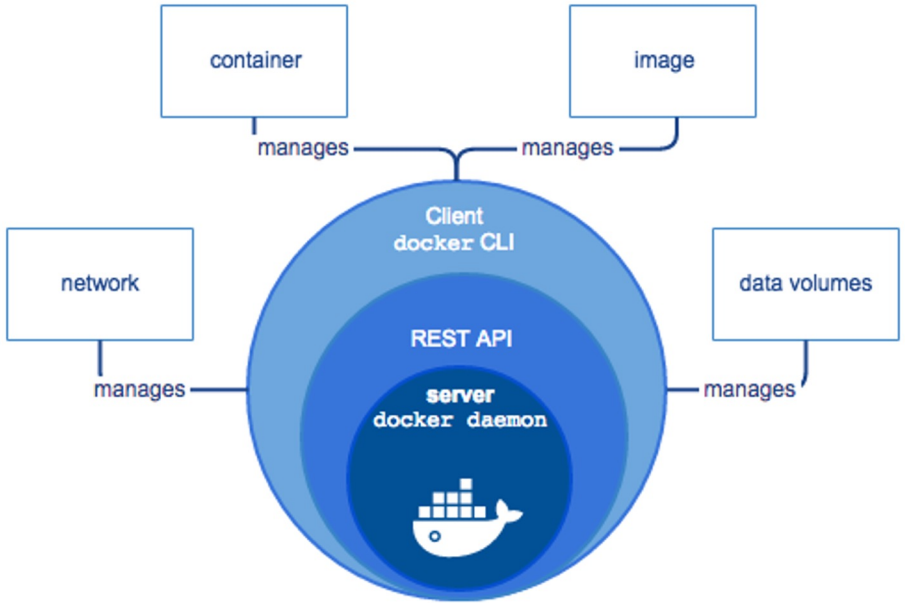



How would a local cluster empower development.  We'll look into a number of ways



# What is Docker?

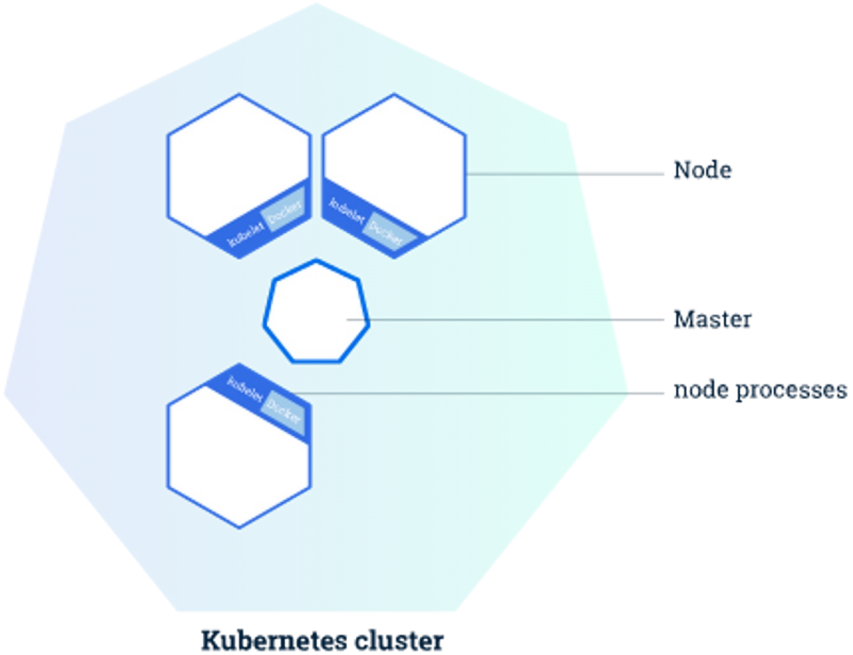
Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.



# Creating a Cluster

Creating a Kubernetes cluster is as simple as `kind create cluster`.

By default, the cluster will be given the name `kind`. Use the `--name` flag to assign the cluster a different context name.



# Developing, Testing and Optimizing Microservice Applications on Kubernetes – Getting Started


Develop a passion for learning.

# Experiment – k3d and PVC


# Working with Helm

# What is Helm

Helm with Kubernetes : Helm helps in combining several Kubernetes factors such as service, deployments, configmaps; many more to the single unit known as Helm Charts. The Cloud Native Computing Foundation maintains the latest version of Helm in association with Microsoft and Google.

# Installing Helm

**Install Helm on Windows using the following command:**

```
> choco install kubernetes-helm
```

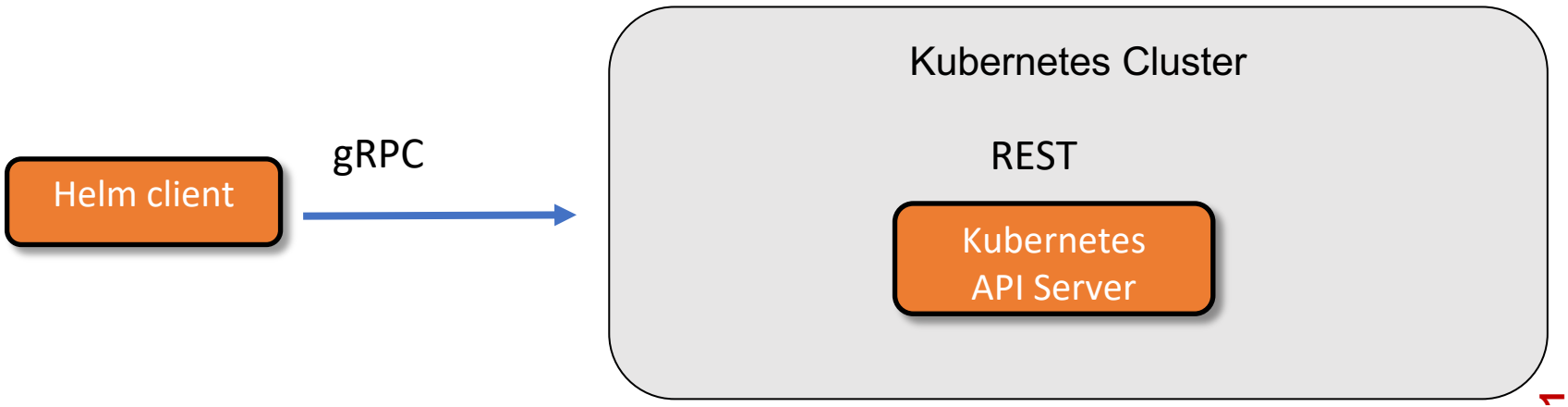
**Install Helm on macOS using the following command:**

```
$ brew install helm
```

# Helm Architecture

Helm client

- command-line
- Interacts with Tiller
- Local chart development



# Helm Features

Charts

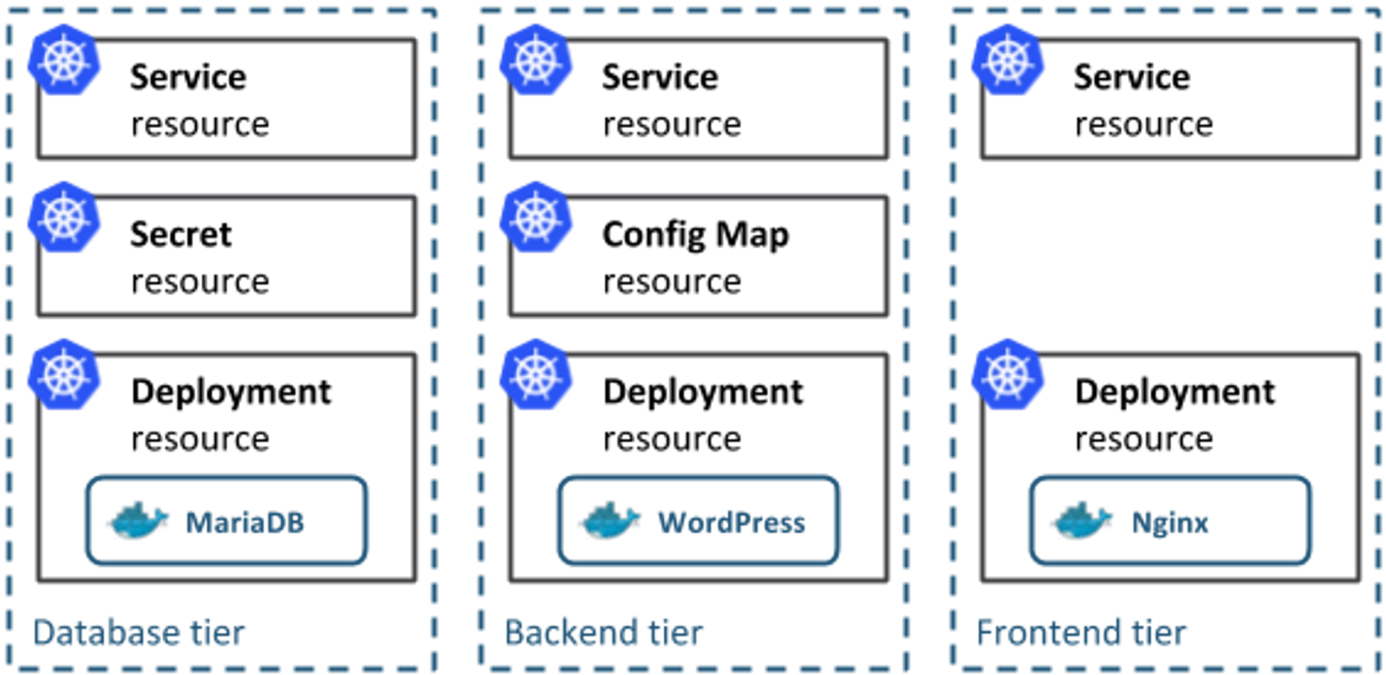
Templates

Dependencies

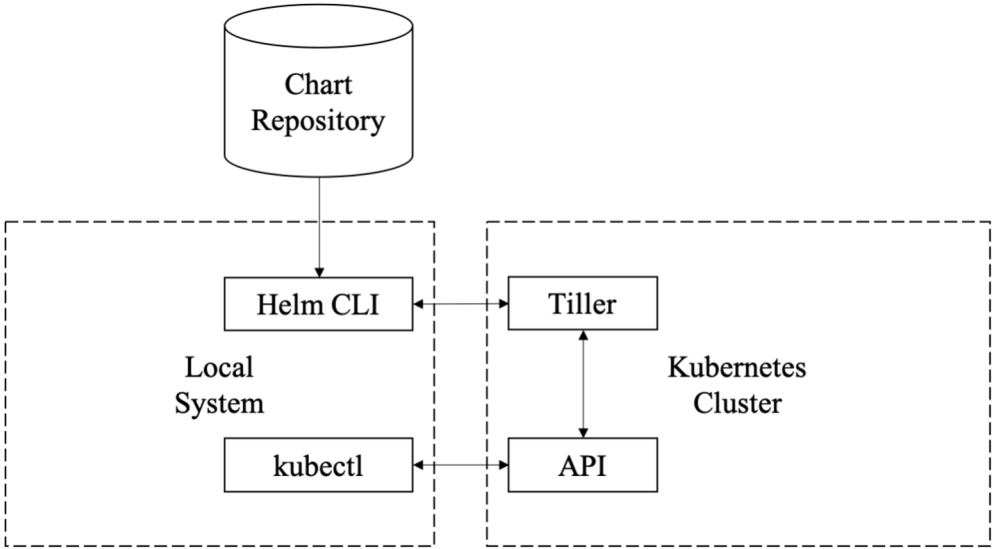
Repositories

# Helm Chart

**Helm** uses a packaging format called **charts**. A **chart** is a collection of files that describe a related set of Kubernetes resources



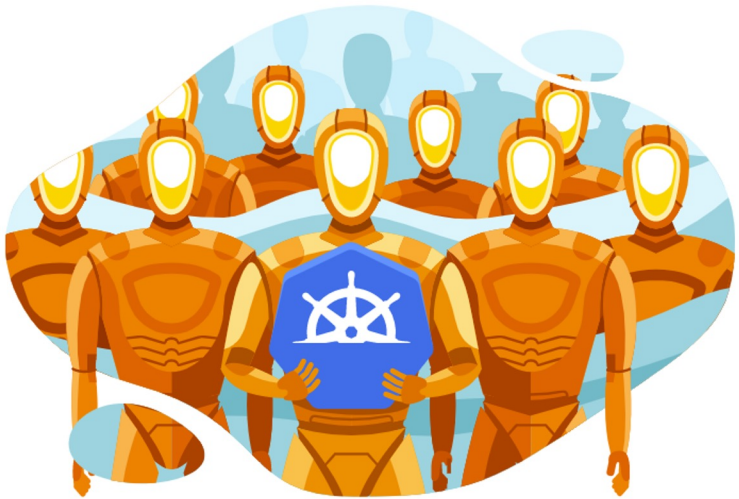
# Helm Charts



# Helm Chart Structure

```
nginx-demo
        Chart.yaml
        README.md
        requirements.yaml
templates
        deployment.yaml
        ingress.yaml
        service.yaml
        NOTES.txt
        tests
                test-connection.yaml

values.yaml
```

Define sub-charts and dependencies



# Helm Chart Structure

Under the top-level directory are the files and directories that comprise the Helm chart. The following table shows each of the possible files and directories.

| File/directory | Definition | Required? |
|---|---|---|
| `Chart.yaml` | A file that contains metadata about the Helm chart. | Yes. |
| `templates/` | A directory that contains Kubernetes resources in YAML format. | Yes, unless dependencies are declared in `Chart.yaml`. |
| `templates/NOTES.txt` | A file that can be generated to provide usage instructions during chart installation. | No. |
| `values.yaml` | A file that contains the chart's default values. | No, but every chart should contain this file as a best practice. |
| `.helmignore` | A file that contains a list of files and directories that should be omitted from the Helm chart's packaging. | No. |

# Helm Commands

| Action | Command |
| --- | --- |
| Install a Release | `helm install` |
| Upgrade a Release revision | `helm upgrade [release] [chart]` |
| Rollback to a Release revision | `helm rollback [release] [revision]` |
| Print Release history | `helm history [release]` |
| Display Release status | `helm status [release]` |
| Show details of a release | `helm get [release]` |
| Uninstall a Release | `helm delete [release]` |
| List Releases | `helm list` |

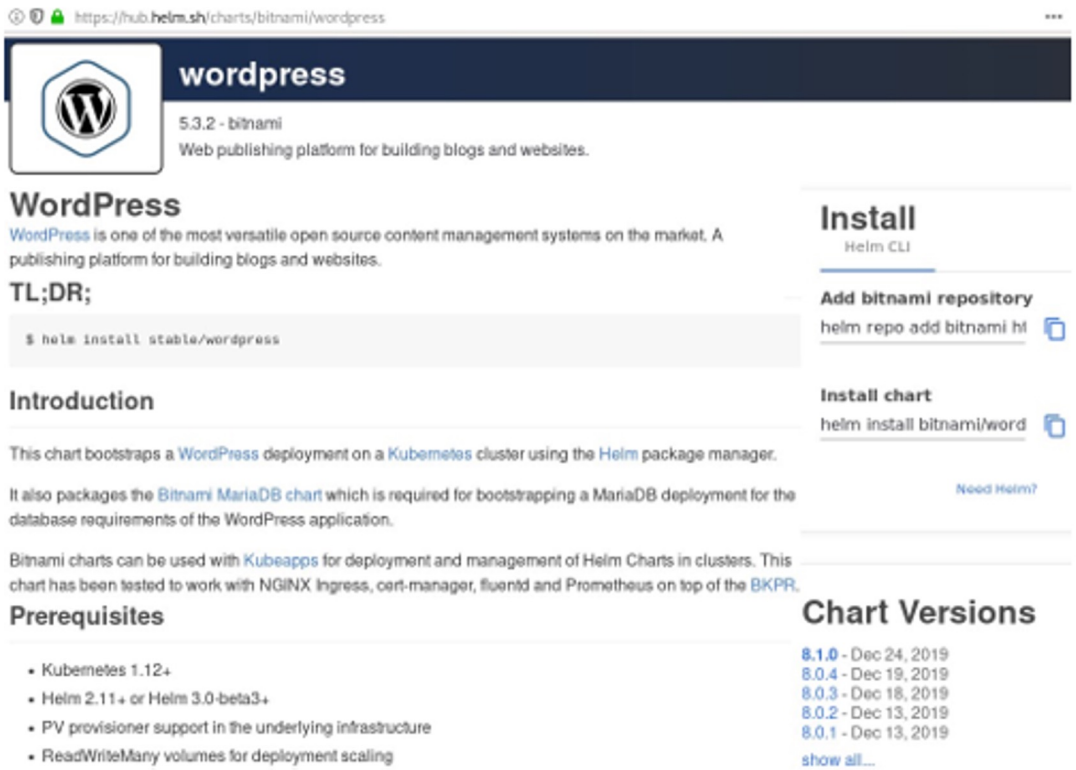
# Finding a WordPress Chart

1. Let's search Helm Hub for any existing WordPress charts. Each chart in Helm Hub has a set of keywords that can be searched against. Execute the following command to locate charts containing the `wordpress` keyword:

Upon running this command, an output similar to the following should be displayed:

```
URL                                               CHART VERSION   APP VERSION     DESCRIPTION
https://hub.helm.sh/charts/bitnami/wordpress      8.1.0           5.3.2           Web publishing
https://hub.helm.sh/charts/presslabs/wordpress-...  v0.6.3          v0.6.3          Presslabs Word
https://hub.helm.sh/charts/presslabs/wordpress-...  v0.7.4          v0.7.4          A Helm chart
```

# Viewing WordPress Chart




The WordPress chart's page from Helm Hub provides many details, including the maintainer of the chart

# Helm Dependencies

This Chart.ynl example declares two dependencies

```
# Chart.yaml
dependencies:
- name: nginx
  version: "1.2.3"
  repository: "https://example.com/charts"
- name: memcached
  version: "3.2.1"
  repository: "https://another.example.com/charts"
```

The 'name' should be the name of a chart, where that name must match the name in that chart's 'Chart.yaml' file.

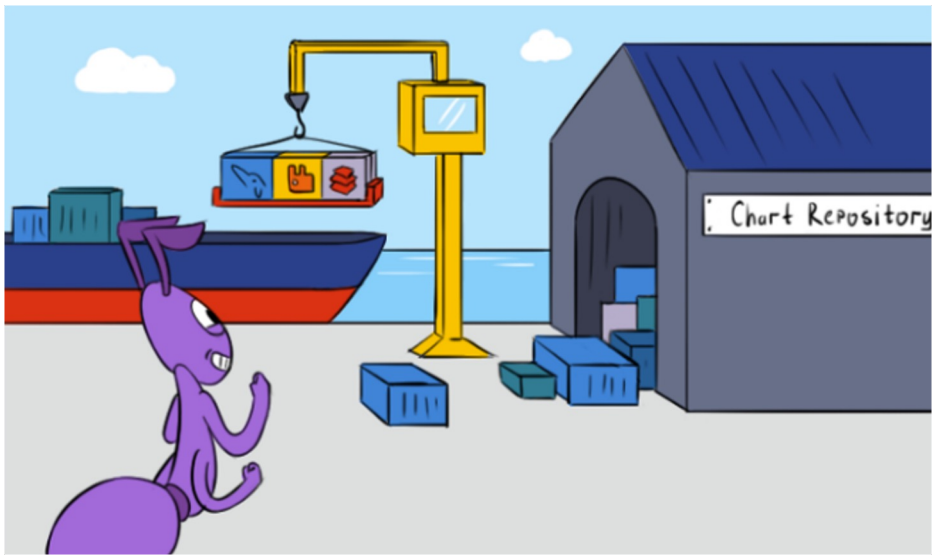
The 'version' field should contain a semantic version or version range.

# Helm Repositories

For example, the layout of the repository

`https://example.com/charts` might look like this:

```
charts/
  |
  |- index.yaml
  |
  |- alpine-0.1.2.tgz
  |

    |- alpine-0.1.2.tgz.prov
```



# Helm Templates

Helm includes many template functions you can take advantage of in templates. They are listed here and broken down by the following categories:

- Cryptographic and Security
- Date
- Dictionaries
- Encoding
- File Path
- UUID
- Kubernetes and Chart
- Logic and Flow Control
- Lists
- Math
- Network
- Reflection
- Regular Expressions
- Semantic Versions
- String
- Type Conversion
- URL

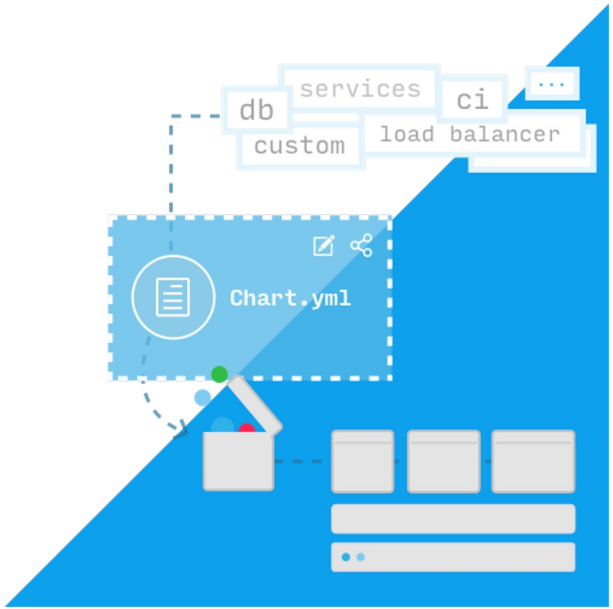
# Linting Helm Charts and Templates

Linting your charts is important to prevent errors in your chart's formatting or the chart's definition file and provide guidance on best practices when working with Helm charts.

The helm lint command has the following syntax:

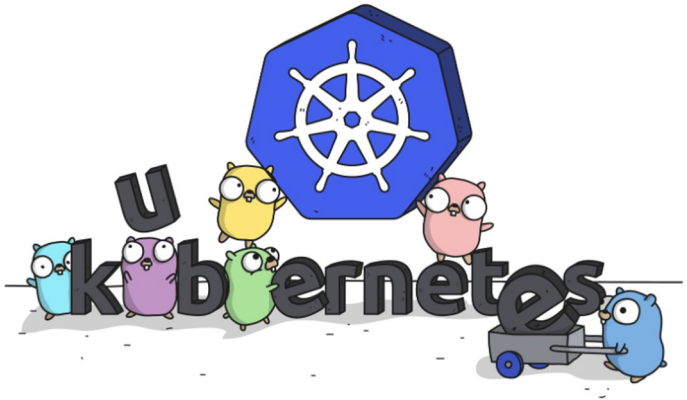
$ helm lint PATH [flags]

The helm lint command is designed to be run against a chart directory to ensure that the chart is valid and properly formatted.



# Debugging Templates

Debugging templates can be tricky because the rendered templates are sent to the Kubernetes API server, which may reject the YAML files for reasons other than formatting.



# Experiment – k3d, Helm, Rancher

# POP QUIZ:

## Kubernetes: Remote Development, Test, and Debug

What is helm?

A: Helm the application package manger that runs on Kubernetes

B: Helm is used for KPT Installation

C: Helm is an open sourced system for running kubernetes

# POP QUIZ:

## Kubernetes: Remote Development, Test, and Debug

What is helm?

A: Helm the application package manager that runs on Kubernetes

B: Helm is used for KPT Installation

C: Helm is an open sourced system for running kubernetes

**POP QUIZ:**

# Kubernetes: Remote Development, Test, and Debug

Why do you need Helm?

A: To help you manage, examine, manipulate, customize, validate, and apply Kubernetes resource configuration files

B: for running local Kubernetes clusters using Docker container “nodes

C: the ability to leverage Kubernetes packages through the click of a button or single CLI command

# POP QUIZ:

## Kubernetes: Remote Development, Test, and Debug

Why do you need Helm?

A: To help you manage, examine, manipulate, customize, validate, and apply Kubernetes resource configuration files

B: for running local Kubernetes clusters using Docker container “nodes

C: the ability to leverage Kubernetes packages through the click of a button or single CLI command

# Experiment – Build Helm Chart

# Controller/ Operator Patterns

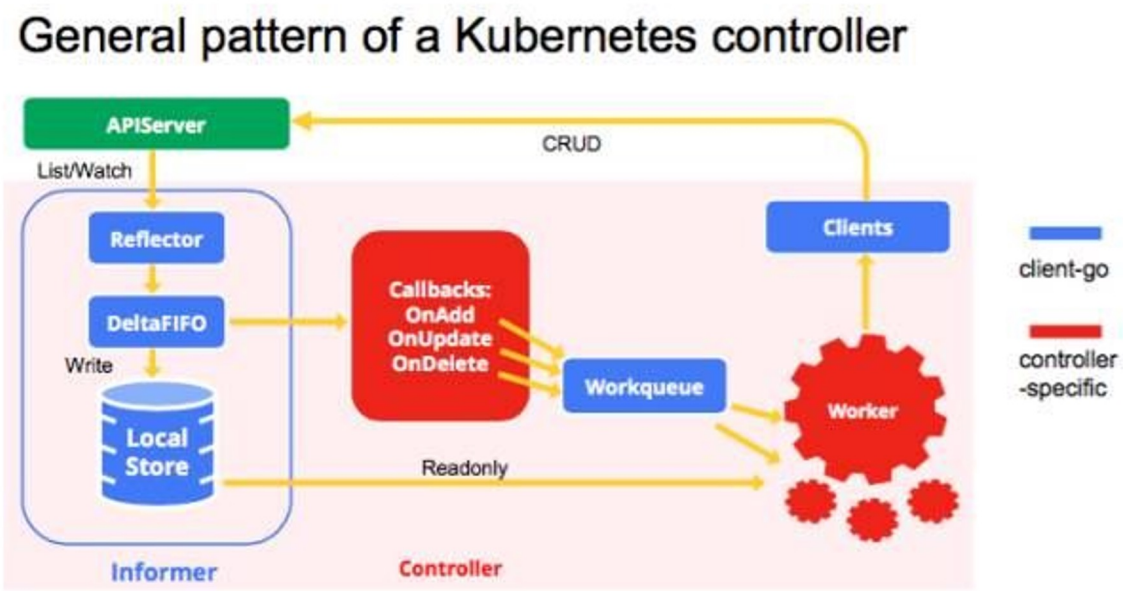
## management, operations, automation

# What is a Controller?

**Controllers take care of routine tasks to ensure the desired state matches the observed state.**

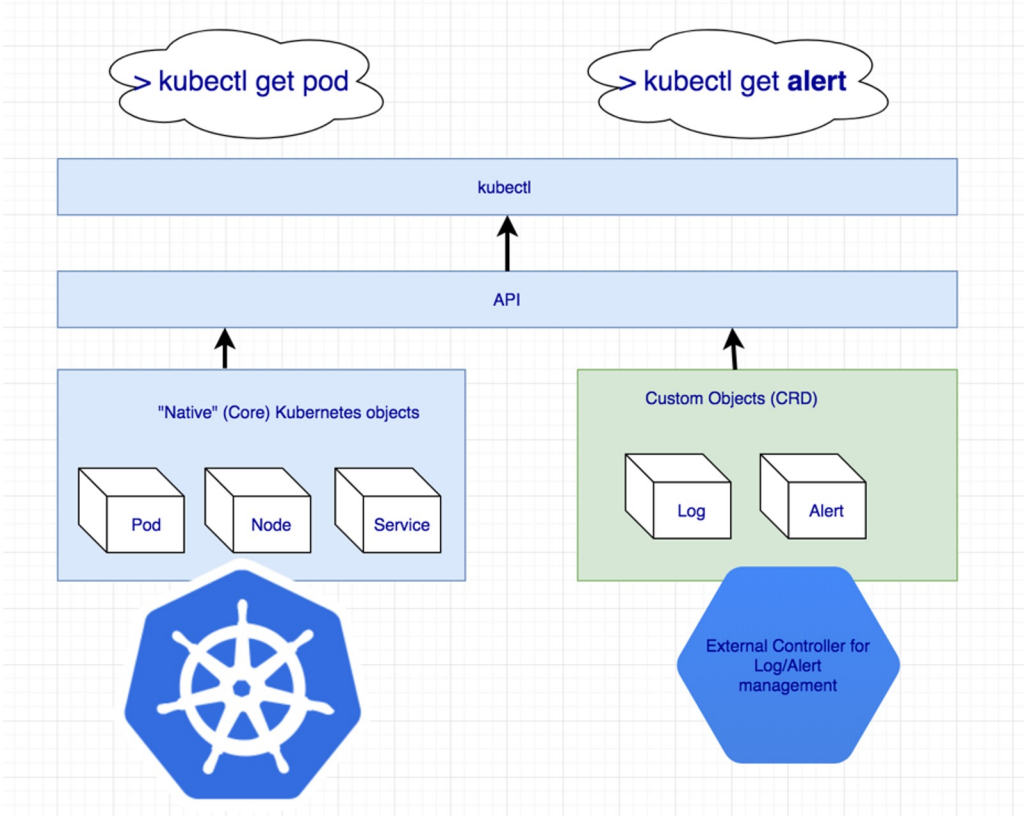
# Controller pattern

Kubernetes runs a group of controllers that take care of routine tasks to ensure the desired state of the cluster matches the observed state.

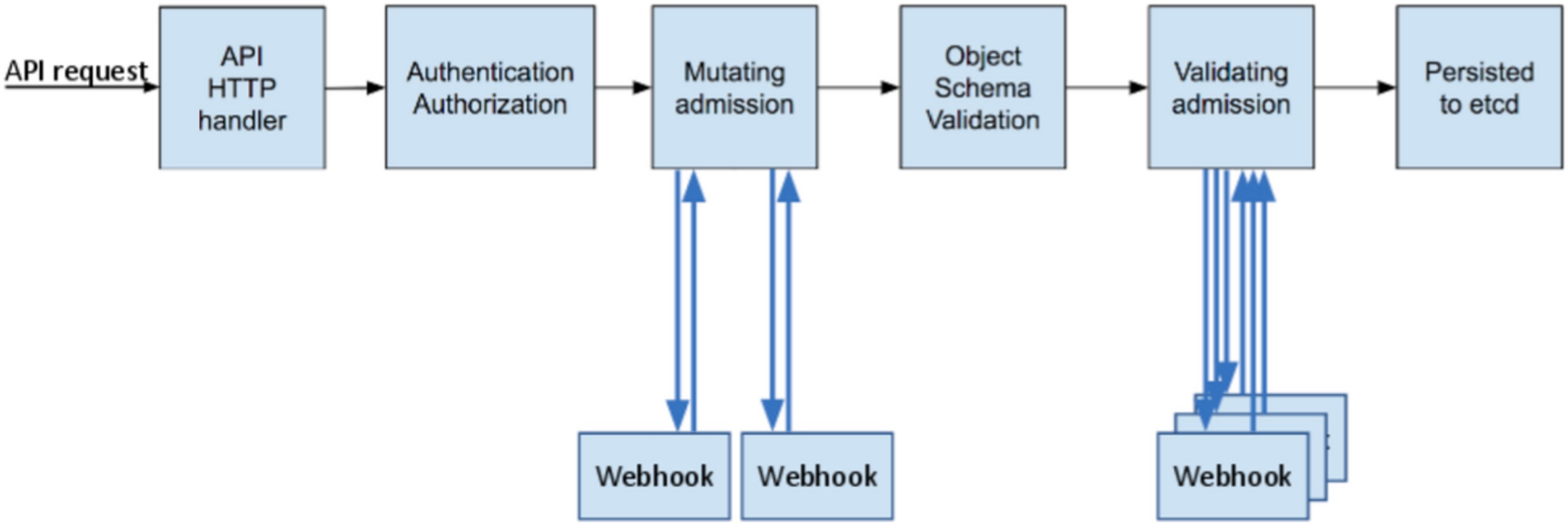


# Controllers in Kubernetes

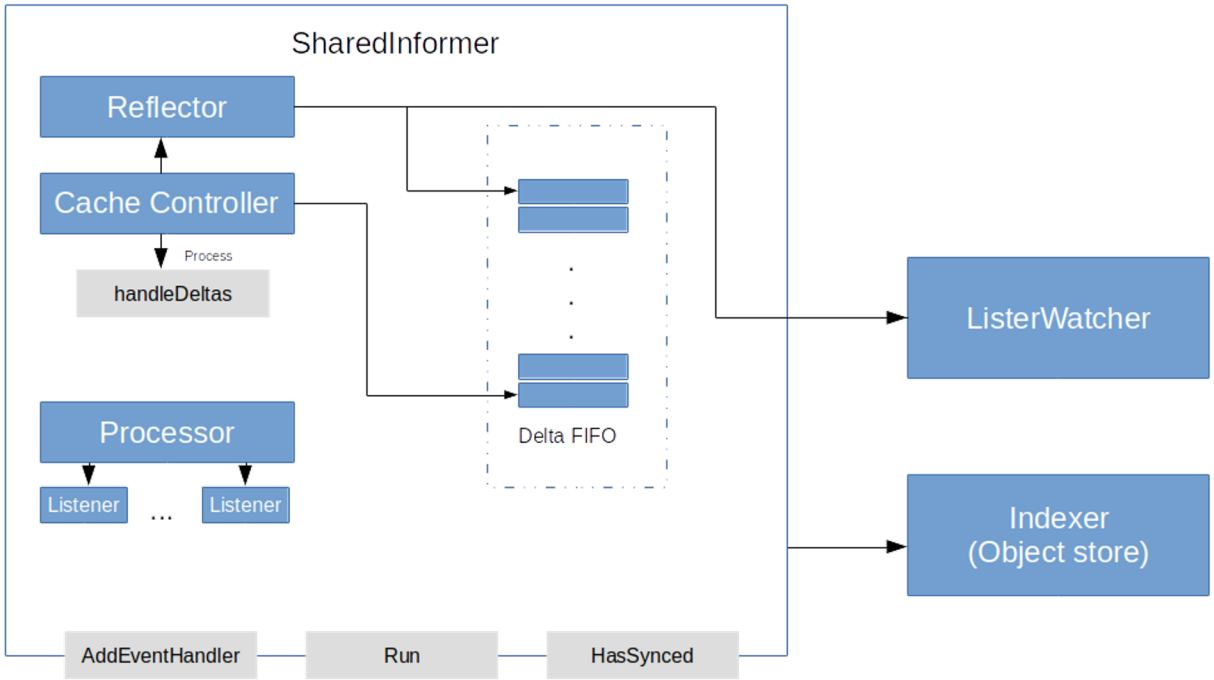
In Kubernetes, a controller is a control loop that watches the shared state of the cluster through the API server and makes changes attempting to move the current state towards the desired state. Examples of controllers that ship with Kubernetes today are the replication controller, endpoints controller, namespace controller, and service accounts controller.



# Example: Admission Controller

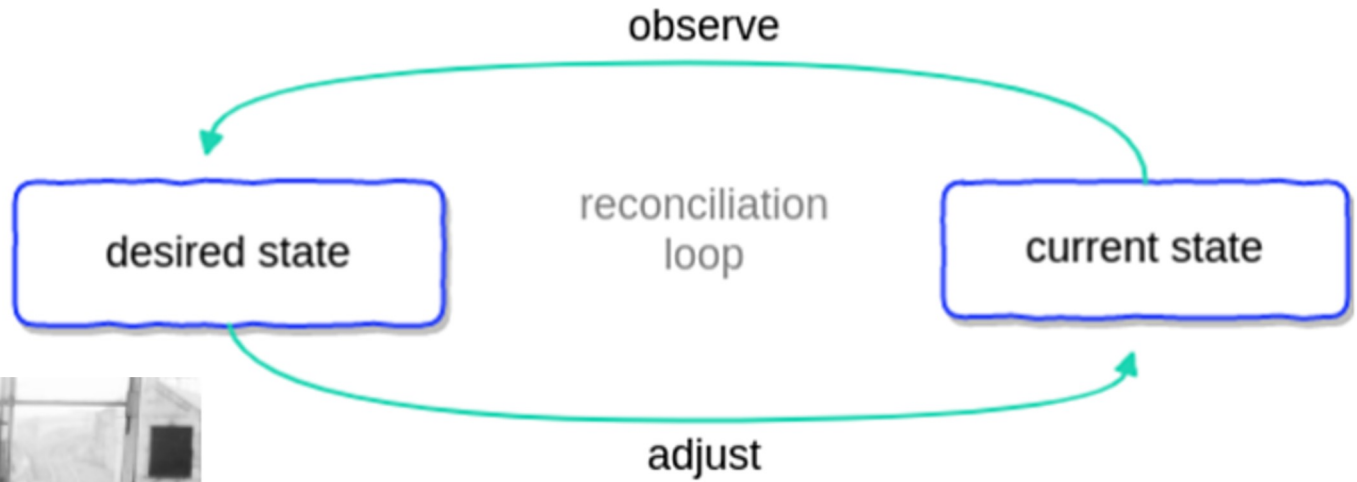


# Controller Components

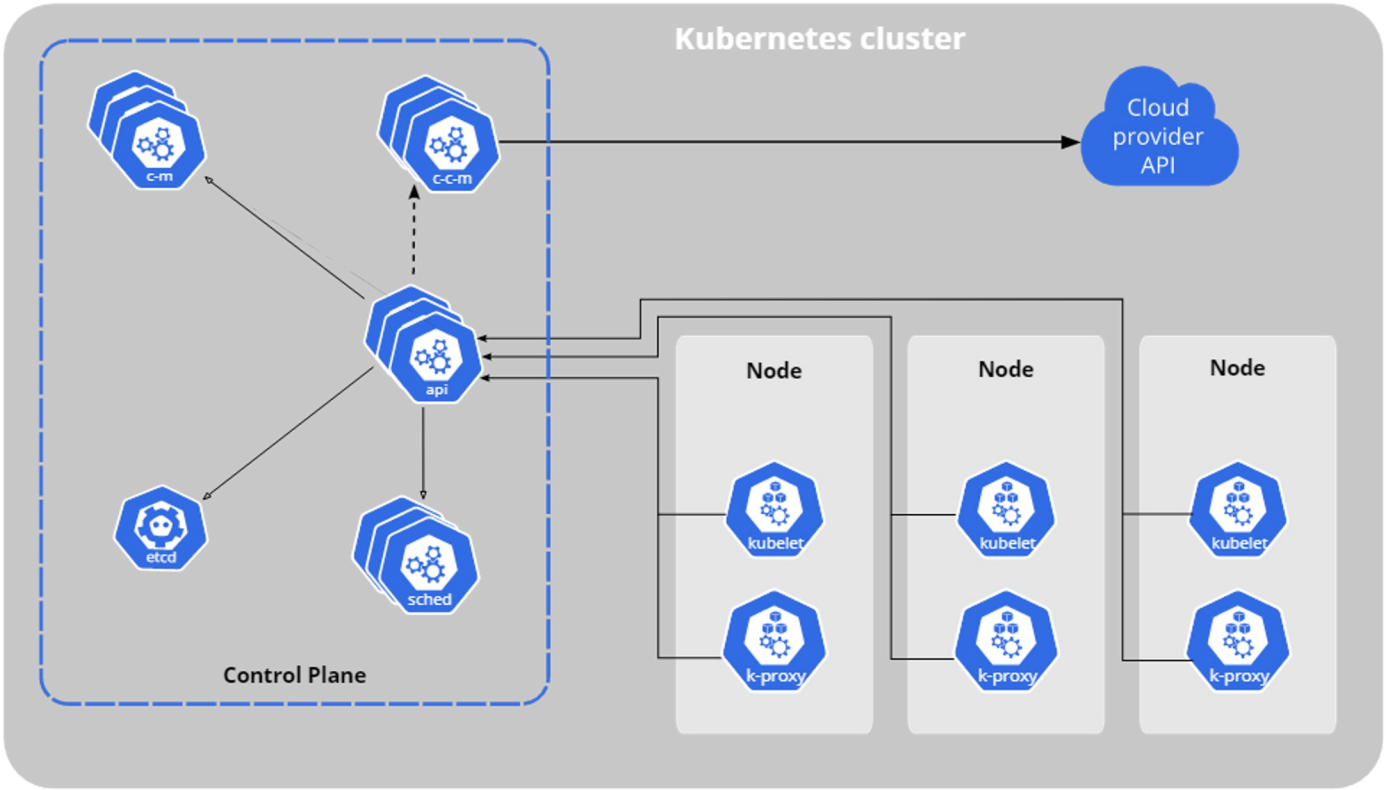



There are two main components of a controller: Informer/SharedInformer and Workqueue. Informer/Shared Informer watches for changes on the current state of Kubernetes objects and sends events to Workqueue where events are then popped up by worker(s) to process.

# Controller loop




# Controller Manager




API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

*kubelet*

*kube-proxy*

Scheduler

Control plane

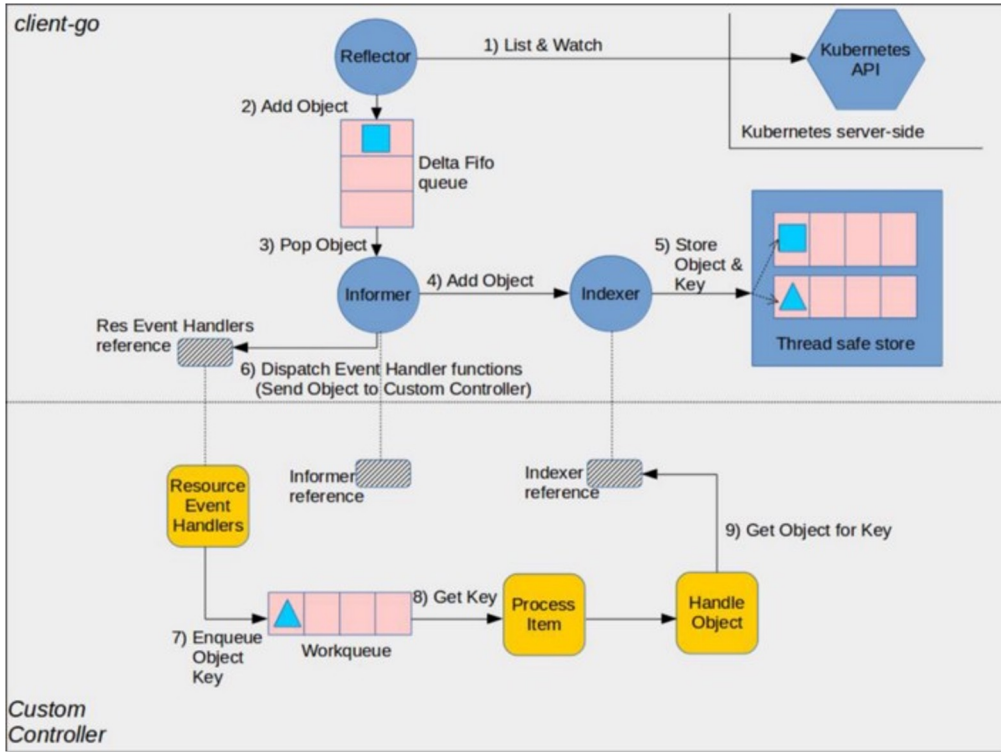
Node

# Node Controller

The controller manager is a daemon that is used for embedding core control loops, garbage collection, and Namespace creation. It enables the running of multiple processes on the master node even though they are compiled to run as a single process.

# Creating Custom Controllers

The picture is divided into two parts— client-go and Custom Controller.

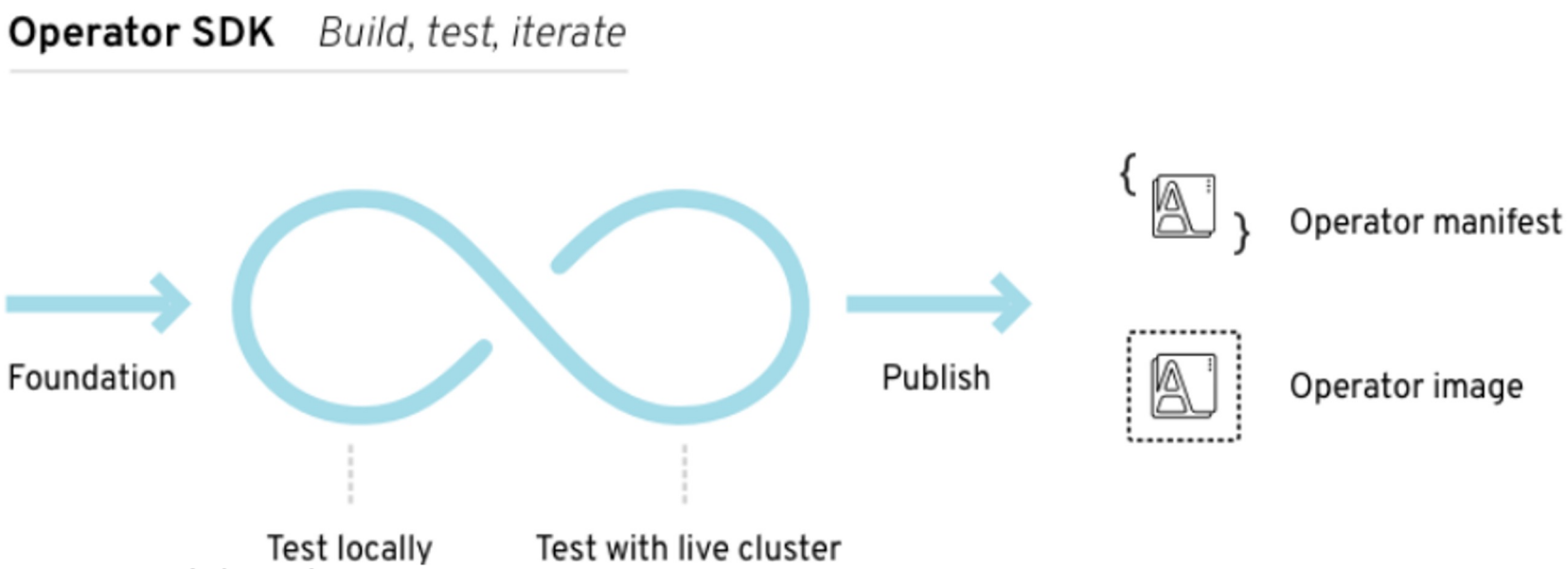


# Building a Custom Controller to Manage Kubernetes Resources

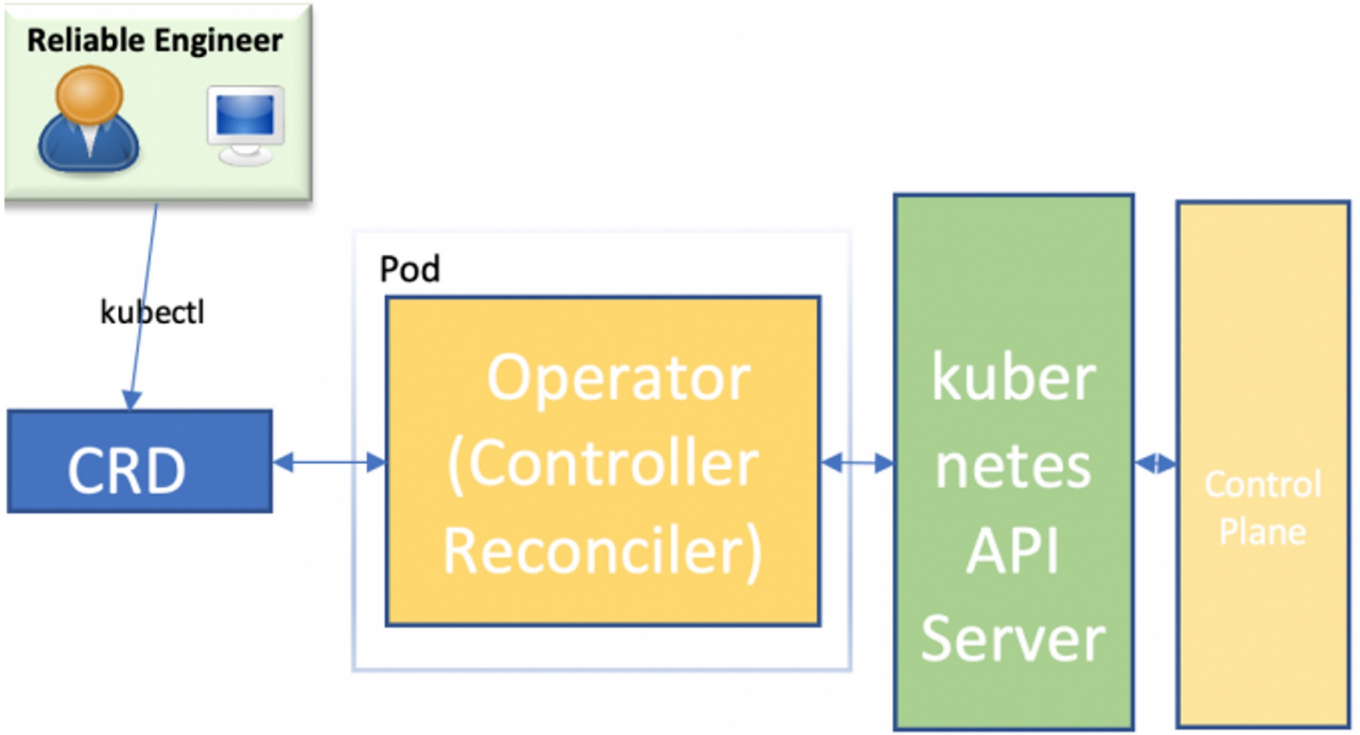
Custom resources simply let you store and retrieve structured data. When you combine a custom resource with a *custom controller*, custom resources provide a true *declarative API*.

A declarative API allows you to *declare* or specify the desired state of your resource and tries to keep the current state of Kubernetes objects in sync with the desired state. The controller interprets the structured data as a record of the user's desired state, and continually maintains this state.

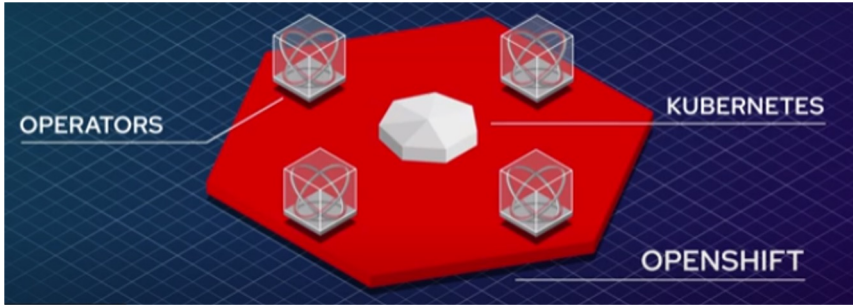
# What is an Operator



# Operator Pattern

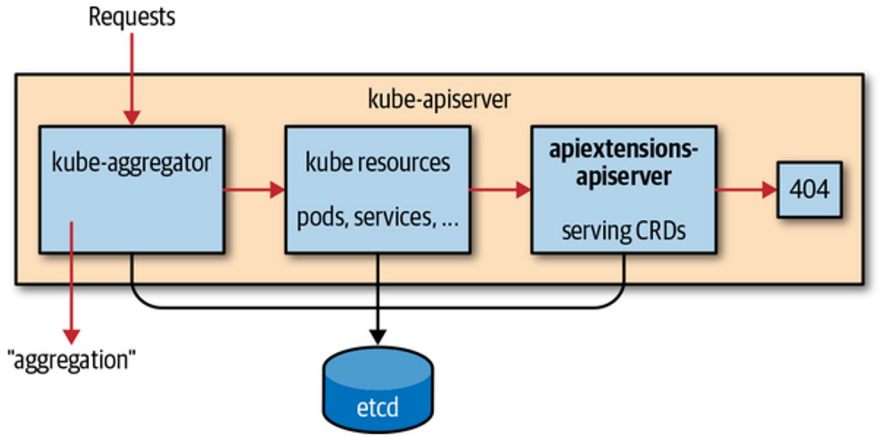


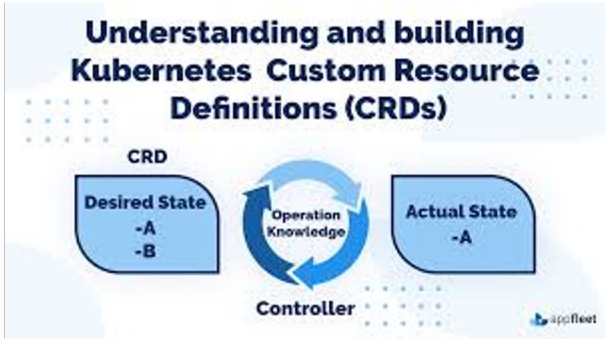
# Developing Operators




| Phase I | Phase II | Phase III | Phase IV | Phase V |
|---|---|---|---|---|
| **Installation** | **Upgrades** | **Lifecycle** | **Insights** | **Auto-pilot** |
| Configuration and one-shot deployment | Patch and minor version upgrades supported | App lifecycle, storage lifecycle (backup, failure recovery) | Metrics, alerts, log processing and workload analysis | Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning... |

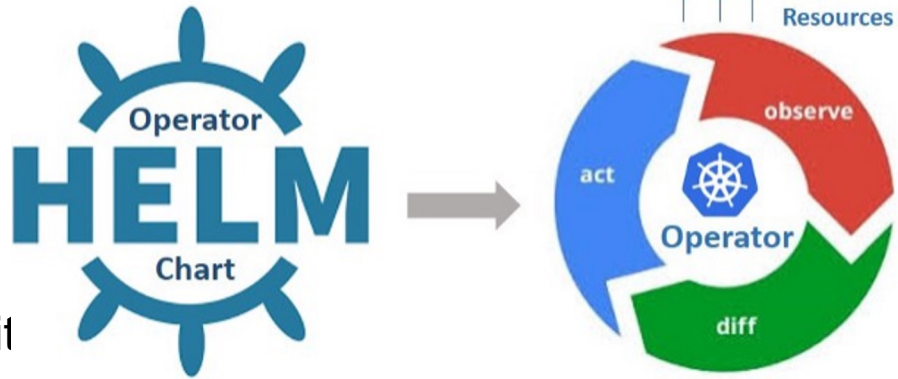
# Building a Custom Resource Definition



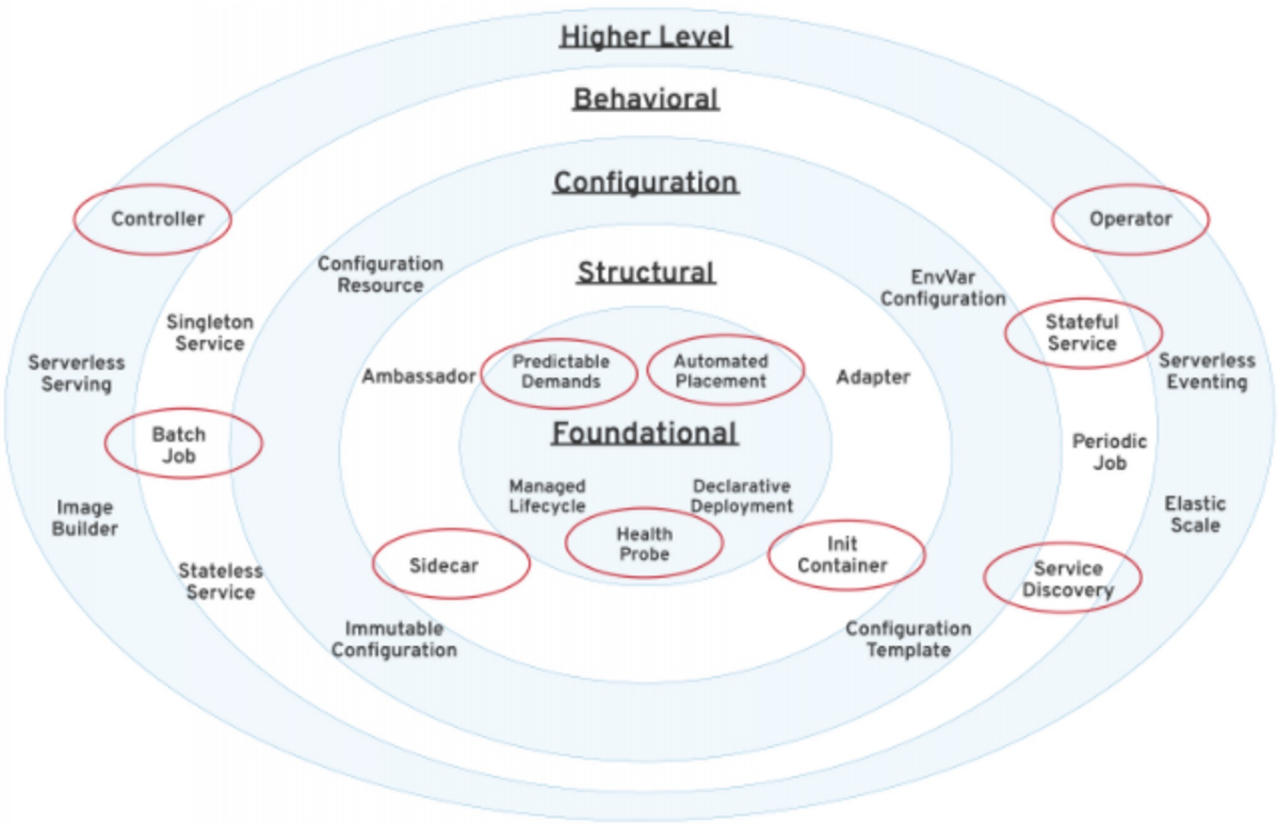



# Operators vs Helm vs Controller

- Helm itself is an operator
- Controllers
  - Operator = controller + CRD
  - Operator = external software
  - Controller = internal
- Only do operators if you can't solve it with Helm



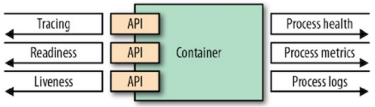
# Application Patterns



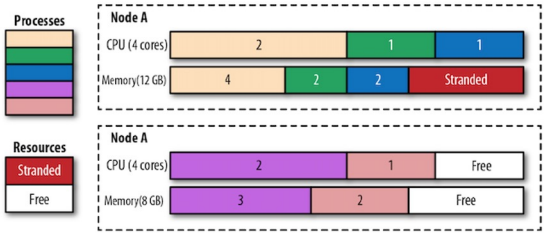
# Foundational Application Patterns

**Foundational Pattern**

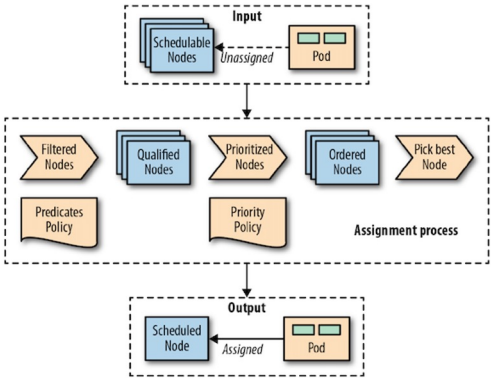
- Predictable demands
- Declarative deployment
- Health probe
- Managed lifecycle
- Automated placement




Health Probe




Predictable Demands

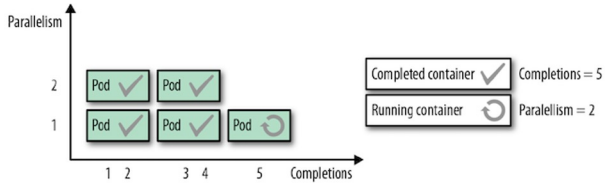



Automated Placement

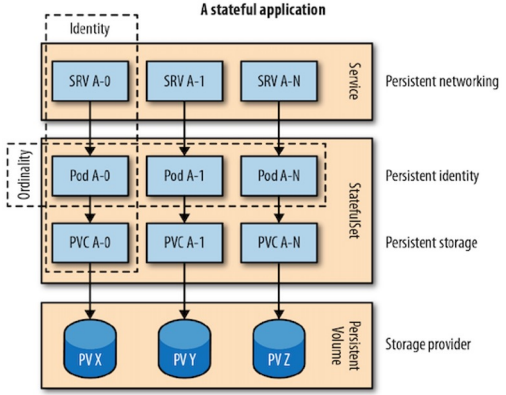
# Behavioral Application Patterns

**Behavioral Pattern**

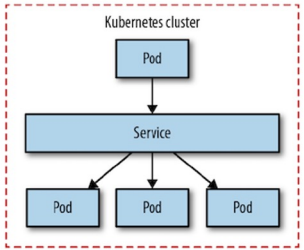
- Batch job
- Periodic job
- Daemon service
- Stateful service
- Self-awareness




Batch Job




Stateful Service

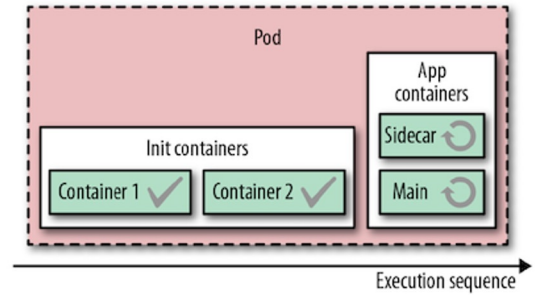



Service Discovery

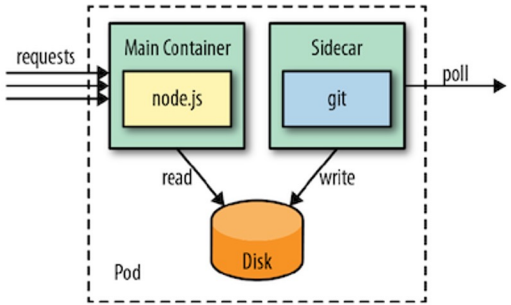
# Structural Application Patterns

**Structural Pattern**

- Init containers
- Sidecar
- Adapter
- Ambassador




Init Container

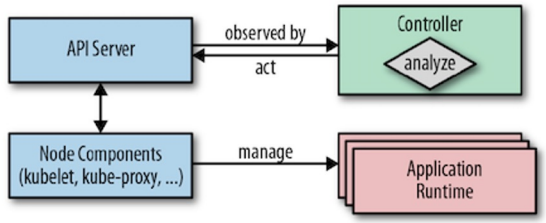



Sidecar

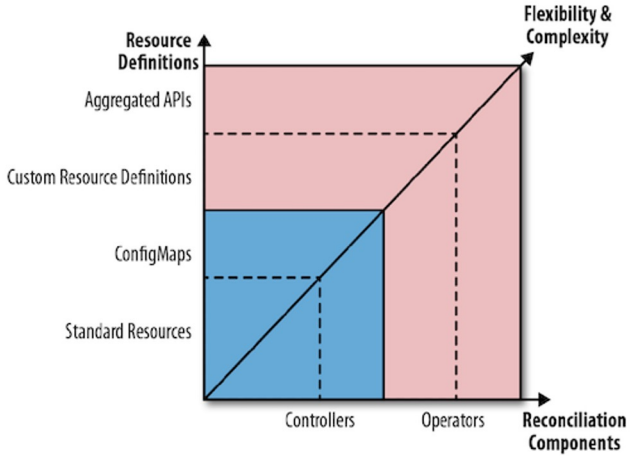
# Higher-level Application Patterns

**Higher-level Patterns**

- Controller Pattern
- Operator Pattern




Controller




Operator

# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

What are custom controllers?

A: patterns that sit on top of foundational patterns and add granularity to concepts for managing various types of container and platform interactions.

B: These patterns are the underlying principles and practices for building container-based cloud-native applications.

C:Patterns that are related to organizing containers within a Kubernetes pod.

# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

What is the Kubernetes Controller Manager?

A:  A set of machine elements that are nodes.

B: A set of pods that run a set of machine elements that are nodes.ly once on a host.

C: A daemon that is used for embedding core control loops, garbage collection, and Namespace creation

# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

What is the Kubernetes Controller Manager?

A:  A set of machine elements that are nodes.

B: A set of pods that run a set of machine elements that are nodes.ly once on a host.

C: A daemon that is used for embedding core control loops, garbage collection, and Namespace creation

# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

Which of the following meets the definition of Foundational Patterns.

A: patterns that sit on top of foundational patterns and add granularity to concepts for managing various types of container and platform interactions.

B: These patterns are the underlying principles and practices for building container-based cloud-native applications.

C:Patterns that are related to organizing containers within a Kubernetes pod.

# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

Which of the following meets the definition of Foundational Patterns.

A: patterns that sit on top of foundational patterns and add granularity to concepts for managing various types of container and platform interactions.

B: These patterns are the underlying principles and practices for building container-based cloud-native applications.

C:Patterns that are related to organizing containers within a Kubernetes pod.

# Operator SDK & Helm

# Skaffold

# Installing Skaffold

**Managed IDE**

- CLOUD CODE
  - VSCode
  - IntelIJ
- GOOGLE CLOUD SHELL
- Cloud Workstation

devfest 2022

# Quickstart - Standalone

## Mac

```
# For macOS on x86_64 (amd64)
curl -Lo skaffold https://storage.googleapis.com/skaffold/releases/latest/skaffold-darwin-amd64 && \
sudo install skaffold /usr/local/bin/
# For macOS on ARMv8 (arm64)
curl -Lo skaffold https://storage.googleapis.com/skaffold/releases/latest/skaffold-darwin-arm64 && \
sudo install skaffold /usr/local/bin/
```

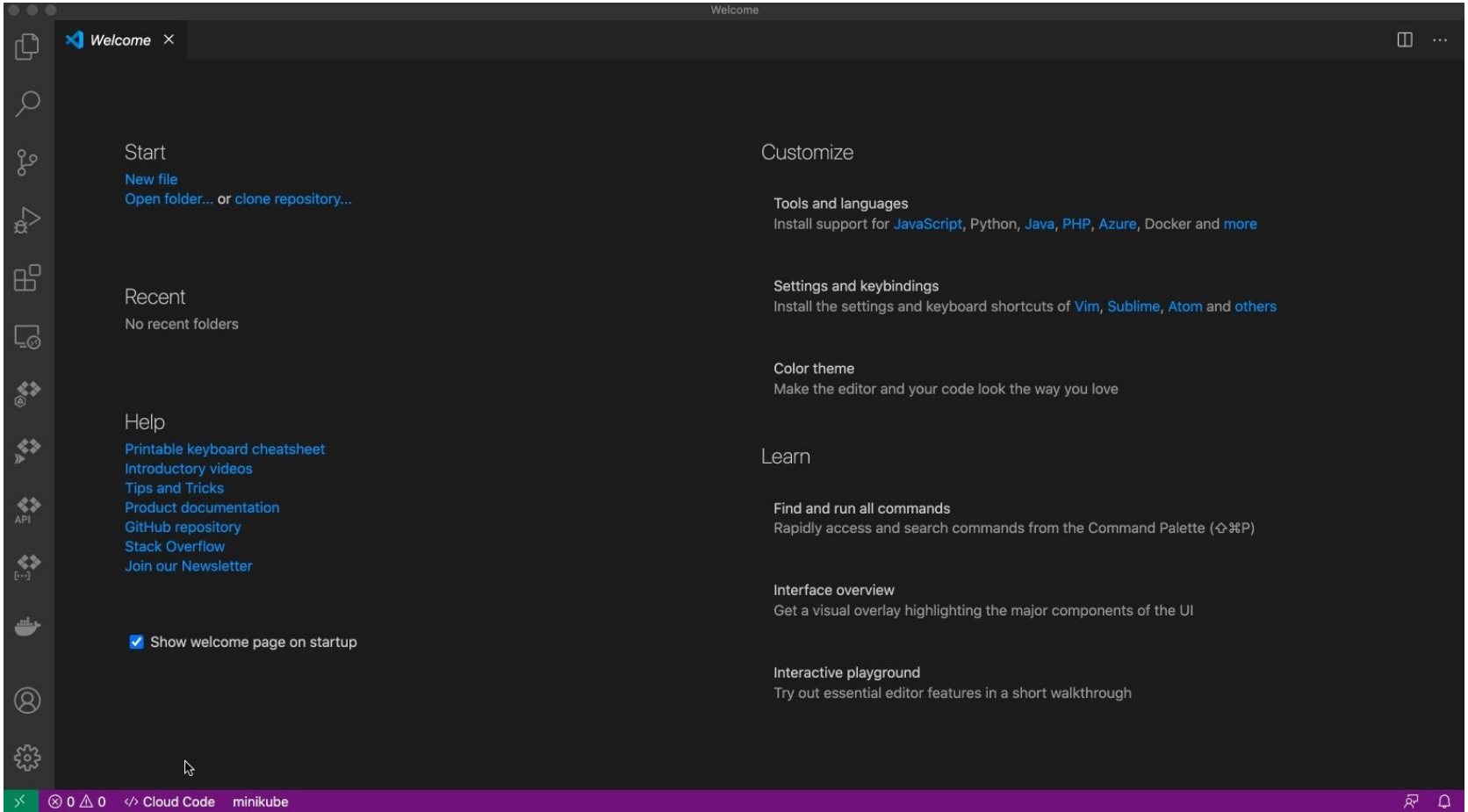
## gcloud

```
gcloud components install skaffold
```

## docker

```
docker run gcr.io/k8s-skaffold/skaffold:latest skaffold <command>


steps:
  - name: gcr.io/k8s-skaffold/skaffold:v1.21.0
    entrypoint: bash
    args:
      - -exc
      - |
        gcloud container clusters get-credentials $_CLOUDSDK_CONTAINER_CLUSTER --zone $_CLOUDSDK_COMPUTE_Z
```

# Quickstart - VSCode / IntellJ



# Guides

Getting Started With Your Project

skaffold dev

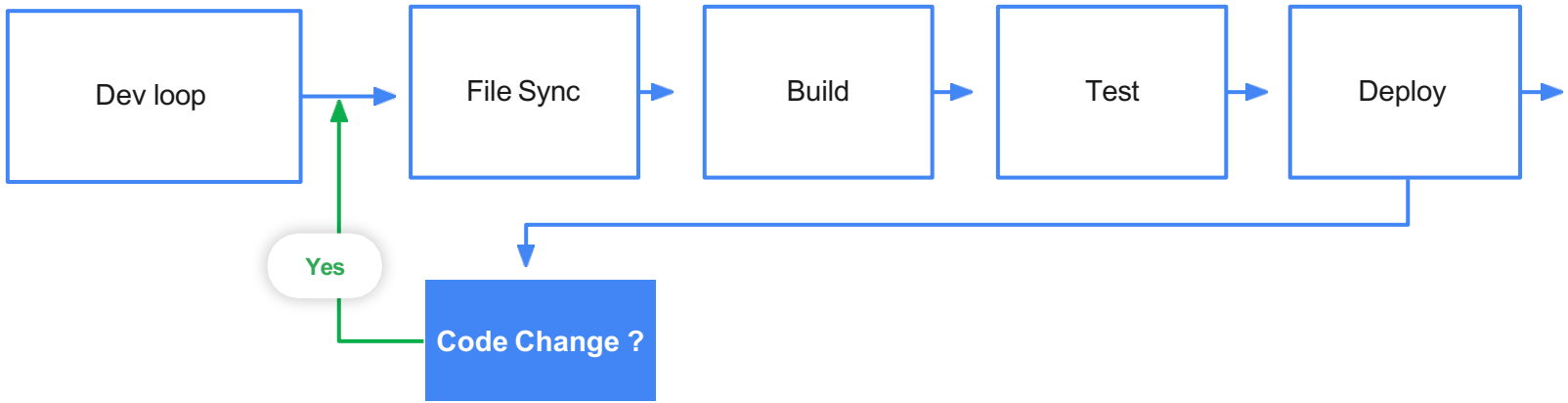
Debugging With Skaffold

Continuous Delivery

Managing ARM workloads [NEW]

# Guides - skaffold dev

**skaffold dev** enables continuous local development on an application. While in dev mode, Skaffold will watch an application's source files, and when it detects changes, will rebuild your images (or sync files to your running containers), push any new images, test built images, and redeploy the application to your cluster.



# Guides - skaffold debug

Skaffold lets you set breakpoints and step through your application, even when deployed to remote Kubernetes clusters, as if the code were running locally on your machine

- Go 1.13+ (runtime ID: **go**) using Delve
- NodeJS (runtime ID: **nodejs**) using the NodeJS Inspector (Chrome DevTools)
- Java and JVM languages (runtime ID: **jvm**) using JDWP
- Python 3.5+ (runtime ID: **python**) using **debugpy** (Debug Adapter Protocol) or **pydevd**
- .NET Core (runtime ID: netcore) using **vsdbg** (only for VS Code)

# Guides - Continuous Delivery

Skaffold provides several features and sub-command "**building blocks**" that make it very useful for integrating with (or creating entirely new) CI/CD pipelines.

- **skaffold build** - build, tag and push artifacts to a registry
- **skaffold deploy** - deploy built artifacts to a cluster
- **skaffold render** - export the transformed Kubernetes manifests
- **skaffold apply** - send hydrated Kubernetes manifests to the API server to create resources on the target cluster

```
$ skaffold build -f skaffold-v2.yaml --default-repo gcr.io/gcp-10-mins --file-output build.json
$ skaffold deploy --build-artifacts=./build.json

{"builds":[{"imageName":"square","tag":"gcr.io/gcp-10-mins/square:latest@sha256:665db6a58565ee192093df04470
ea86705d8b33b69c1e7f6d7a6ea07e18491f6"}]}
```

# Guides - Continuous Delivery Cont.

beta

Skaffold allows separating the generation of **fully-hydrated Kubernetes manifests** from the actual deployment of those manifests, using the **skaffold render** and **skaffold apply** commands.

```
$ skaffold render -f skaffold-v2.yaml --default-repo gcr.io/gcp-10-mins   > render.yaml

$ skaffold render -f skaffold-v2.yaml apply render.yaml
```

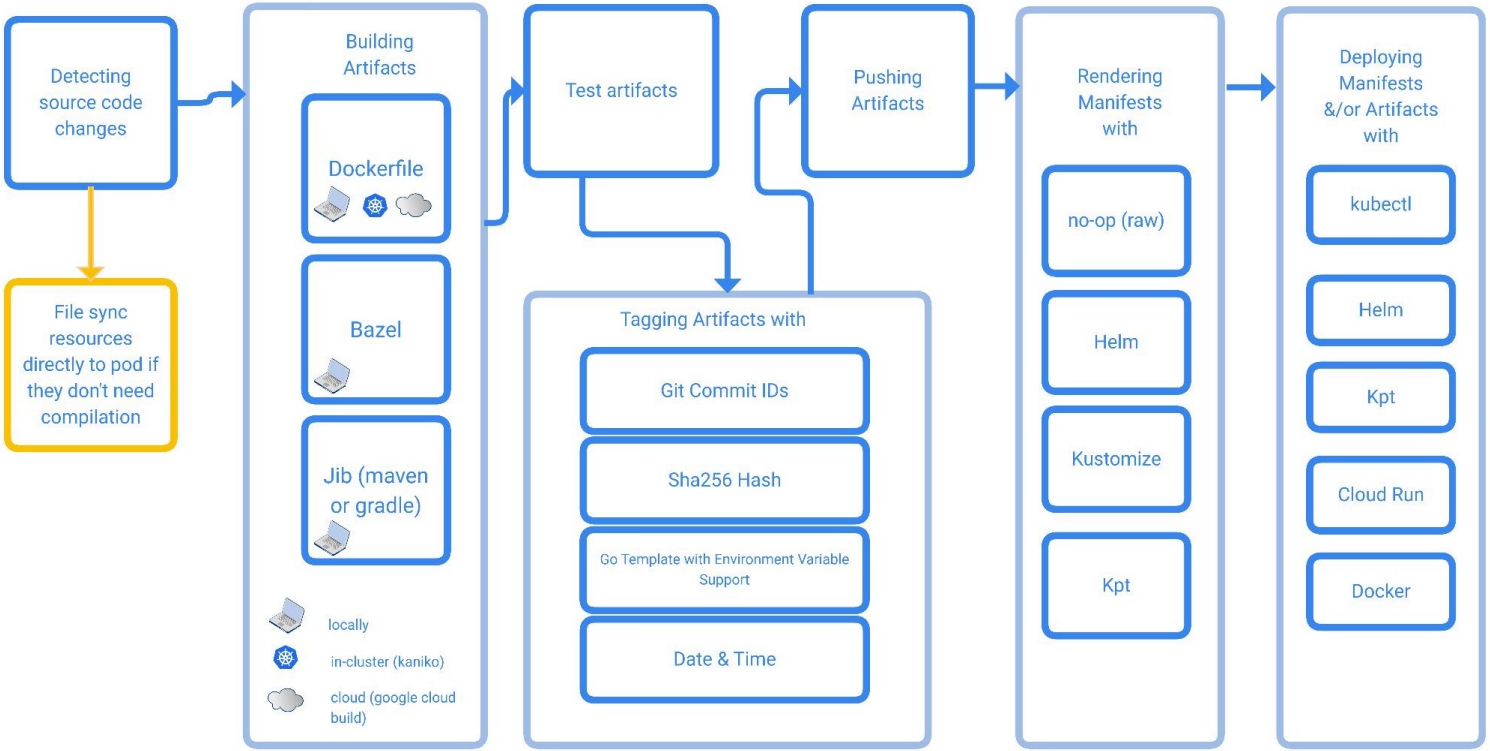
New

# Guides - Managing ARM workloads

Skaffold will check the active Kubernetes cluster node architecture and provide that as an argument to the respective image builder. If the cluster has multiple architecture nodes, then Skaffold will also create appropriate Kubernetes **affinity** rules so that the Kubernetes Pods with these images are assigned to matching architecture nodes.

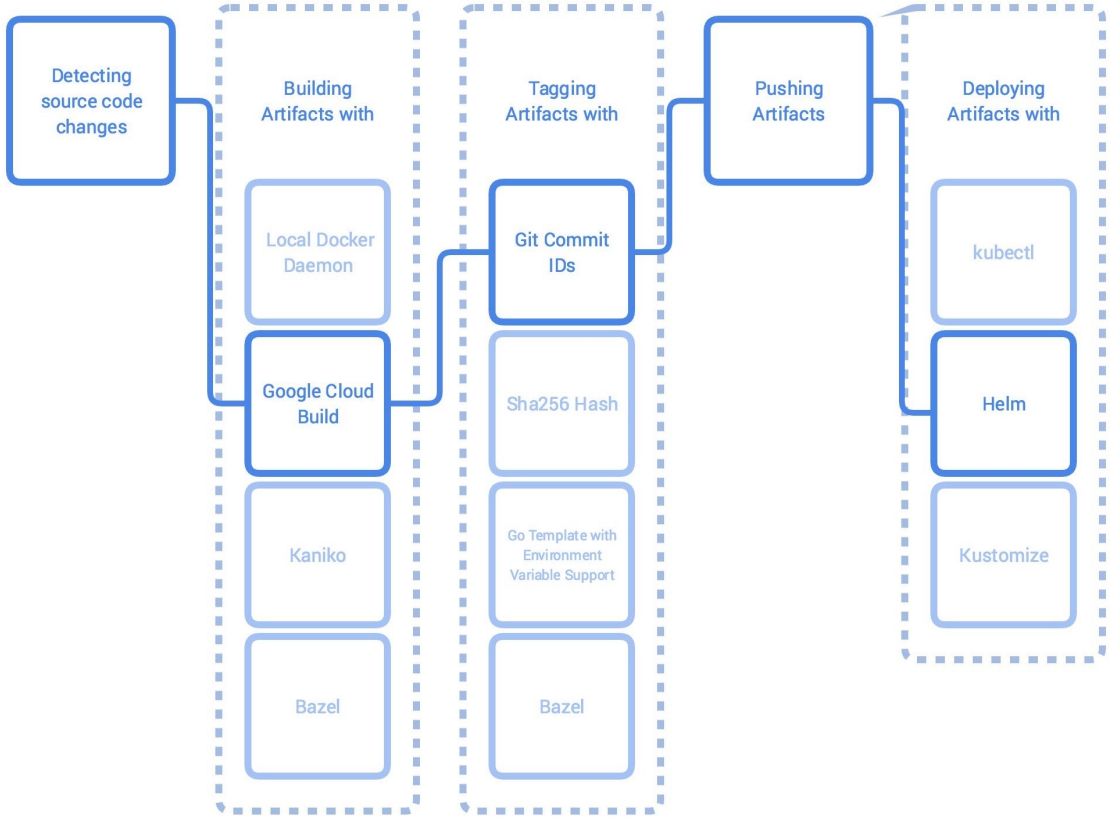
```
$ skaffold run -f skaffold-v2.yaml --default-repo gcr.io/gcp-10-mins --cache-artifacts=false
Generating tags...
 - square -> gcr.io/gcp-10-mins/square:latest
Starting build...
Building [square]...
Target platforms: [linux/amd64]
v1: Pulling from buildpacks/builder
Digest: sha256:de4d669b82419307072df7b35c3c9a81aca35e308be2bac5ec4ca7dbcba31f24
Status: Image is up to date for gcr.io/buildpacks/builder:v1
```

# Architecture



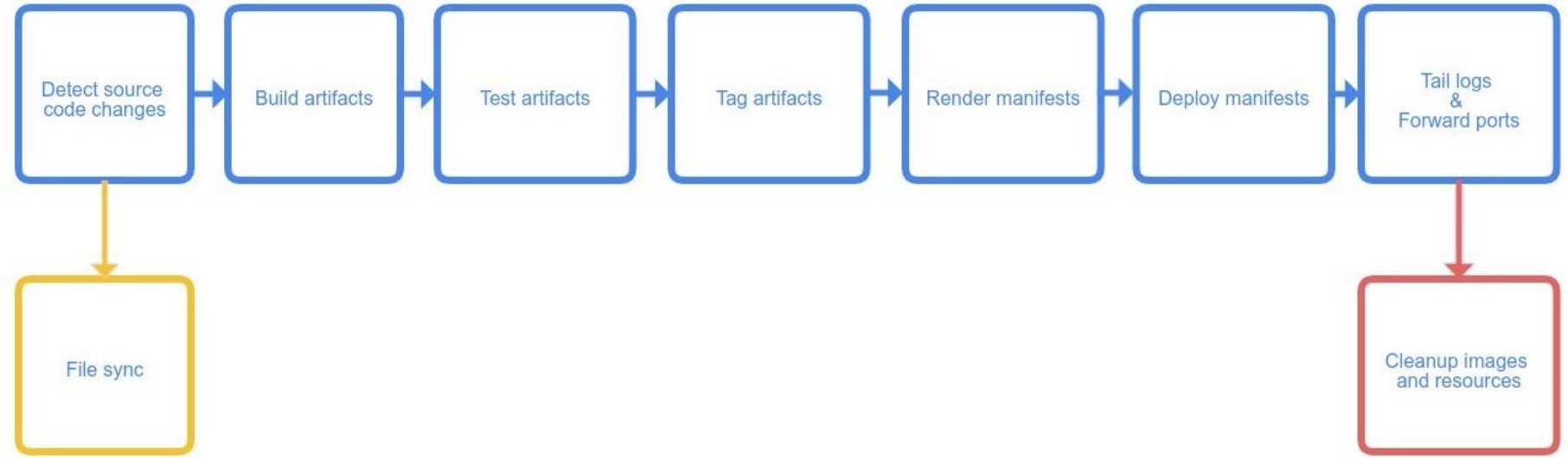

Architecture and Design | Skaffold - https://skaffold.dev/docs/design/

# Architecture - Cloud Build / Helm




Architecture and Design | Skaffold - https://skaffold.dev/docs/design/

# Skaffold Pipeline Stages




Skaffold Pipeline Stages | Skaffold - https://skaffold.dev/docs/pipeline-stages/

# Init

- **skaffold init** helps you get started using Skaffold by running you through a wizard and generating the required **skaffold.yaml** file in the root of your project directory.

- The generated **skaffold.yaml** defines your build and deploy config.

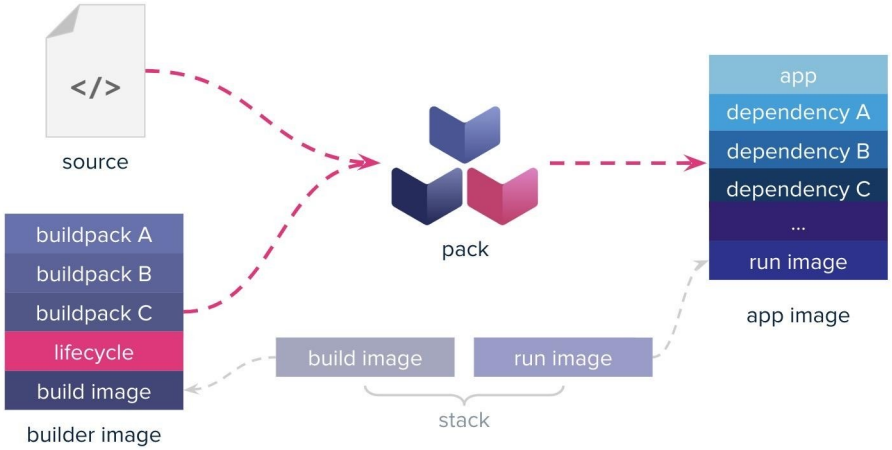
```yaml
apiVersion: skaffold/v4beta1
kind: Config
metadata:
  name: '-'
build:
  artifacts:
  - image: gcr.io/gcp-10-mins/add
    buildpacks:
      builder: gcr.io/buildpacks/builder:v1
manifests:
  helm:
    flags:
      upgrade:
        - --install
    releases:
    - name: square
      chartPath: deployments/square
      setValues:
        square.image: square
      setValueTemplates:
        square.env.QS_LOG_LEVEL:   "{{.QS_LOG_LEVEL}}"
deploy:
  helm:
    {}
```

# Build

The **build** section in the Skaffold configuration file, **skaffold.yaml**, controls how artifacts are built. To use a specific tool for building artifacts, add the value representing the tool and options for using that tool to the **build** section.

| | Local Build | In Cluster Build | Remote on Google Cloud Build |
|---|---|---|---|
| Dockerfile | Yes | Yes | Yes |
| Jib Maven and Gradle | Yes | - | Yes |
| Cloud Native Buildpacks | Yes | - | Yes |
| Bazel | Yes | - | - |
| ko | Yes | - | Yes |
| Custom Script | Yes | Yes | - |

# Build - Cloud Native Buildpacks




```yaml
apiVersion: skaffold/v4beta1
kind: Config
metadata:
  name: '-'
build:
  artifacts:
  - image: square
    buildpacks:
      builder: gcr.io/buildpacks/builder:v1
      env:
        - GOOGLE_BUILDABLE=cmd/square/main.go
        - GOOGLE_RUNTIME_VERSION=1.14
      dependencies:
        paths:
          - cmd/square/main.go
          - internal/app/square/**
          - internal/pkg/**
tagPolicy:
  sha256: {}
local:
  push: true
```

# Build - Cloud Native Buildpacks Cont.

```
$ QS_LOG_LEVEL=debug skaffold run --default-repo gcr.io/gcp-10-mins -f skaffold-v2.yaml
Generating tags...
 - square -> gcr.io/gcp-10-mins/square:latest
...
Target platforms: [linux/amd64]
v1: Pulling from buildpacks/builder
Digest: sha256:de4d669b82419307072df7b35c3c9a81aca35e308be2bac5ec4ca7dbcba31f24
Status: Image is up to date for gcr.io/buildpacks/builder:v1
...
===> BUILDING
[builder] === Go - Runtime (google.go.runtime@0.9.1) ===
[builder] Using runtime version from GOOGLE_RUNTIME_VERSION: 1.14
[builder] === Go - Gomod (google.go.gomod@0.9.0) ===
[builder] ---------------------------------------------------------------------------------------
[builder] Running "go mod download (GOPATH=/layers/google.go.gomod/gopath GO111MODULE=on)"
[builder] Done "go mod download (GOPATH=/layers/google.go.gomod/gopath GO111..." (536.9738ms)
[builder] === Go - Build (google.go.build@0.9.0) ===
[builder] ---------------------------------------------------------------------------------------
[builder] Running "go build -o /layers/google.go.build/bin/main cmd/square/main.go
(GOCACHE=/layers/google.go.build/gocache)"
[builder] Done "go build -o /layers/google.go.build/bin/main cmd/square/main..." (7.2431905s)
[builder] === Utils - Label Image (google.utils.label@0.0.2) ===
```

Update

# Deploy - Helm Cont.

v1

```
deploy:
  helm:
    flags:
      upgrade:
        - --install
    releases:
      - name: square
        chartPath: ./deployments/helm/square
        artifactOverrides:
          square.image: gcr.io/gcp-10-mins/square
        setValues:
          square.env.QS_LOG_LEVEL:  info
```

v2

```
manifests:
  helm:
    flags:
      upgrade:
        - --install
    releases:
    - name: square
      chartPath: deployments/square
      setValues:
        square.image: square
      setValueTemplates:
        square.env.QS_LOG_LEVEL:  "{{.QS_LOG_LEVEL}}"
deploy:
  helm:
    {}
```

Update

# Deploy - Helm Cont.

```
$ QS_LOG_LEVEL=debug skaffold run --default-repo gcr.io/gcp-10-mins -f skaffold-v2.yaml
Generating tags...
-  square -> gcr.io/gcp-10-mins/square:latest
Checking cache...
-  square: Found Remotely
Starting test...
Tags used in deployment:
 - square ->
gcr.io/gcp-10-mins/square:latest@sha256:36ff3a328a97a4ae13937516b697d63fc5d89621d69fba255683acaad6fa5eb0
Starting deploy...
Release "square" has been upgraded. Happy Helming!
NAME: square
LAST DEPLOYED: Tue Nov 29 08:21:19 2022
NAMESPACE: default
STATUS: deployed
REVISION: 4
TEST SUITE: None
Waiting for deployments to stabilize...
 - deployment/square: creating container square
    - pod/square-59ddb57855-cvt4b: creating container square
-  deployment/square is ready.
Deployments stabilized in 11.323 seconds
You can also run [skaffold run --tail] to get the logs
```

# Monitoring

## reliable, resilient, observable

# Observability & Monitoring

**Observability** is the property of the system that defines what we can tell about the state and behavior of the system, right now and historically.

- Gain understanding actively
- Ask questions based on hypotheses
- Build to tame dynamic environments with changing complexity
- Preferred by developers of systems with variability and unknown permutations

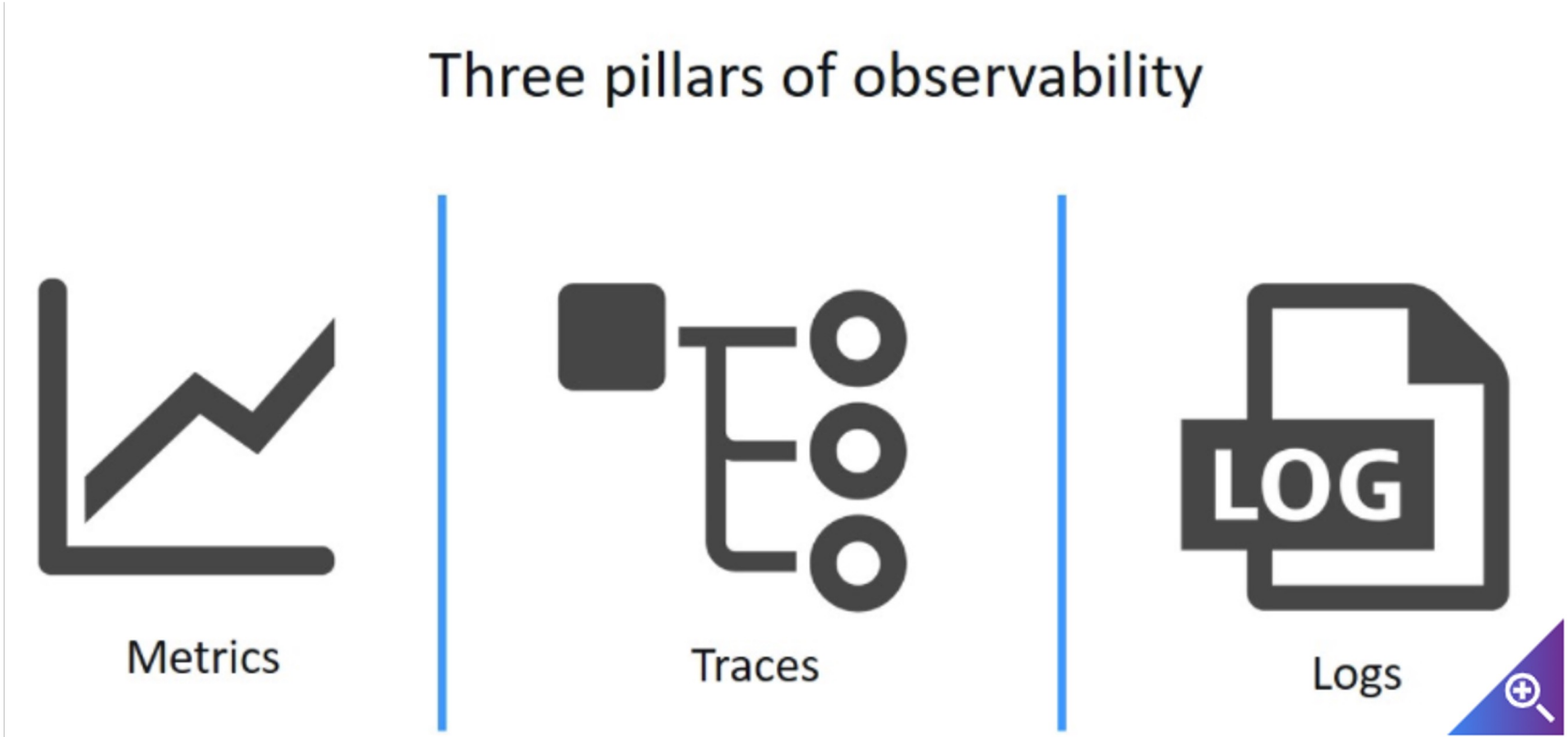
**Monitoring** is the collection of tolls, processes, and techniques we use to increase the observability of the system.

- Consume information passively
- Ask questions based on dashboards
- Built to maintain based on dashboards

Built to maintain static environments with little variation

Used by developers of systems with little change and known permutations

# Observability



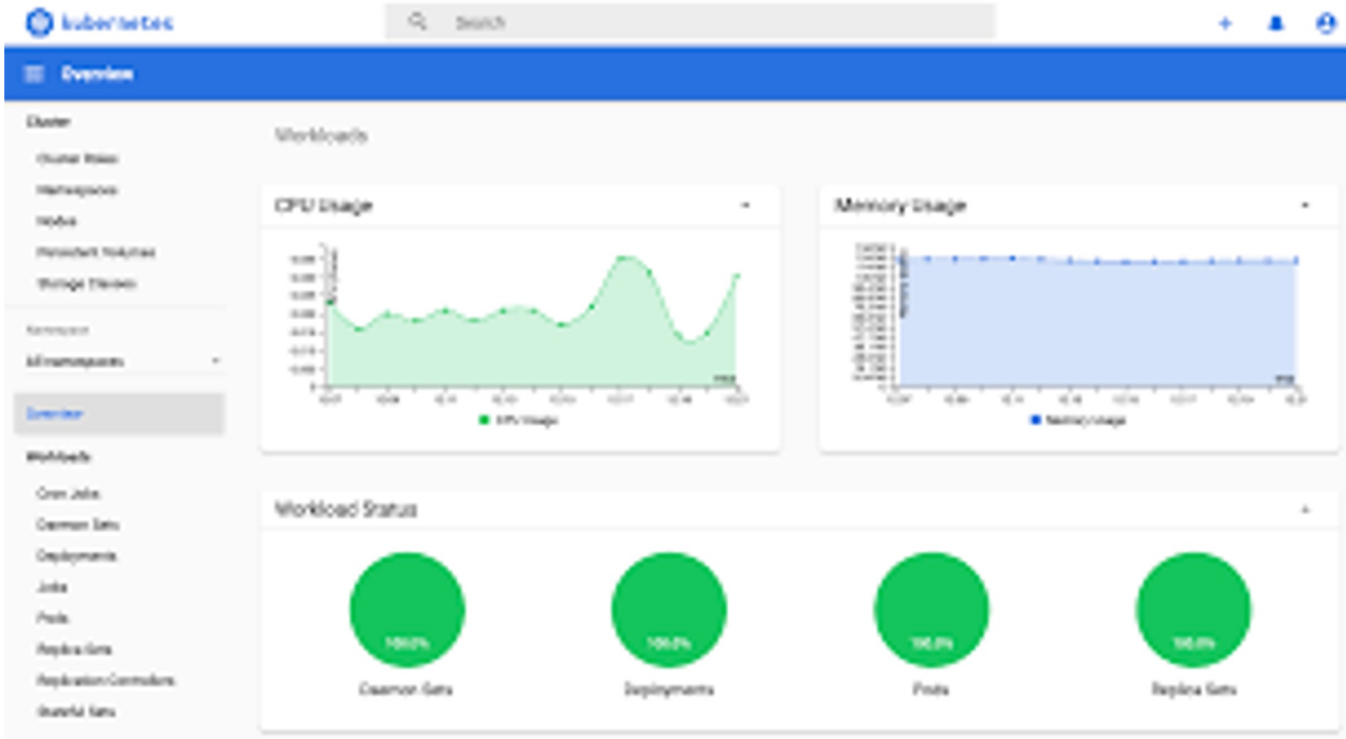
# Why?

# Metrics

Metrics measure the same aspect of the system over time. Metrics are time series of numerical values (typically, floating-point numbers). Each metric has a name and often a set of labels that help later in slicing and dicing. For example, the CPU utilization of a node or the error rate of a service are metrics.

# Monitoring Metrics using Kubernetes Dashboard

Kubernetes Dashboard doesn't display detailed metrics unless Kubernetes Metrics Server is installed and the kubernetes-metrics-scraper sidecar container is running



# Rise of Prometheus

The following diagram illustrates the entire system:

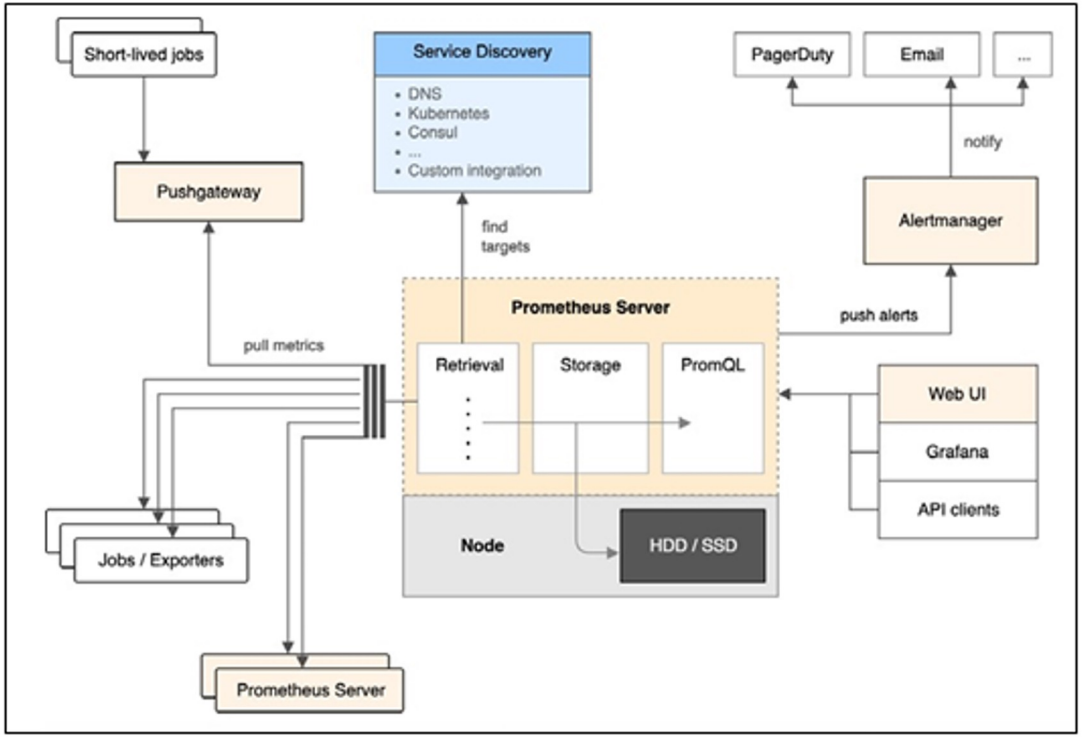
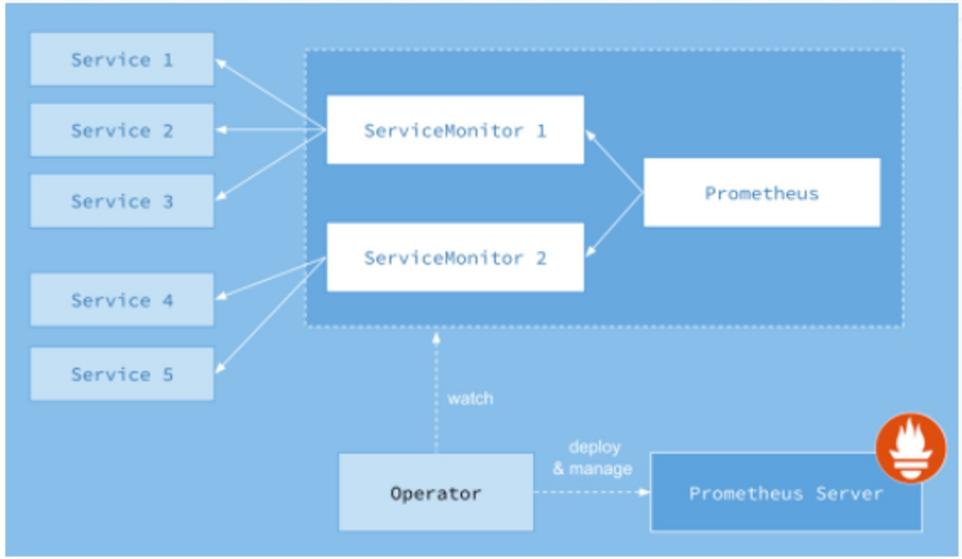



Figure 13.7: the Prometheus system

# Installing Prometheus

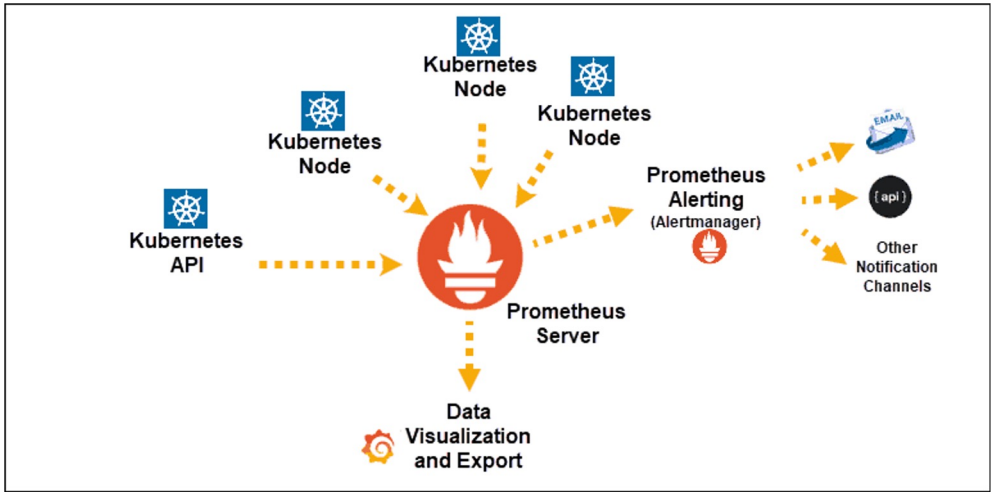



# Monitoring with Prometheus

**Why Use Prometheus for Kubernetes Monitoring**

Two technology shifts took place that created a need for a new monitoring framework:

1. DevOps culture
2. Containers and Kubernetes



# Interacting With Prometheus

Prometheus has a basic web UI that you can use to explore its metrics. Let's do port forwarding to localhost:

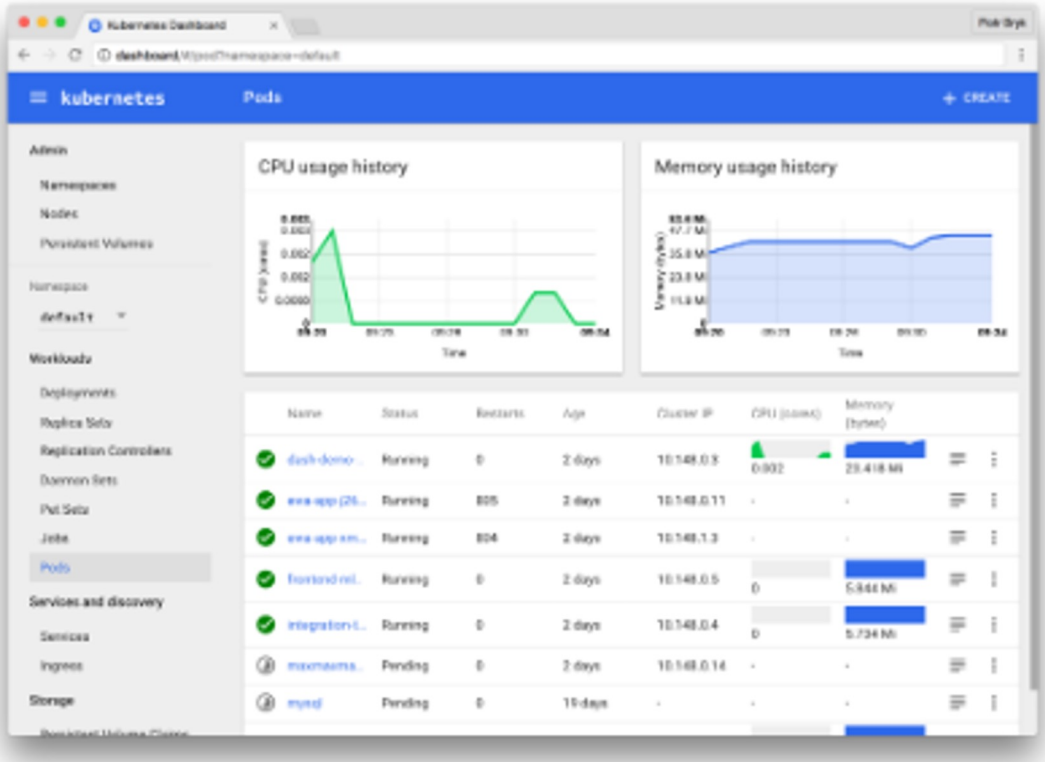
```
$ POD_NAME=$(kubectl get pods -n
monitoring -l "app=prometheus" \
            -o
            jsonpath="{.items[0].metadata
            .name}")
$ kubectl port-forward -n monitoring
$POD_NAME 9090
```

Then, you can browse to `http://localhost:9090`, where you can select different metrics and view raw data or graphs:

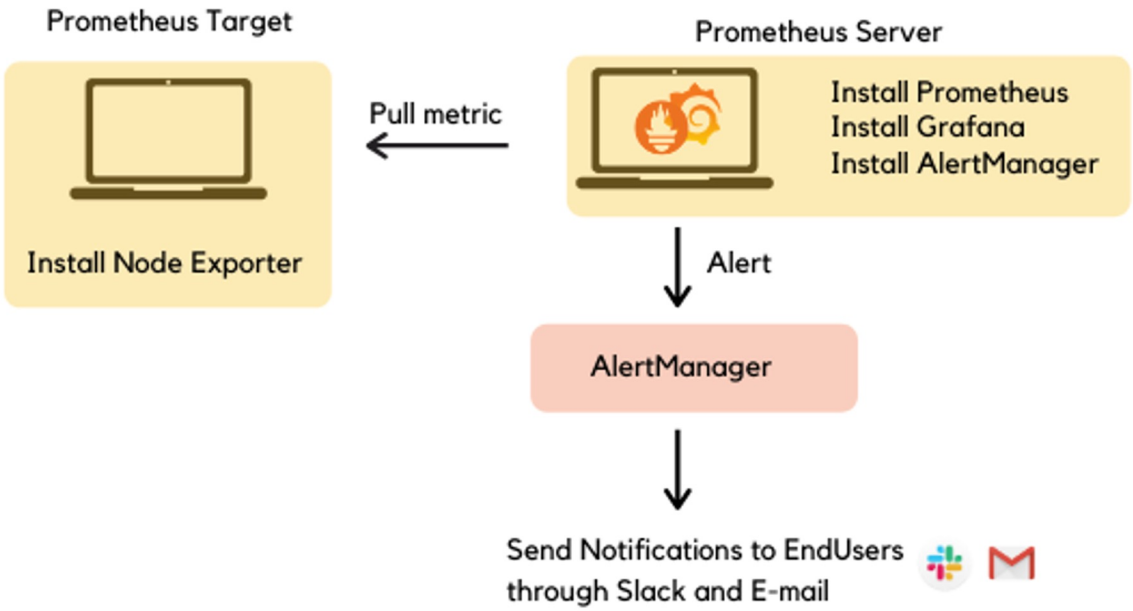

Prometheus records an outstanding number of metrics (990, in my current setup). The most relevant metrics on Kubernetes are the metrics exposed by kube-state- metrics and node exporters.

# Incorporating Kube-state-metrics

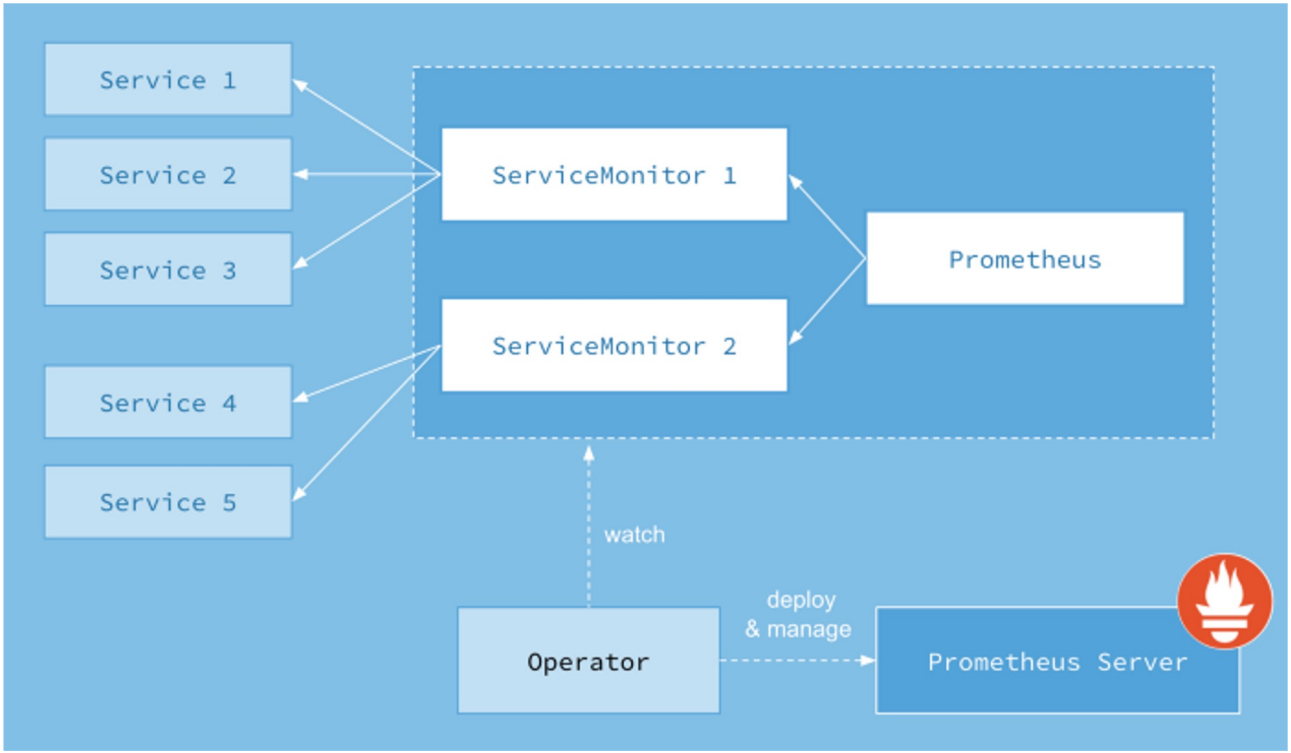
The Prometheus operator already installs kube-state-metrics. It is a service that listens to Kubernetes events and exposes them though a `/metrics` HTTP endpoint in the format that Prometheus expects.



# Triggering Alerts with AlertManager



# Prometheus Operator

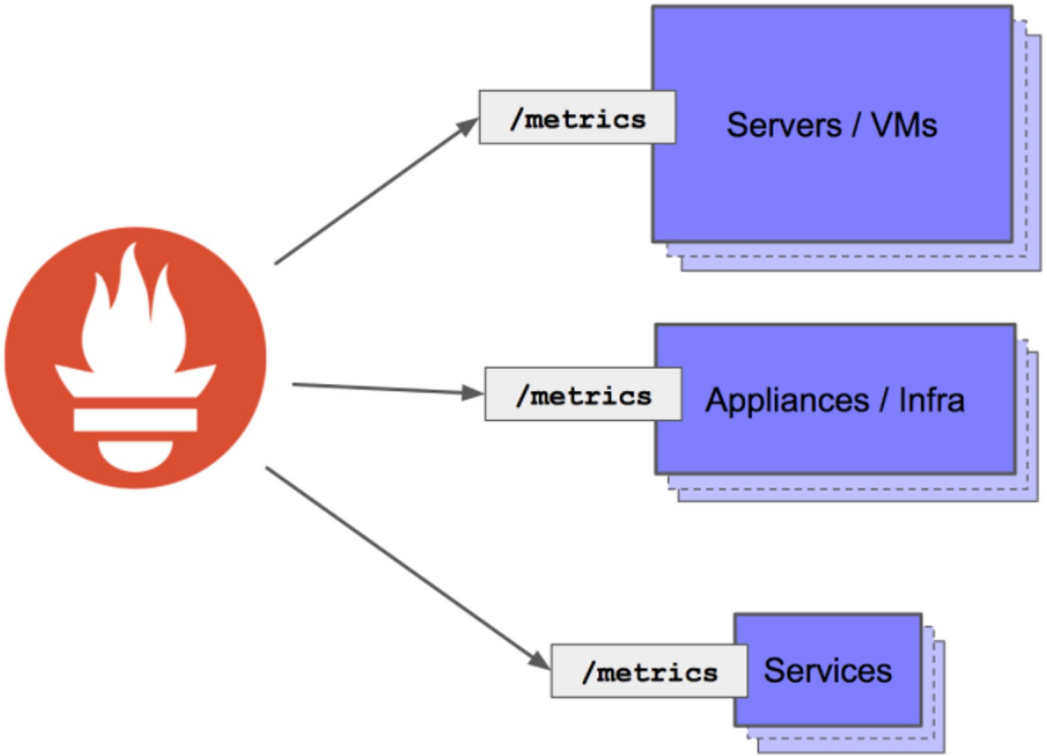



The Prometheus Operator provides easy monitoring for k8s services and deployments besides managing Prometheus, Alertmanager and Grafana configuration.

# Prometheus Operator


- **Kubernetes Custom Resources**: Use Kubernetes custom resources to deploy and manage Prometheus
- **Simplified Deployment Configuration**: Configure the fundamentals of Prometheus like versions, persistence, retention policies, and replicas
- **Prometheus Target Configuration**: Automatically generate monitoring target configurations based on Kubernetes label queries

# Exposing Metrics with Prometheus



# Node Exporter

kube-state-metrics collects node information from the Kubernetes API server, but this information is pretty limited.

Prometheus comes with its own node exporter, which collects tons of low-level information about the nodes.

Remember that Prometheus may be the de facto standard metrics platform on Kubernetes, but it is not Kubernetes-specific.

Here is a small subset of the metrics exposed by the node exporter:

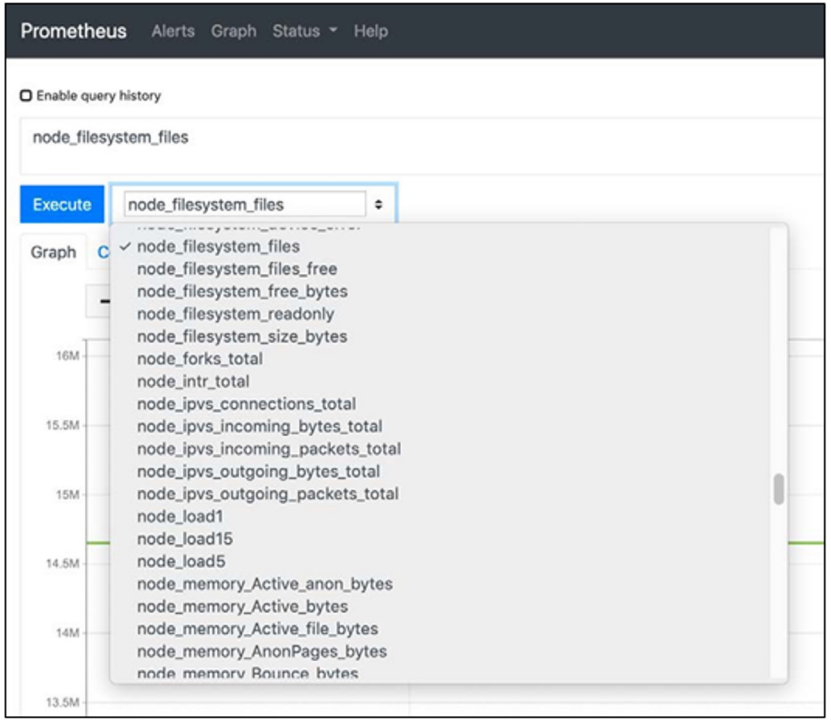


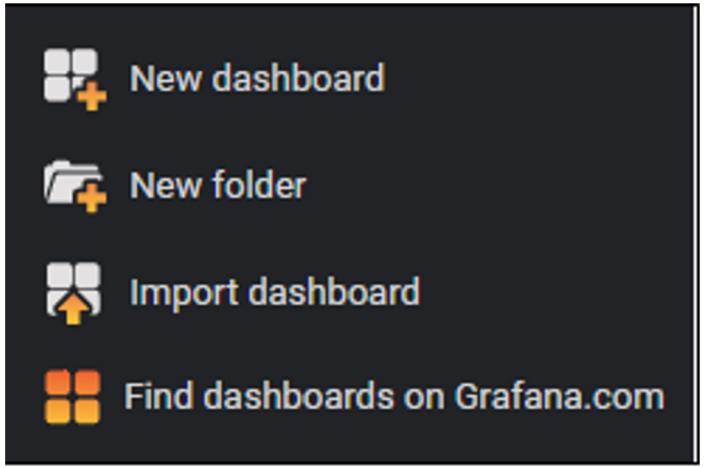

Figure 13.9: Metrics exposed by the node exporter

# Grafana

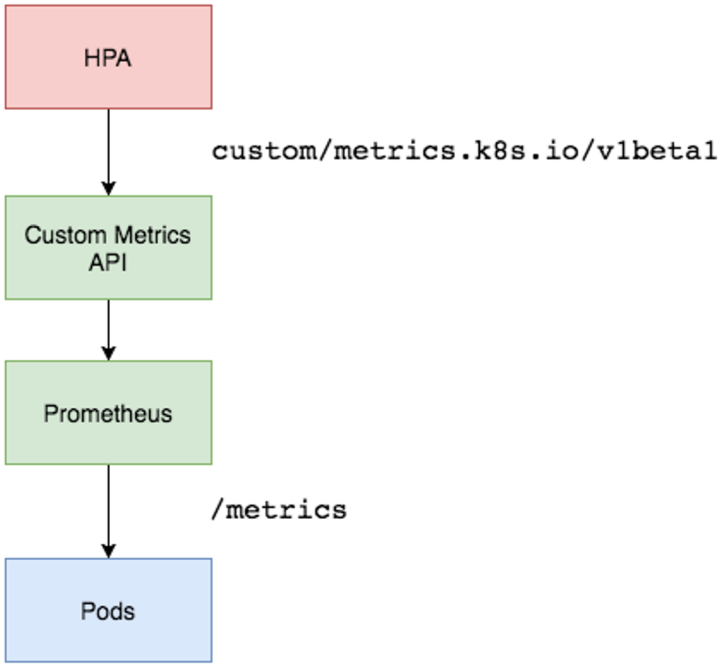
Grafana is an open source analytics and monitoring solution. By default, Grafana is used for querying Prometheus. Follow these instructions to expose the included Grafana service instance and access it through your web browser:

# Grafana Dashboard

Grafana is an open sourced monitor used to visualize the metrics stored on Prometheus. It offers dynamic and reusable dashboards with template variables. In this recipe, we will learn how to add a new dashboard from the library of pre-built dashboards to monitor an application deployed on Kubernetes.



# Creating Custom Metrics to Trigger Cluster Autoscaling

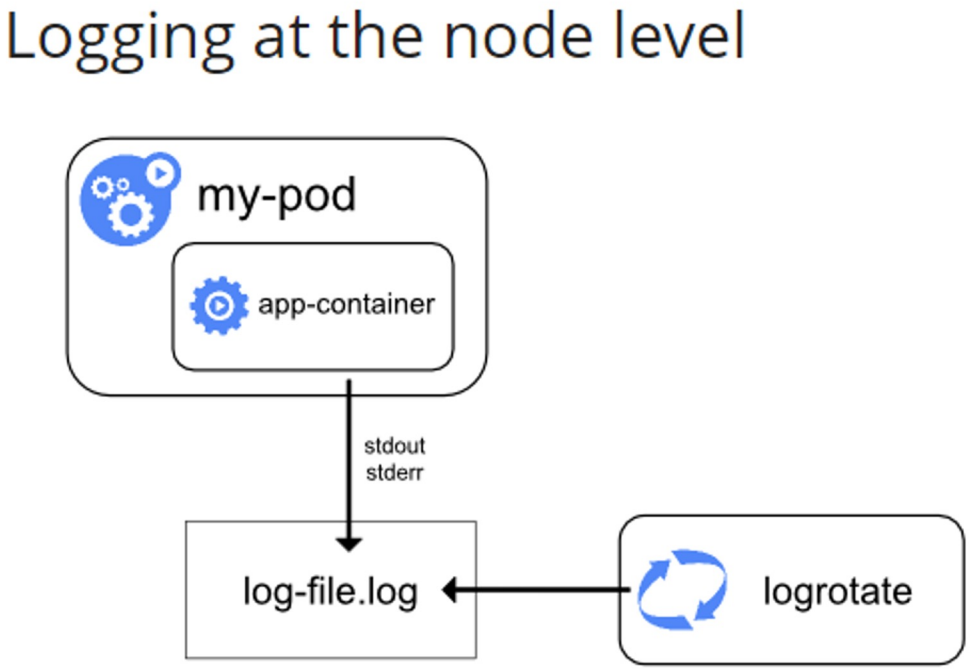



# Logging

Logging is a key monitoring tool. Every self-respecting long-running software must have logs. Logs capture timestamped events. They are critical for many

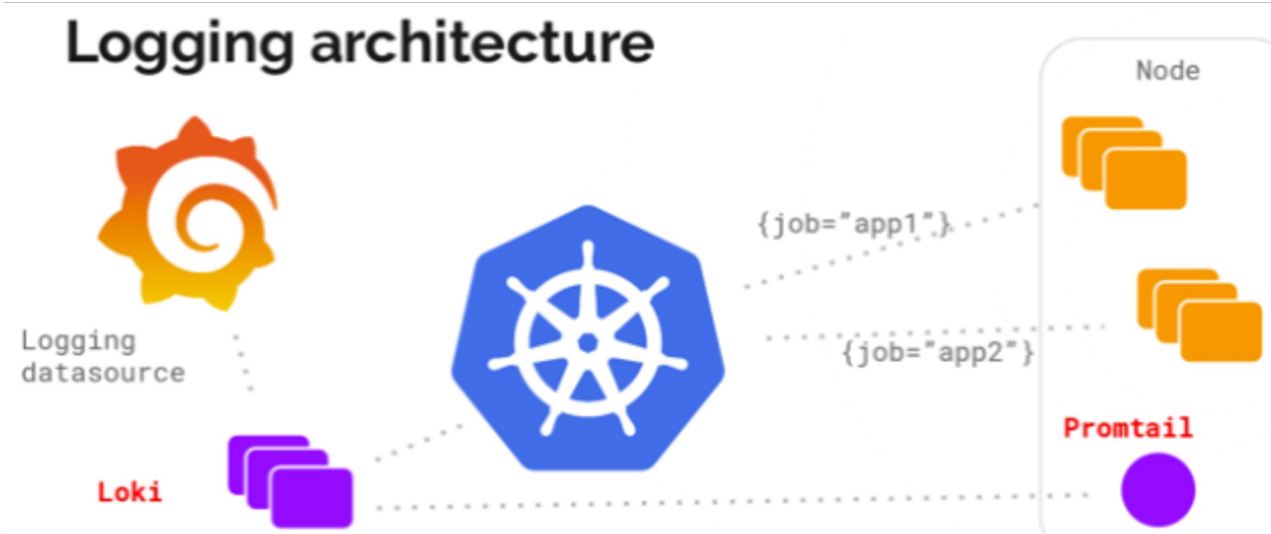
applications, like business intelligence, secu

# Logging Solution



# Loki

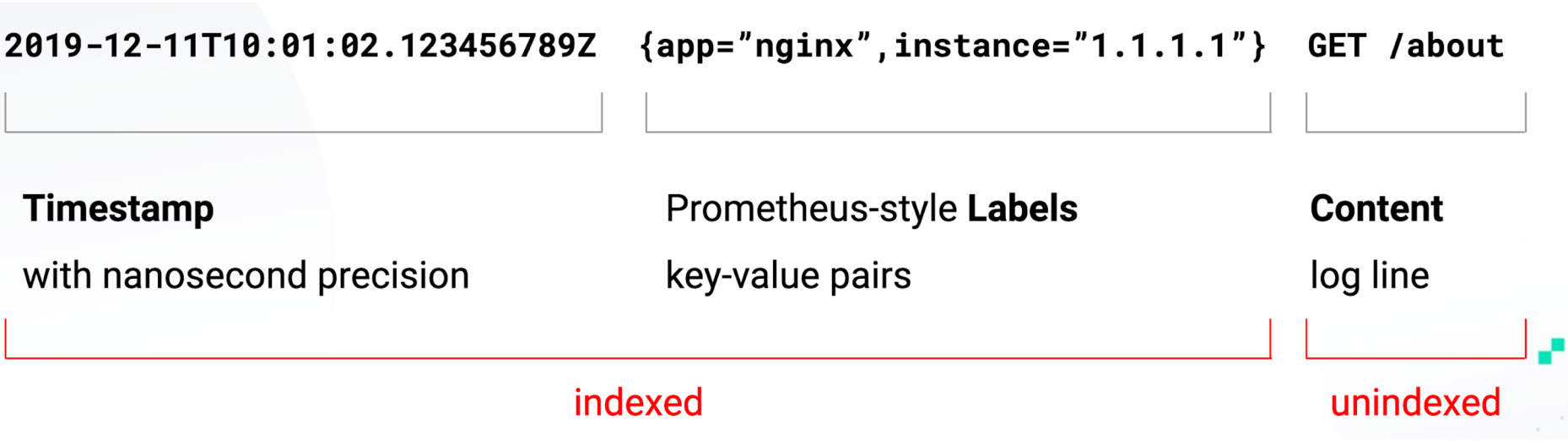
Loki is a logging backend optimized for users running Prometheus and Kubernetes.




Loki was built for efficiency alongside the following goals:

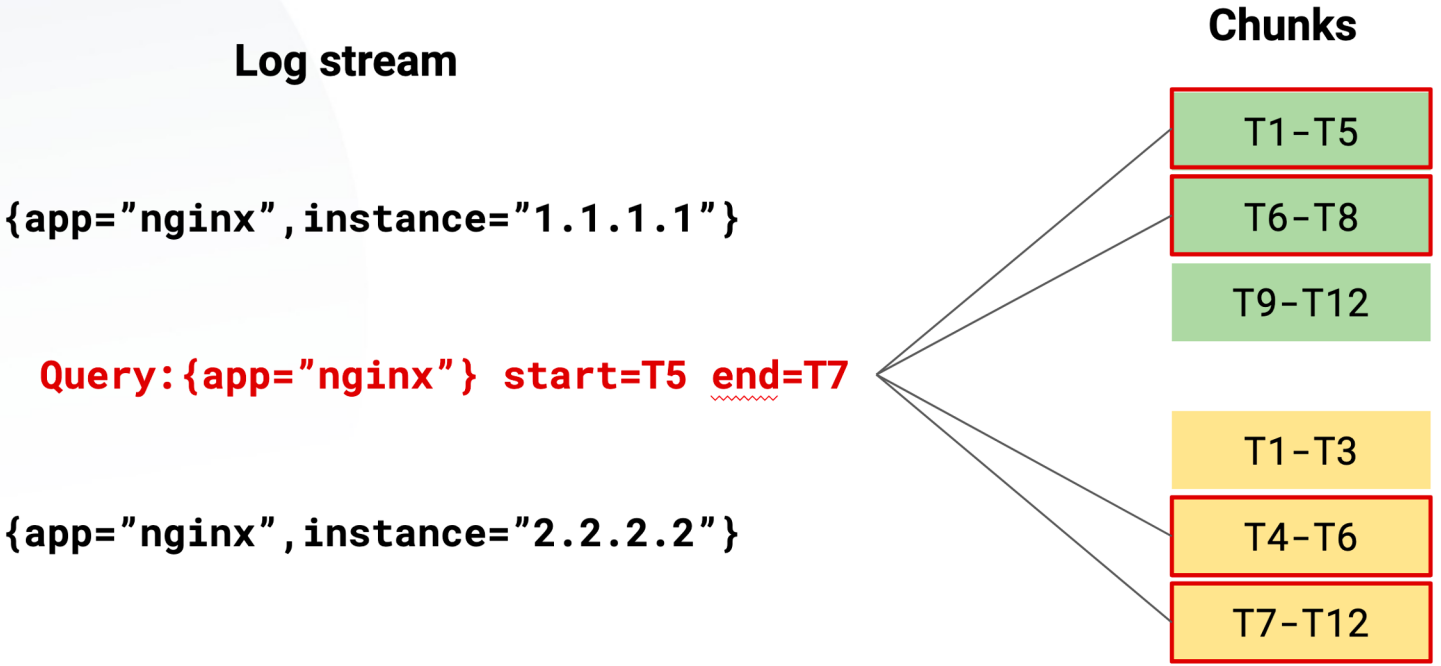
- Logs should be cheap. Nobody should be asked to log less.
- Easy to operate and scale.
- Metrics, logs (and traces later) need to work together.

# Logging with Loki



# Selecting Log Streams

Selecting logstreams, is done by using **label matchers** and **filter expressions**



# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

What are the three pillars of observability?

A: Metrics, Traces, Logs

B: Technique, Process, Metrics

C: Efficiency, Metrics, Monitoring

# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

What are the three pillars of observability?

A: Metrics, Traces, Logs

B: Technique, Process, Metrics

C: Efficiency, Metrics, Monitoring

# POP QUIZ:

## Kubernetes: Controller, Operator Patterns, Creation

What is the difference between Grafana and Prometheus?

A: Grafana is an open-source visualization software, which helps the users to understand the complex data with the help of data metrics.Prometheus is an open-source event monitoring and alerting tool. It stores the majority of the data locally after scrapping metrics

B: Prometheus Is an open-source visualization software, which helps the users to understand the complex data with the help of data metrics. Grafana is an open-source event monitoring and alerting tool. It stores the majority of the data locally after scrapping metrics

# POP QUIZ:

## KUBERNETES DEVOPS

What is the difference between Grafana and Prometheus?

A: Grafana is an open-source visualization software, which helps the users to understand the complex data with the help of data metrics.Prometheus is an open-source event monitoring and alerting tool. It stores the majority of the data locally after scrapping metrics

B: Prometheus is an open-source visualization software, which helps the users to understand the complex data with the help of data metrics. Grafana is an open-source event monitoring and alerting tool. It stores the majority of the data locally after scrapping metrics

# POP QUIZ:

## KUBERNETES DEVOPS

What is the mission of Prometheus Operator?

A: To make running prometheus as efficient as possible

B: To make running prometheus as fast as possible

C: To make running Prometheus on top of Kubernetes as easy as possible

# POP QUIZ:

## KUBERNETES DEVOPS

What is the mission of Prometheus Operator?

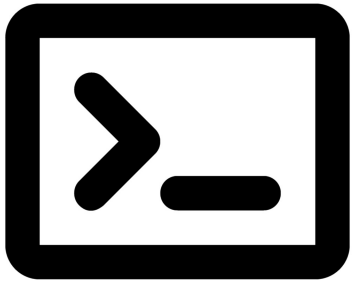
A: To make running prometheus as efficient as possible

B:To make running prometheus as fast as possible

C: To make running Prometheus on top of Kubernetes as easy as possible

# Tilt

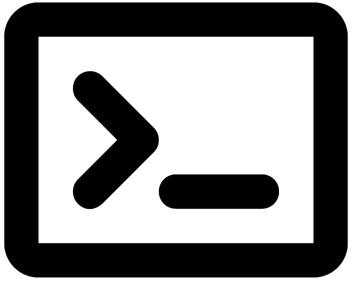
# Tilt Resource Definitions



Resource Definitions
A resource is a collection of workloads managed by Tilt. Some of the workloads include,

- Container images
- Kubernetes manifests (YAML)
- A local command

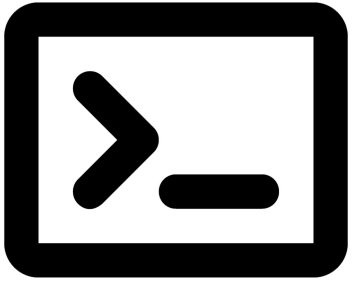
# Tilt Resource Definitions



tilt-resource-definition-example.yaml

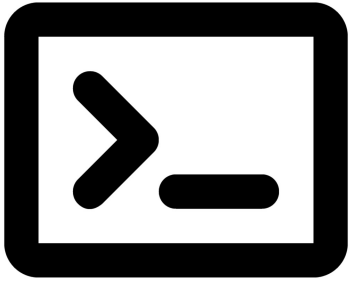
```yaml
# Default values for bots.
# This is a YAML-formatted file.
# Declare variables to be passed into your
templates.
replicaCount: 1
image:
  repository: bots
autoscaling:
  enabled: false
serviceMonitor:
  enabled: false
```

# Tiltfile



A Titfile is a configuration file written in Starlark, a dialect of python. It consists of all the resource definitions and how they connect. It’s actual code; you can use conditionals, loops, and functions to define your microservice environment.

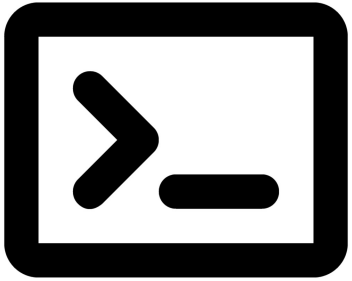
# Tiltfile Example



Tiltfile Example

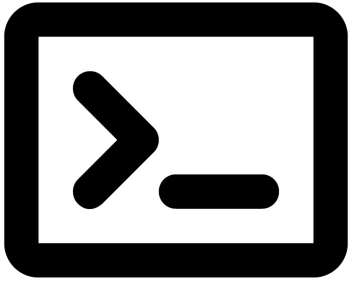
```
# Example Tiltfile

# Read the configuration
if not os.path.exists("./tilt_config.json"):
    fail("Config not found")

config.define_string_list("microservices")

cfg = config.parse()

# Build each microservice image as stated in the tilt_config.json
file
for microservice in cfg.get("microservices"):
    docker_build(
        microservice,' pkg'
    )
```

# Tiltfile



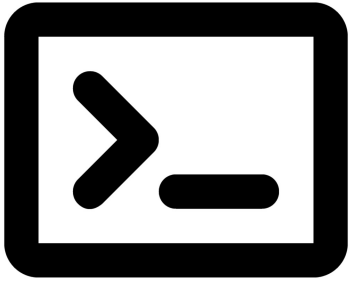
When the Tiltfile is executed, the YAML is packaged as a resource and sent to the Tilt engine.

# Tilt Engine



The Tilt engine is a controller that constantly watches the files fed into it. If Tilt detects any changes that might affect the output of the Tiltfile, it re-evaluates the Tiltfile.

# Tilt Engine

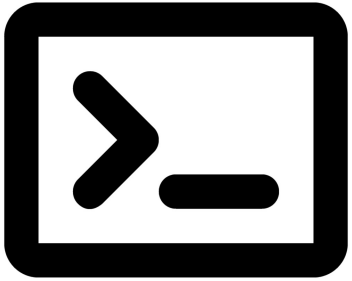


The Tilt engine watches critical events like:

- Change the resource's definition in the Tiltfile.
- Manual triggering of a particular resource by the developer

For example, as you edit your code, Tilt will automatically update steps such as building an updated container image, and if any resources contain Kubernetes objects, these end up automatically deploying to your development cluster.

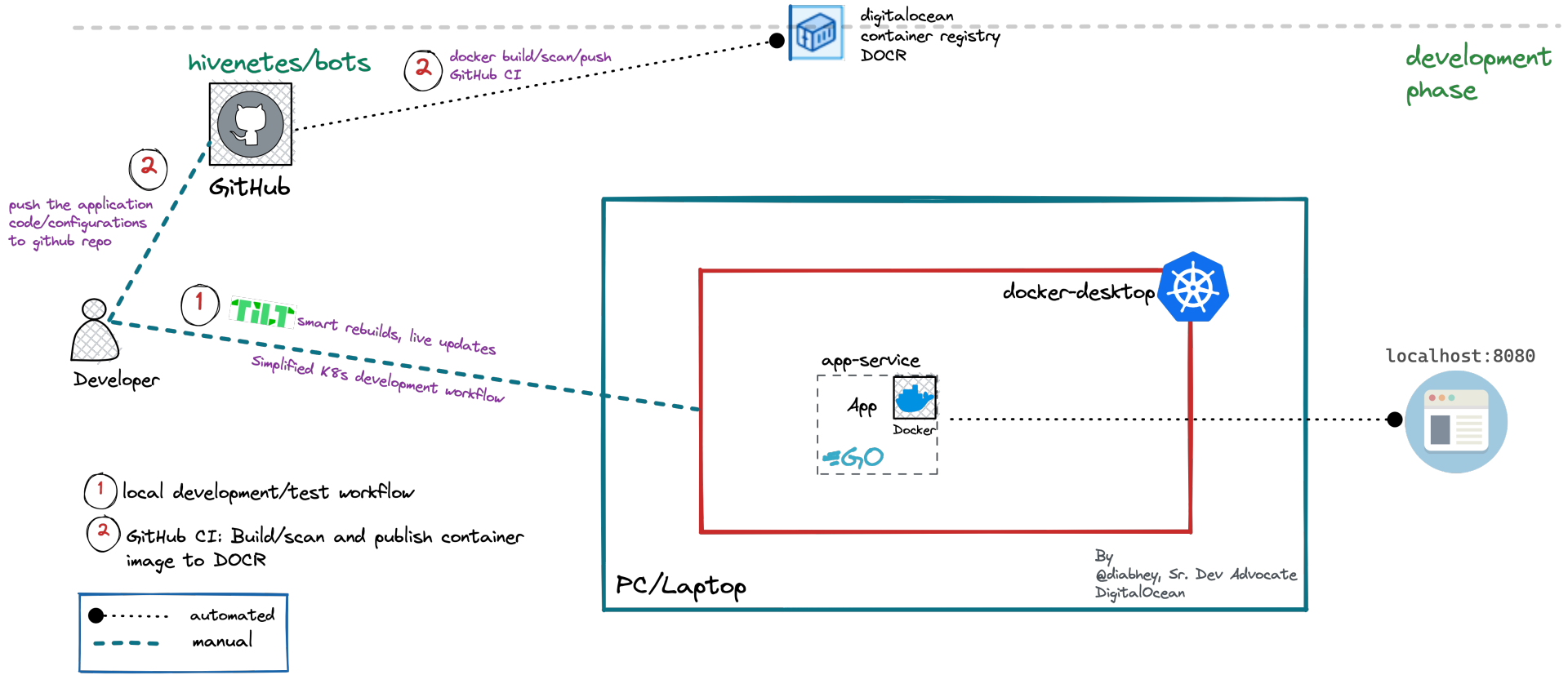
# Microservices Dev Flows



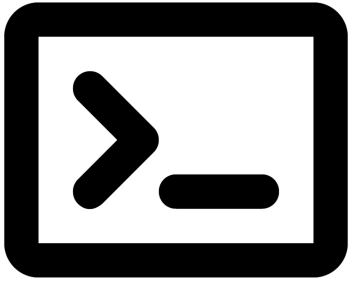
Inner development loop:
iterative process of writing, building, and debugging code that a single developer performs before sharing the code publicly or with their team.

Outer development loop:
starts as soon as your code is pushed into version control (or you send a PR for the same). This is where you hand over most of the responsibilities to automated systems and CI/CD pipelines to do the job for you as per a typical GitOps workflow.

# Microservices Dev Flows



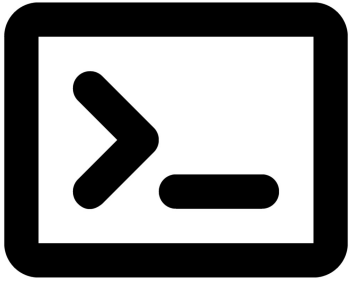
# Development inner loop



A common workflow is:

1. Save changes in code editor
2. Build and tag Docker image
3. Push image to registry
4. Re-deploy Kubernetes manifests
5. Run commands to observe changes

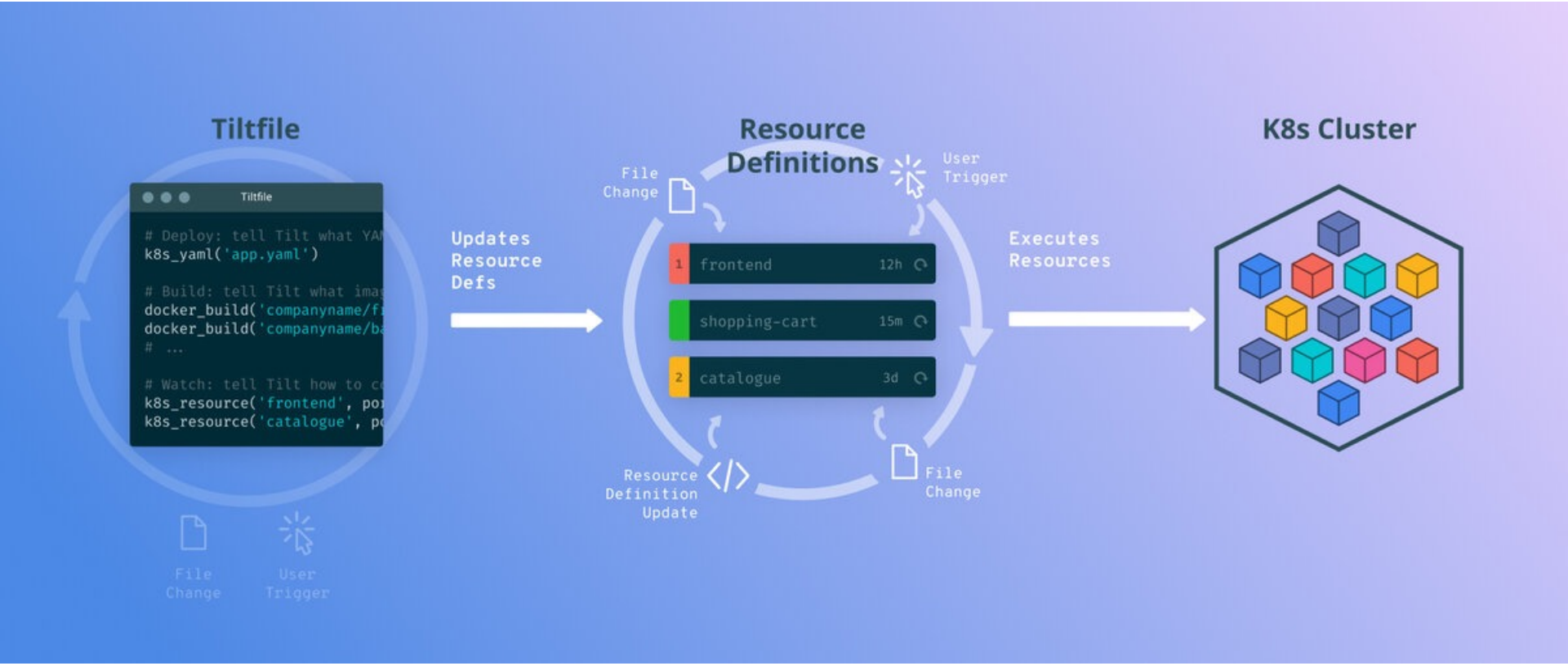
# Tilt simplifies



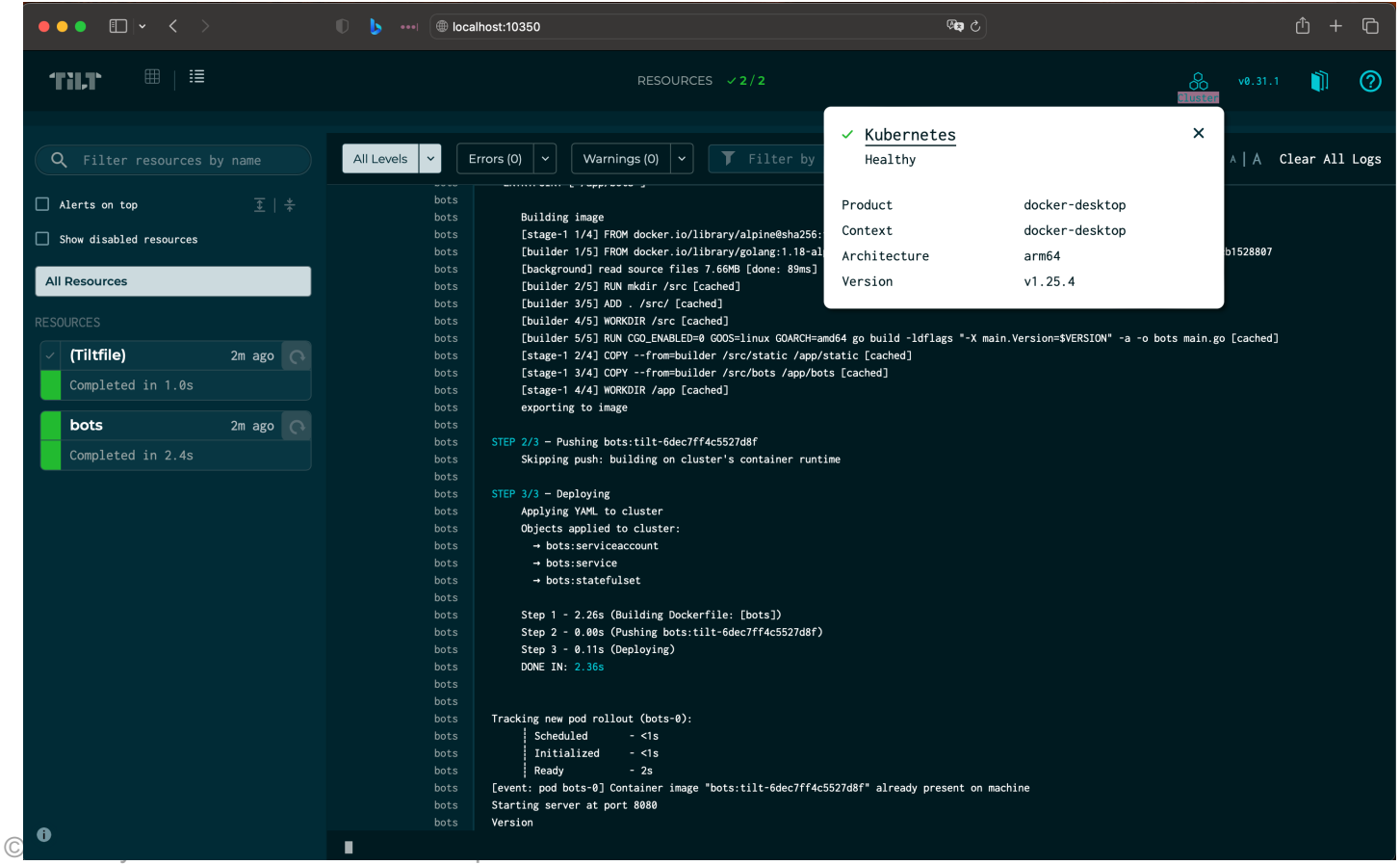
Til workflow:

1. Save changes in code editor
2. View logs and errors of deployed code in UI

# Tilt Control Flow



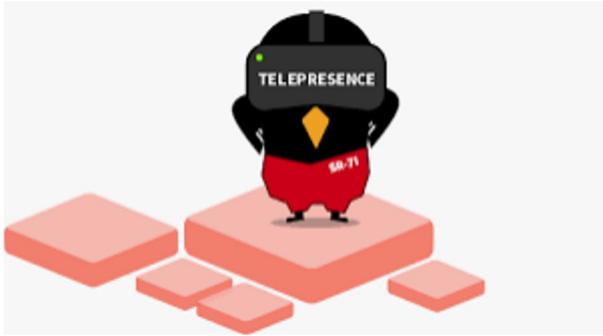
# Tilt User Interface



# Telepresence

telepresence is an open source tool that lets you run a single service locally, while connecting that service to a remote Kubernetes cluster.

# Using telepresence to run local code in a remote server




You can run a local process using Telepresence that can access that service, even though the process is local but the service is running in the Kubernetes cluster:

```
$ telepresence --run curl http://myservice:8000/
```

```
Hello, world!
```

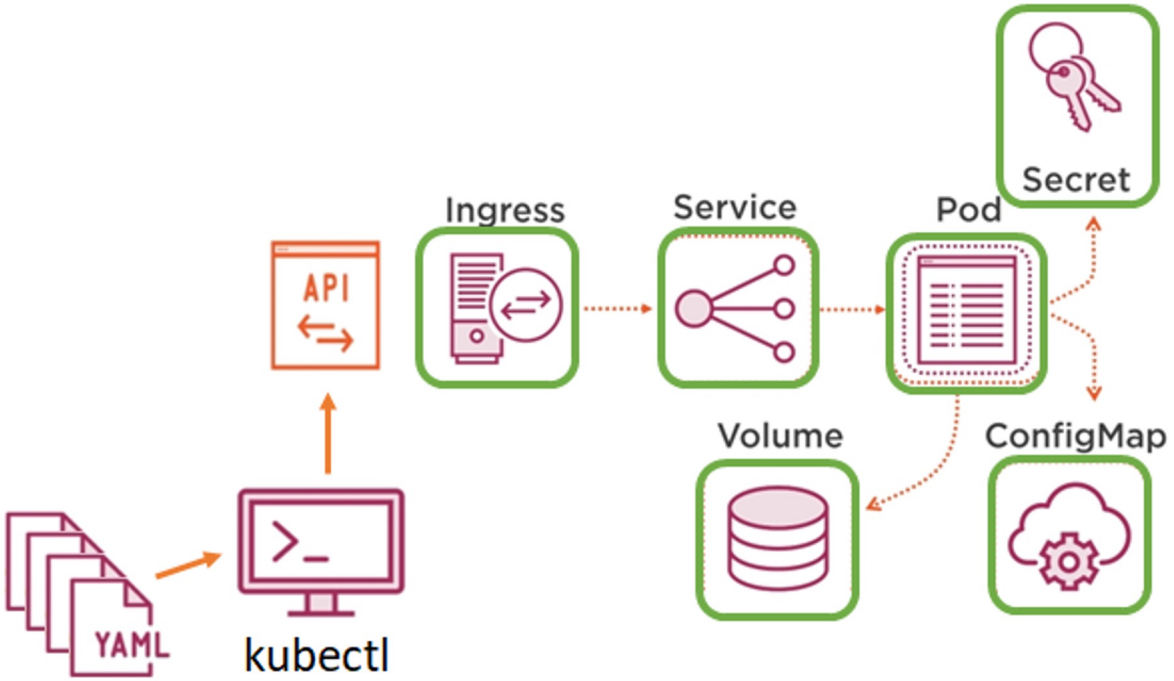
# Attaching a debugger to a running container on a remote server

This debugging session type allows you to connect the IDE to a remote target and debug a process running there. By remote target, we can consider containers running on the local machine remote targets or actual servers either on-premise or in the cloud.

# Attaching a debugger to a running container on a Local machine

To ensure that the debugging session is successful and you can debug the application without issues, all you need to do is to compile your application with a special flag. The IDE will add these flags automatically for the other configuration types, so these are necessary only when compiling the application manually.

# Kubernetes Does Everything



# Problem Statement

# Does Kubernetes need a package manager?

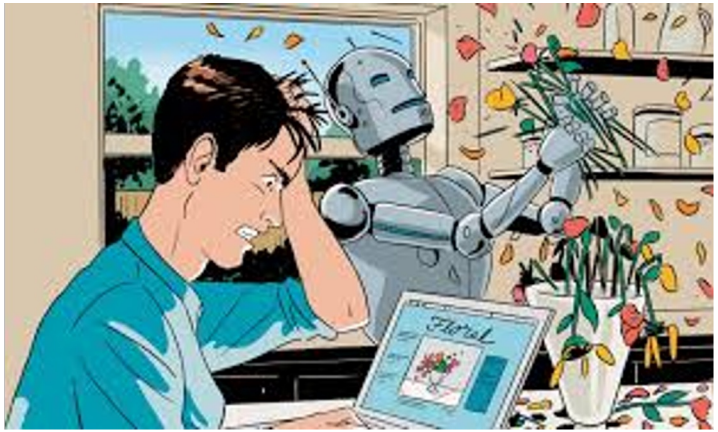



YUM

# Humans on the other hand ...

tend to struggle with YAML manifests which can be hundreds or thousands of lines long. Optimally we don't want to maintain manifests for every permutation of an application or manage its versioning and lifecycle.

We'd much rather offload those tasks to a reliable (and automatable) tool with the flexibility to modify parameters for us when instructed to do so.



# Reference

Develop a passion for learning.

# BOOK: ANTIFRAGILE SOFTWARE

https://leanpub.com/antifragilesoftware

Written by Russ Miles with Sylvain Hellegoarch, and Grant Tarrant-Fisher.

These are the folks who have driven a boatload of the progress in Chaos Engineering and the Chaos Toolkit in the last two years.

# SITE: MEDIUM.COM

https://medium.com/search?q=kind

https://medium.com/search?q=k3d

https://medium.com/search?q=observability

Medium.com is an emerging website with comprehensive technical content for developers. The content is detailed and with plenty of examples.  Great online resource for cloud-native engineering content.

# SITE: Rancher


https://rancher.com/resources/

# Kubernetes @ Edge

https://forums.rancher.com/

Rancher Labs are arguably the folks have that driven the work on Kubernetes at the Edge and IoT. Their forum is a great place to learn more about k3d and implementation for IoT and Edge computing

# Docker Cheat Sheet

https://www.docker.com/sites/default/files/d8/2019-09/docker-cheat-sheet.pdf

Quick reference documenting the majority of commands and the key commands used in not only our sessions but most people's day to day work

# Still More Kubernetes

- Kubernetes Guide, with 20+ articles and tutorials
- BMC DevOps Blog
- Bring Kubernetes to the Serverless Party
- How eBay is Reinventing Their IT with Kubernetes & Replatforming Program

Interesting additional articles and references on Kubernetes

# More Kubernetes

- What is Kubernetes?
- eBook: Storage Patterns for Kubernetes
- Test drive OpenShift hands-on
- eBook: Getting started with Kubernetes
- An introduction to enterprise Kubernetes
- How to explain Kubernetes in plain terms
- Latest Kubernetes articles

Interesting additional articles and references on Kubernetes