Kubernetes Ecosystem

phkoan, lichen (2026, CC BY-SA)

Outline

- Container Image
- Custom Resource Definition (CRD)
- Cloud Native Computing Foundation (CNCF)

Container Image

Container Image - Intro

- A container image
  - Binary data
  - Encapsulates applications and all their software dependencies
  - Runs standalone
- For example, Docker image

Container Image - OCI

- Open Container Initiative (OCI) by Linux Foundation
  - Runtime Specification (runtime-spec)
    - The spec for runtime interface
    - opencontainers/runtime-spec: OCI Runtime Specification · GitHub
  - Image Specification (image-spec)
    - The spec for the format of image
    - opencontainers/image-spec: OCI Image Format · GitHub
  - Distribution Specification (distribution-spec)
    - The spec for image registry API
    - opencontainers/distribution-spec: OCI Distribution Specification · GitHub

Recap: Container Image - Dockerfile

- Docker builds images by executing the instructions in a Dockerfile.
- A Dockerfile is a text file containing instructions for building your image.
- Some common instructions
  - `FROM` - Defines a base for your image.
  - `RUN` - Execute build commands.
  - `WORKDIR` - Sets the working directory.
  - `COPY` - Copy files and directories.
  - `CMD` - Specify default commands.
  - `ENV` - Set environment variables.

Recap: Image - Build

- Registry
  - docker.io, ghcr.io
- Image name
  - ubuntu, busybox, redis
- Image tag
  - latest, 8.7-alpine
- Build with tag
  - `docker build -t {registry}/{image name}:{image tag} {Dockerfile path}`
- Multi-arch build
  - `docker buildx build --platform linux/amd64,linux/arm64 -t {registry}/{image name}:{image tag} {Dockerfile path}`

Recap: Image - Layer

Each layer in an image represents a set of filesystem changes - additions, deletions, or modifications.

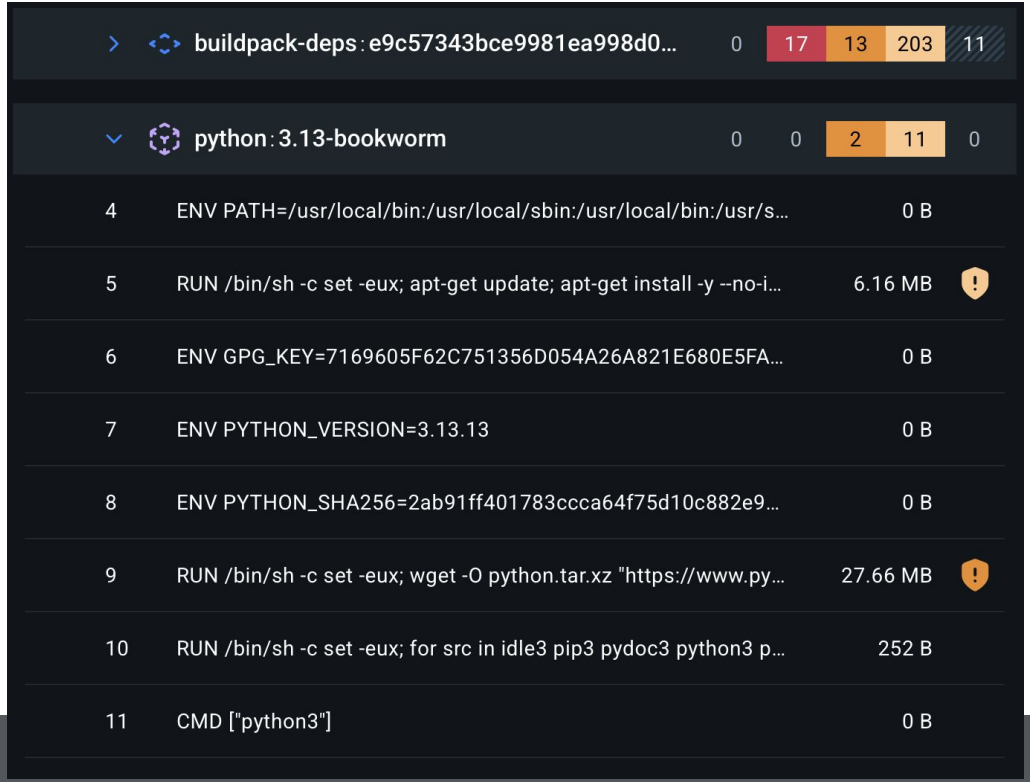


Image - Registry

- A registry is a centralized location that stores and manages container images.
- A repository is a collection of related container images within a registry.

Image - Registry Platform

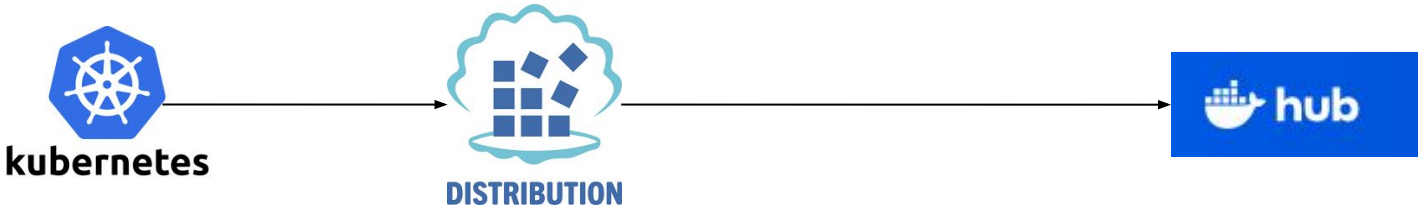
- Docker Hub
- GitHub Container Registry
- Amazon ECR Public Gallery

Image - Self Hosted Registry

- In a company, the images can be confidential.
- Uploading the images to public registries is not a choice.
- We can self-host a registry in the company.
- Some common self-hosted registries
  - GitLab Container Registry
  - Harbor
  - Distribution

Image - Cache Server

- Most public registry platforms have rate limits.
  - E.g., 200 requests per 6 hours for authenticated users in Docker Hub (ref.)
- We can use some self-hosted registries as cache server.
  - CNCF Distribution has cache server feature.



Custom Resource Definition

Recap: K8s Kind

```
kubectl api-resources --api-group=''
```

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
events	ev	v1	true	Event
limitranges	limits	v1	true	LimitRange
namespaces	ns	v1	false	Namespace
nodes	no	v1	false	Node
persistentvolumeclaims	pvc	v1	true	PersistentVolumeClaim
persistentvolumes	pv	v1	false	PersistentVolume
pods	po	v1	true	Pod
podtemplates		v1	true	PodTemplate
replicationcontrollers	rc	v1	true	ReplicationController
resourcequotas	quota	v1	true	ResourceQuota
secrets		v1	true	Secret
serviceaccounts	sa	v1	true	ServiceAccount
services	svc	v1	true	Service

CRD - Why CRD?

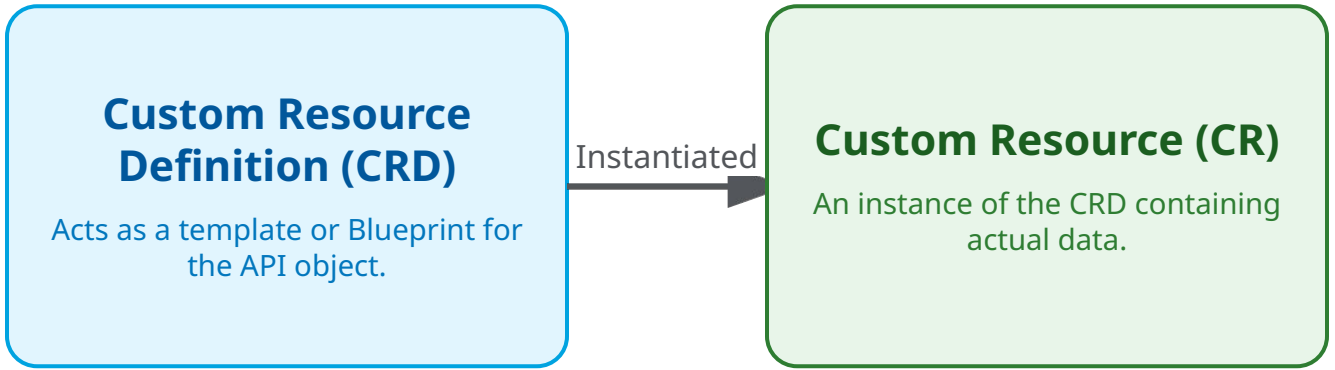
- CRD: **Custom** Resource Definition
- The built-in kinds in K8s may be insufficient for custom logic.
- For example,
  - Complex configuration
  - Create/Delete hooks
  - Auto discovery

CRD - Intro

- The CustomResourceDefinition (CRD) API resources allows users to define Custom Resource.
- Custom Resource is a extension of the Kubernetes API.
- For example, IPPools (one of Calico CRD) can be used to define a set of IP addresses.

CRD - Relation

- Custom Resources (CRs) store structured data.
  - Like advanced ConfigMap
- CRDs define the schema and API for Custom Resources.
- CRD vs. CR is like class vs. object in OOP.
- The users have to apply CRDs first, and they can create CRs.



CRD - Custom Controller

- Custom Resources let users to store and retrieve structured data.
- When you combine a custom resource with a custom controller, custom resources provide a true declarative API.

```
kubectl get po -n calico-system
NAME                                       READY   STATUS    RESTARTS        AGE
calico-kube-controllers-5685945ff6-rjctn   1/1     Running   0               11d
```

CRD - Schema

- kind
  - `Shirt`
- group
  - `stable.example.com`
- versions
  - v1

Resource name: `{kind}.{group}/{version}`

- scope
  - `Namespaced`
- openAPIV3Schema

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: shirts.stable.example.com
spec:
  group: stable.example.com
  scope: Namespaced
  names:
    plural: shirts
    singular: shirt
    kind: Shirt
  versions:
  - name: v1
    served: true
    storage: true
    schema:
      openAPIV3Schema:
        type: object
        properties:
          spec:
            type: object
            properties:
              color:
                type: string
              size:
                type: string
```

CRD - Schema / Custom Resources

The CRs created from the Shirt CRDs look like below.

```yaml
---
apiVersion: stable.example.com/v1
kind: Shirt
metadata:
  name: example1
spec:
  color: blue
  size: S
---
apiVersion: stable.example.com/v1
kind: Shirt
metadata:
  name: example2
spec:
  color: blue
  size: M
```

```
kubectl get shirts.stable.example.com
```

The output is:

```
NAME       COLOR   SIZE
example1   blue    S
example2   blue    M
```

CRD - Schema / OpenAPI

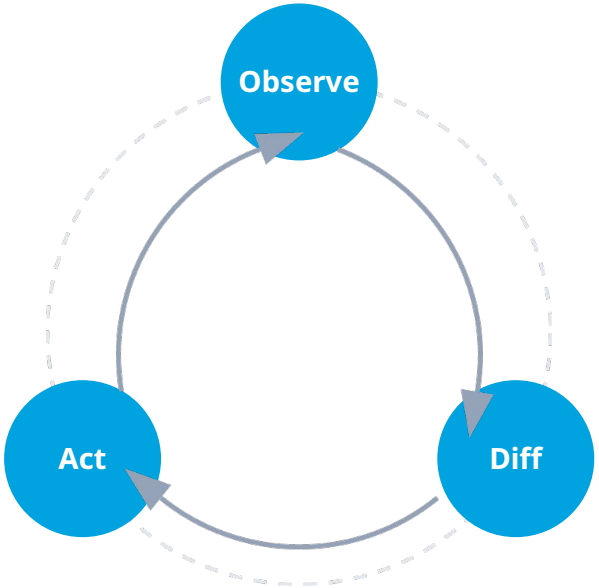
- OpenAPI Specification v3 (OAS3) is an industry-standard, language-agnostic format for describing RESTful APIs
- It is maintained by the OpenAPI Initiative (a Linux Foundation project), with v3.0 released in 2017 and v3.1 released in 2021
- It can be written in JSON or YAML; both formats are fully supported.

CRD - Operator (Controller of CR)

- Kubernetes' **operator pattern** concept lets you extend the cluster's behaviour without modifying the code of Kubernetes itself by linking controllers to one or more custom resources.
- Operators are clients of the Kubernetes API that act as **controllers for a Custom Resource**.
- Recap: kube-controller-manager

CRD - Operator Pattern

- Reconcile Loop
  - Observe
  - Diff
  - Act
- Just like the controllers in control plane



CRD - With vs. Without

	With CRDs (Writing Operator)	Without CRDs (Using Aggregated API)
Programming	Users can choose any language for a CRD controller.	Requires programming and building binary and image.
Additional Service	No additional service to run; CRDs are handled by API server.	An additional service to create and that could fail.
Integration	No ongoing support once the CRD is created. Any bug fixes are picked up as part of normal Kubernetes Master upgrades.	May need to periodically pickup bug fixes from upstream and rebuild and update the Aggregated API server.
Versioning	No need to handle multiple versions of your API	You need to handle multiple versions of your API; for example, when developing an extension to share with the world.

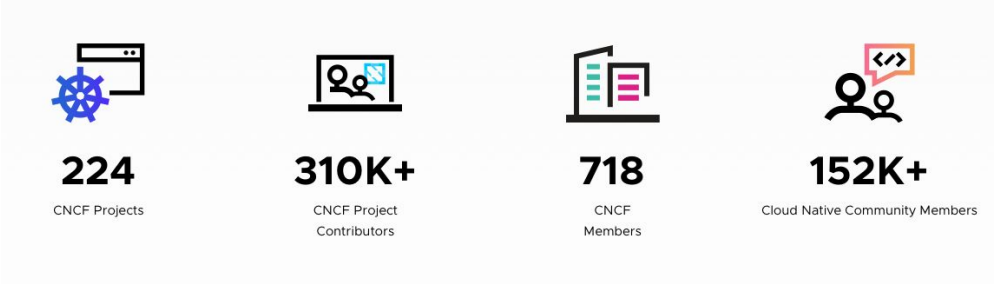
Appendix: CRD - Operator Pattern Example

1. A custom resource named SampleDB, that you can configure into the cluster.
2. A Deployment that makes sure a Pod is running that contains the controller part of the operator.
3. A container image of the operator code.
4. Controller code that queries the control plane to find out what SampleDB resources are configured.
5. The core of the operator is code to tell the API server how to make reality match the configured resources.
    a. If you add a new SampleDB, the operator sets up PersistentVolumeClaims to provide durable database storage, a StatefulSet to run SampleDB and a Job to handle initial configuration.
    b. If you delete it, the operator takes a snapshot, then makes sure that the StatefulSet and Volumes are also removed.
6. The operator also manages regular database backups. For each SampleDB resource, the operator determines when to create a Pod that can connect to the database and take backups. These Pods would rely on a ConfigMap and / or a Secret that has database connection details and credentials.
7. Because the operator aims to provide robust automation for the resource it manages, there would be additional supporting code. For this example, code checks to see if the database is running an old version and, if so, creates Job objects that upgrade it for you.

CNCF

CNCF - Intro

- The Cloud Native Computing Foundation (CNCF) hosts critical components of the global technology infrastructure.
- It is under Linux Foundation.



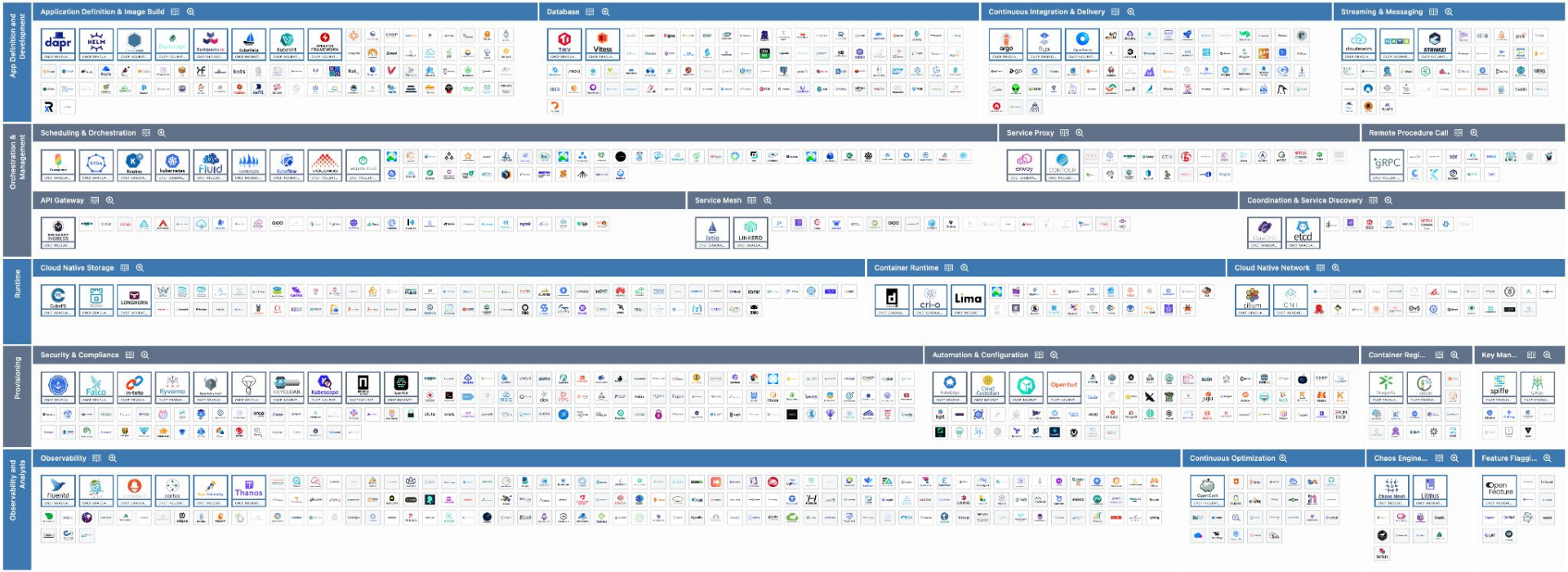
CLOUD NATIVE COMPUTING FOUNDATION

CNCF - Projects

- Prometheus
- Thanos
- OpenSearch
- Vector
- Jaeger
- Grafana

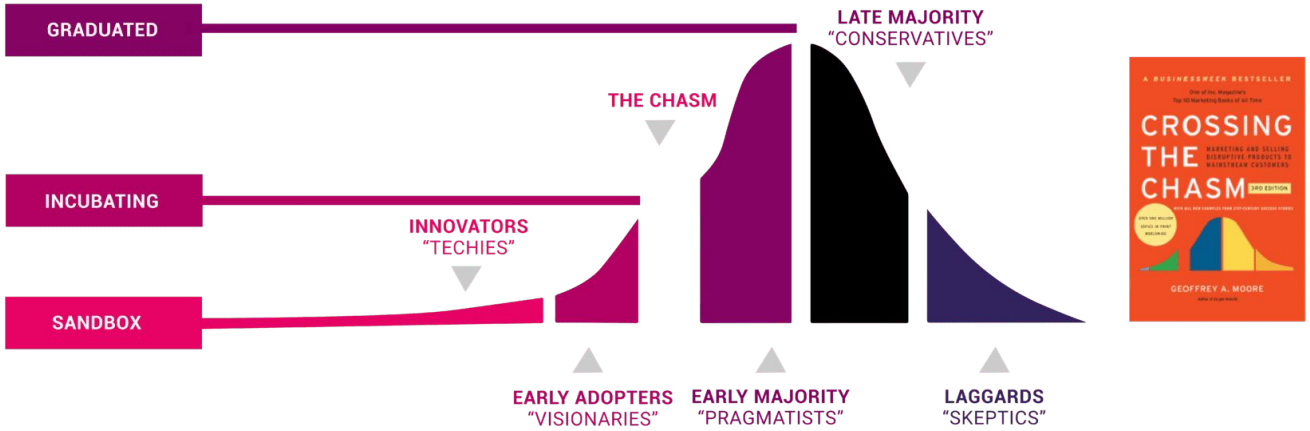
CNCF - Landscape

CNCF Landscape



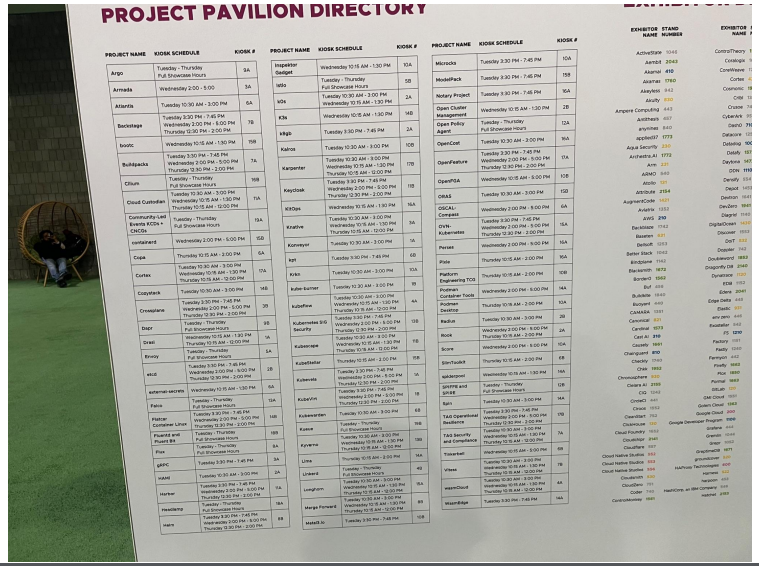
CNCF - Project Status

- Sandbox
- Incubating
- Graduated (35)
- Archived



CNCF - KubeCon

Four days of collaboration, learning, and innovation to drive the future of cloud native computing.



Reference

- https://docs.docker.com/
- https://opencontainers.org/
- https://d7y.io/
- https://prometheus.io/
- https://thanos.io/
- https://opensearch.org/
- https://vector.dev/
- https://www.jaegertracing.io/
- https://grafana.com/
- https://jsonnet.org/
- https://cncf.io/
- https://microsoft.github.io/
- https://awsmorocco.com/thanos-deep-dive-addressing-prometheus-limitations-at-scale-84a6b02b01bc?gi=c4519d136b4c