Real-World CKA

CKA

Domains	Description
Cluster Architecture, Installation, and Configuration	All things infrastructure. How does Kubernetes ACTUALLY work?
Workload Scheduling	Figuring out how Kubernetes has the ability to run workloads
Services and Networking	Kubernetes is nothing without networking. It's how the entire platform works.
Storage	Think hard drives or storage that isn't ephemeral
Troubleshooting	Understanding how to fix issues from the cluster to the workloads to the networks.

The Real World	A cert cannot fully prepare you for the real world. Let's dive into what you need.

CKA vs CKAD

- Infra focused
- Environment configurations at the platform level

- Developer focused
- Environment configuration at the application level

CKA vs KCNA

- Infra focused
- Environment configurations at the platform level

- Same focus as the CKA, just at a lower level

CKA vs CKS

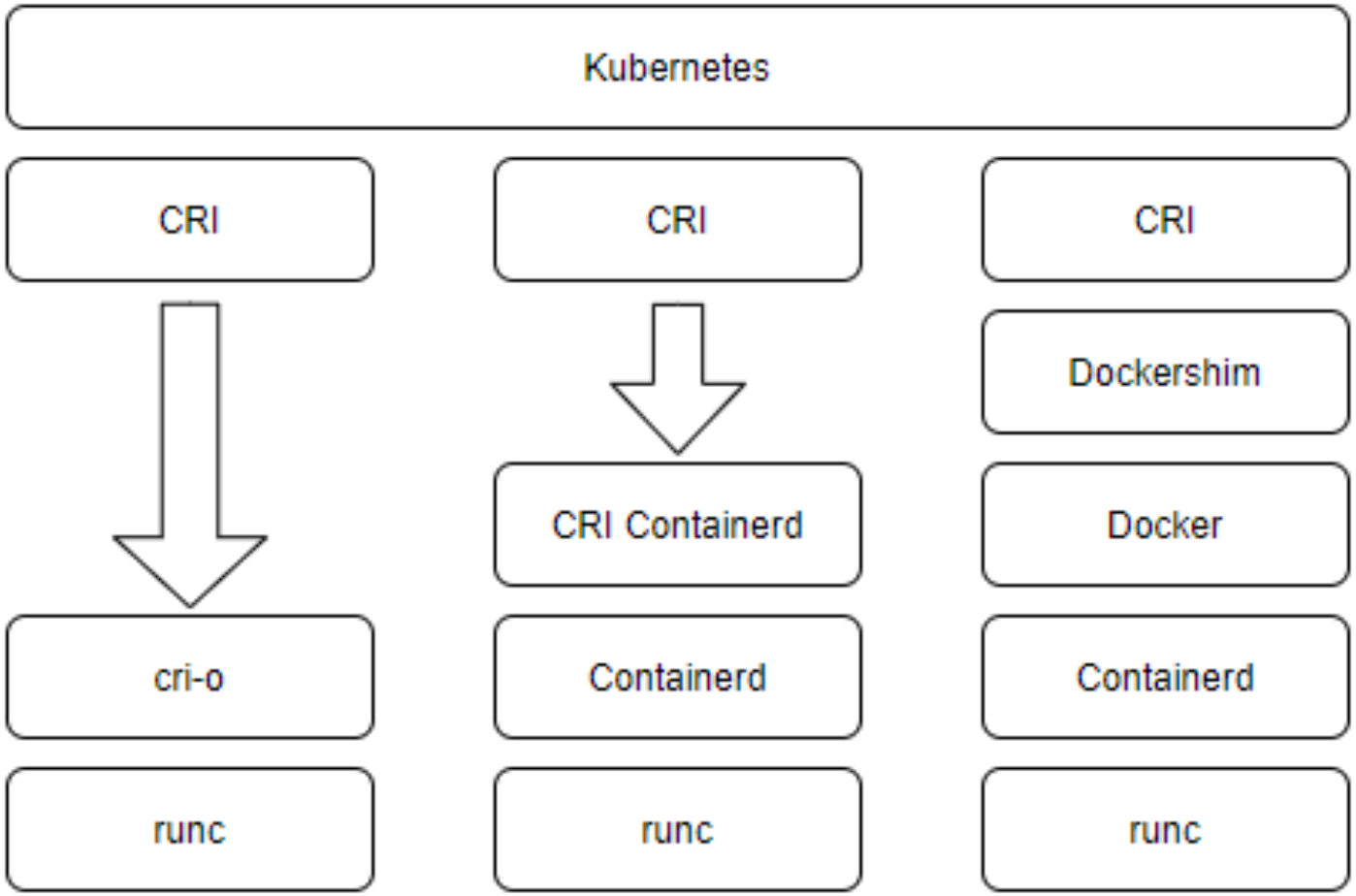
- Infra focused
- Environment configurations at the platform level

- Everything and anything at the security level

CRI-O Runtime

- CRI Compliant
- Solid out-of-the-box implementation for Kubeadm
- Can be used in the cloud

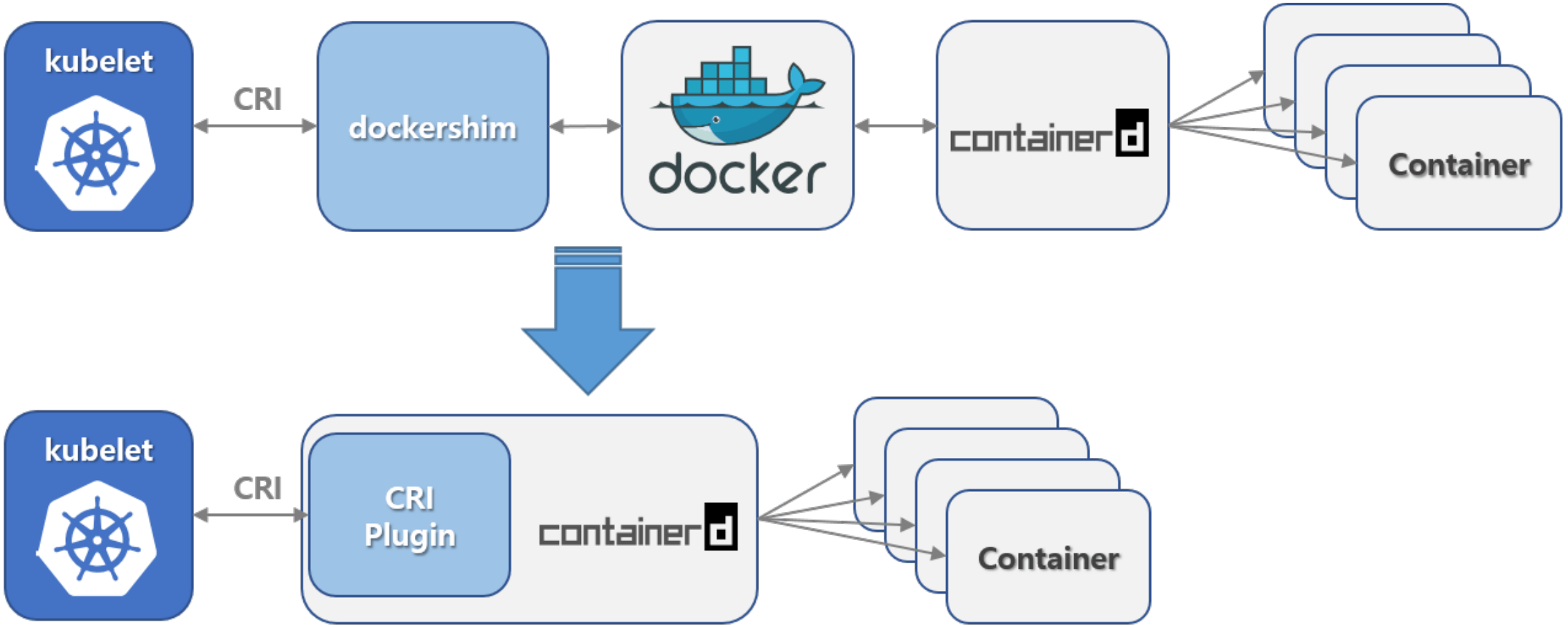
CRI-O Runtime



Containerd Runtime

- Graduated CNCF Project
- Available as a Daemon for Windows and Linux
- OCI Compliant
- Default for AKS and EKS

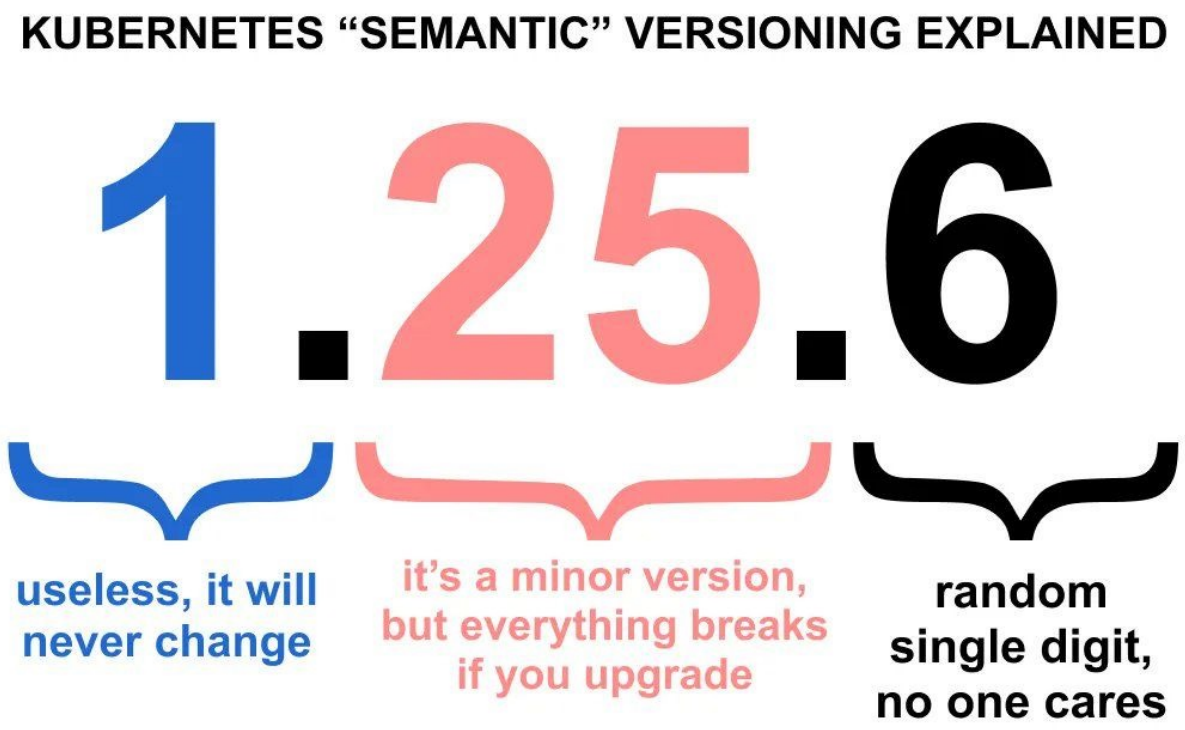
Containerd Runtime



Kubernetes Version Information

- Kubernetes versions are expressed as x.y.z, where x is the major version, y is the minor version, and z is the patch version, following Semantic Versioning terminology.
- The Kubernetes project maintains release branches for the most recent three minor releases (For example - 1.28, 1.27, 1.26).
- Kubernetes 1.19 and newer receive approximately 1 year of patch support.
- Kubernetes 1.18 and older received approximately 9 months of patch support.

Versioning


KUBERNETES “SEMANTIC” VERSIONING EXPLAINED
1.25.6
useless, it will never change
it’s a minor version, but everything breaks if you upgrade
random single digit, no one cares

DNS

- The Domain Name System is a hierarchical and distributed naming system for computers, services, and other resources in the Internet or other Internet Protocol networks.

CoreDNS

The Kubernetes default

eBPF

Remove the need for ip-tables, which means better scalability

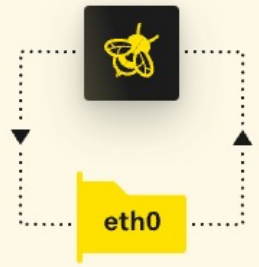
Far better kernel efficiency as it's JIT compiled and runs directly in the kernel

Networking, observability, tracing, and security out of the box

Used by organizations such as Google and Netflix

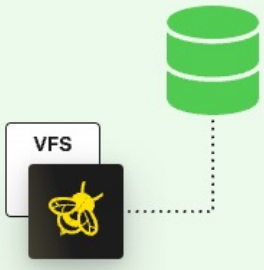
eBPF

What's possible with eBPF?



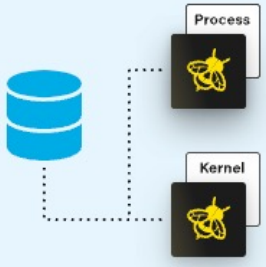
Networking

Speed packet processing without leaving kernel space. Add additional protocol parsers and easily program any forwarding logic to meet changing requirements.



Observability

Collection and in-kernel aggregation of custom metrics with generation of visibility events and data structures from a wide range of possible sources without having to export samples.



Tracing & Profiling

Attach eBPF programs to trace points as well as kernel and user application probe points giving powerful introspection abilities and unique insights to troubleshoot system performance problems.

Security

Combine seeing and understanding all system calls with a packet and socket-level view of all networking to create security systems operating on more context with a better level of control.

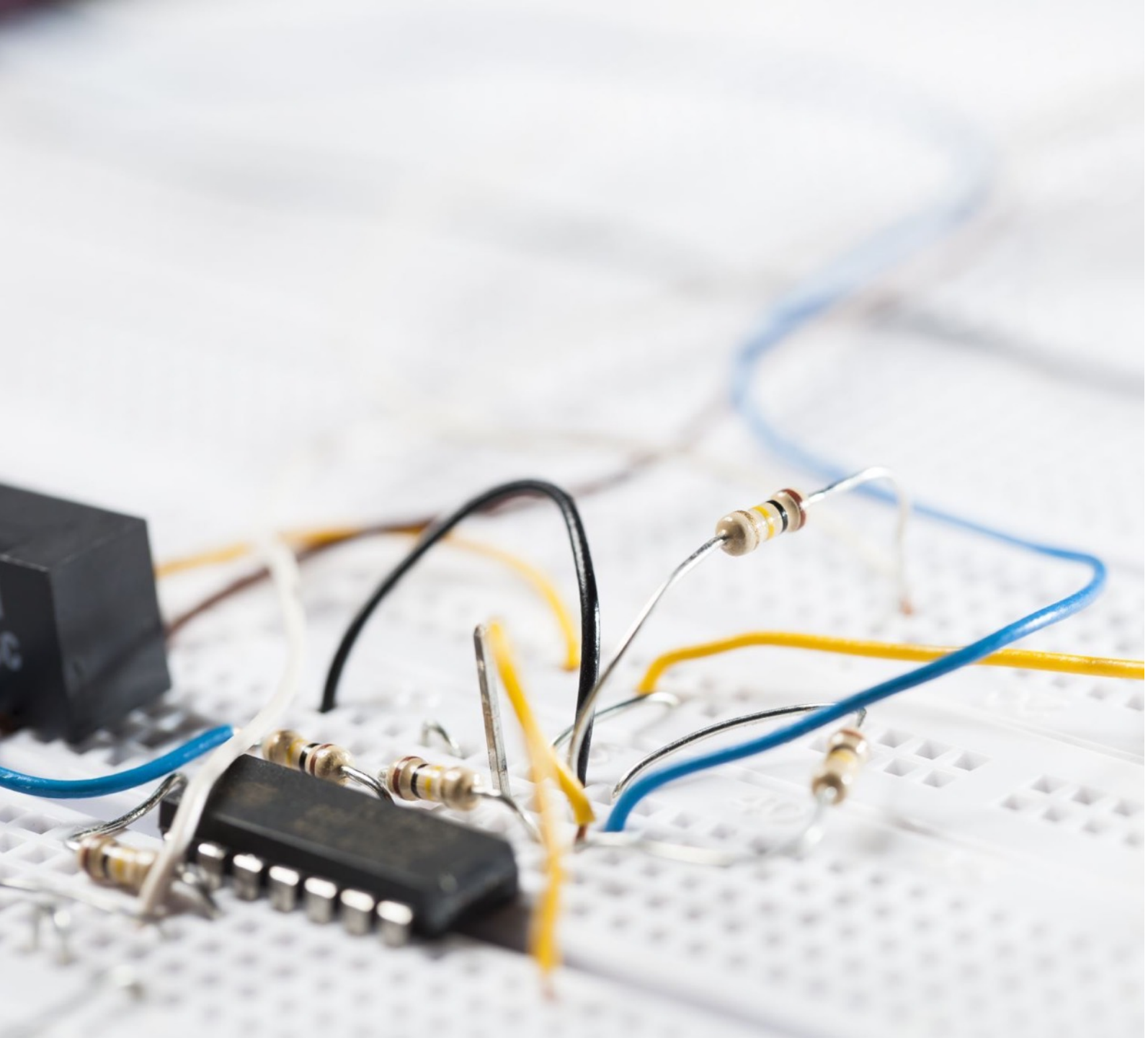
Source: https://ebpf.io/static/observability-0e7ded4a90e87ea6a804f95a26f0ca71.svg

Pods

- A group of one or more containers
- If there are more than one container in a Pod, it’s called a “sidecar container”
- Smallest deployable unit

Deployment

- Manages a set of Pods
- Is considered a ”high level Controller”
- Has the ability for self healing
- Deployments do not manage the Replicas/Pods, ReplicaSets do



DaemonSet

- Like the Deployment Controller, it's a high-level Controller
- Default behavior is to ensure that a Pod is running on each Worker Node
- Purposes include any time that you need a Pod to run on each worker node. For example, if there's an agent running that's needed to run on each node
- You can specify which nodes you want the Pods to run on with Labels.

StatefulSet

- A higher-level Controller like the Deployment and DaemonSet
- Ensures that a Pod is unique with it's own unique identifier
- Maintains the sticky identity for each Pod
- StatefulSets are used for apps that require unique network identifiers, stable persistent storage, graceful deployments/scaling, and rolling updates in an ordered fashion

Custom Resource Definition (CRD)

- Extends the Kubernetes API
- Allows you to have your very own Objects/Kind
- A config that’s not enabled by default
- You can create your own or use what already exists

Controllers

Ensures that the current state is the desired state

3 replicas? That means you should have 3 replicas

Reconciliation loop

Most k8s objects contain a Controller

Operators

Controllers + CRD’s == Operators

Kubebuilder

Operator SDK Framework

Authentication and authorization

Authentication == logging in.

Authorization == permissions/RBAC

RBAC Verb List

Get	View individual resources like a Pod
List	Retrieve a list of resources. Example – view all Deployments in a Namespace
Watch	Watch changes to resources in real-time
Create	Create a new resource
Update	Modify an existing resource
Patch	Make partial updates to a resource
Delete	Remove resources
DeleteCollection	Delete a collection of resources

Backups

Etcd

volumes

Options

Velero

Veeam

S3 Backups

Azure Backups

Core API Group vs Named API Group

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: projectx_policy
  namespace: default
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-web
```

Declarative

- Tell me what to do, not HOW to do it

Imperative

- Tell me what to do and HOW to do it

Imperative and Declarative

```
spec:
  containers:
  - name: nginxdeployment
    image: nginx:latest
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
    ports:
    - containerPort: 80
```

Declarative

- A list of things you want done
- Code that focuses on specifying the result of what you want

```
OS=xUbuntu_20.04
VERSION=1.22

echo "deb https://download.opensuse.org/repo
echo "deb http://download.opensuse.org/repos

curl -L https://download.opensuse.org/reposi
curl -L https://download.opensuse.org/reposi

exit

sudo apt update -y
sudo apt install cri-o cri-o-runc -y

sudo systemctl daemon-reload
sudo systemctl enable crio --now
```

Imperative

- A list of things you want AND how to do them
- Focused on an explicit sequence of commands to describe how you want a computer to do things

Resource Utilization

Namespaces

Namespaces

In Kubernetes, *namespaces* provides a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces. Namespace-based scoping is applicable only for namespaced objects *(e.g. Deployments, Services, etc)* and not for cluster-wide objects *(e.g. StorageClass, Nodes, PersistentVolumes, etc)*.

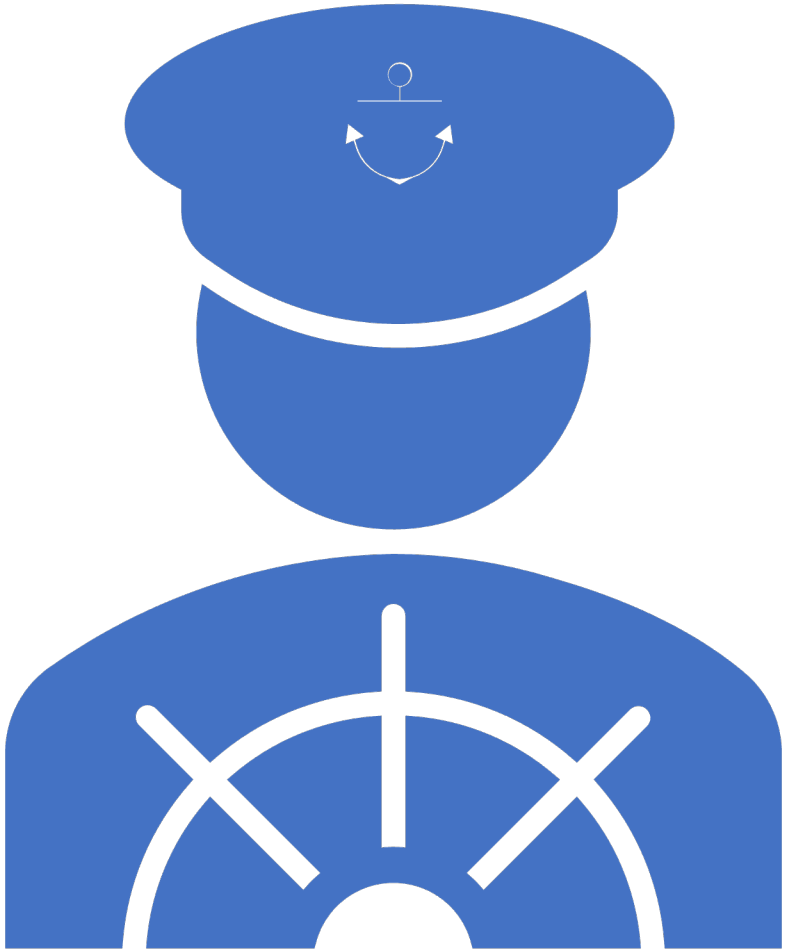
Secrets

Secrets

A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in a container image. Using a Secret means that you don't need to include confidential data in your application code.

Because Secrets can be created independently of the Pods that use them, there is less risk of the Secret (and its data) being exposed during the workflow of creating, viewing, and editing Pods. Kubernetes, and applications that run in your cluster, can also take additional precautions with Secrets, such as avoiding writing sensitive data to nonvolatile storage.

Secrets are similar to ConfigMaps but are specifically intended to hold confidential data.



Helm

- Package Manager (think Aptitude)
- Package up Kubernetes Manifests

Workloads and Scheduling

1

Deployments

2

Upgrades

3

Resource optimization and scaling

What To Log

AUTHENTICATION

CONTROL PLANE LOGS

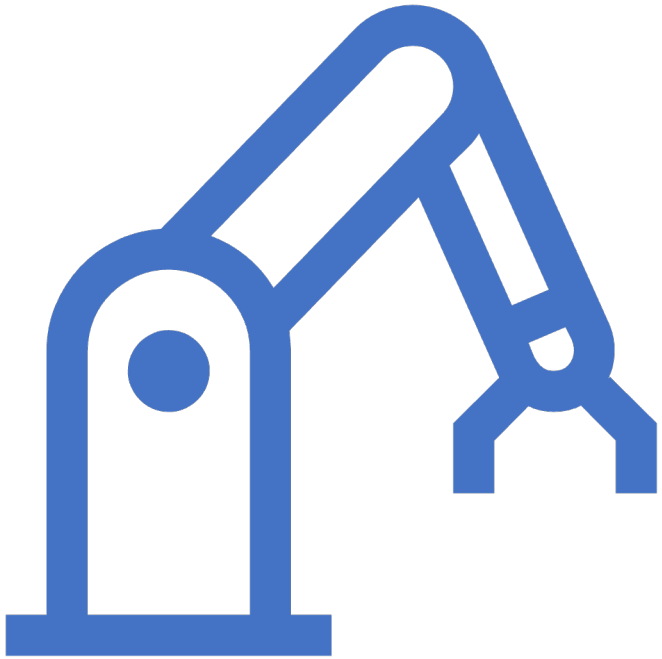
What To Log For Worker Nodes

Network logs from kube-proxy, eBPF, or CNI

Kubelet

Logging Options For Pods

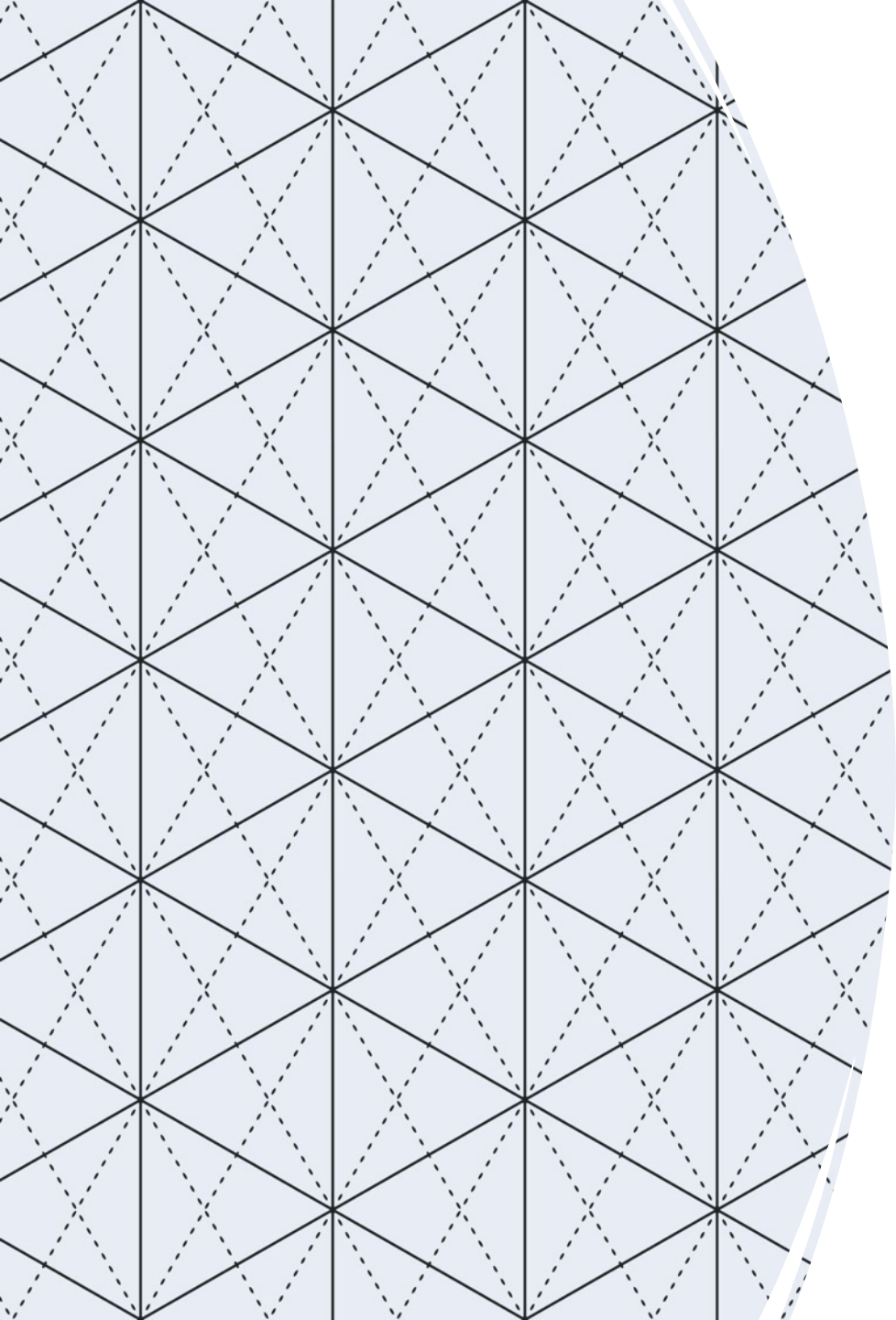
- Stdout and Stderr
- Using Application-level Logging Configuration
- Using Logging Agents



Commands

kubectl logs pod_name

kubectl describe *resource* resource_name



Monitoring

- Seeing the data in real-time
- Having the ability to go back 30/60/90/Etc. days to see what happened
- Alerting capabilities

Observability

- Doing something with the data


Metrics
Collecting time series data, which is used to predict expected ranges and forecast values, showing it in dashboards (like Grafana or another UI-centric dashboard), and alerting on it.
170
160
150
140
130
120
110
Sun
Mon
Tue
Wed
Thu
Fri
Sat

Traces

The health of an application from an end-to-end perspective. It’s the process of investigating how events take place between microservices.

Logs
- Aggregating and storing logged event messages written by programs and systems

Policy Enforcement

- OPA
- Kyverno
- Admissions Controllers

OPA

Gatekeeper

OPA Implementation

01

Config = Definition of what Gatekeeper is allowed to create policies for

02

Constraint Template = Rule/policy that you want to configure for your environment

03

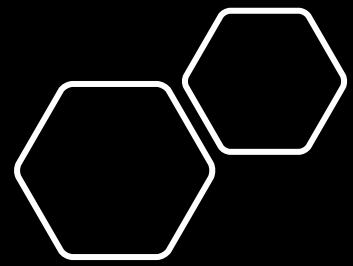
Constraint Config = Bringing the constraint to life (implementing it)

Pod vs Container Logging

Kubernetes captures logs from each container within a Pod

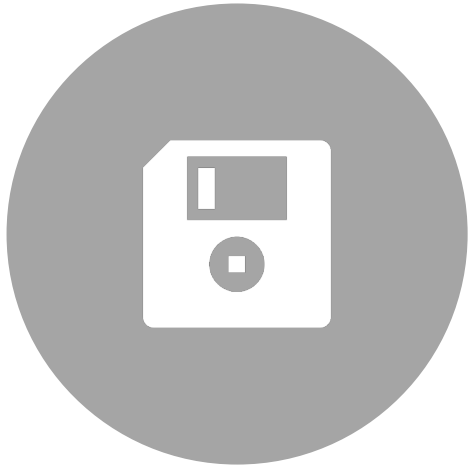
You can see logs per Pod

Those logs can be (and should be) exported



Etcd Backup

IN THE CLOUD, THERE ARE SPECIFIC SERVICES



ON PREM, YOU WOULD HAVE TO CHOOSE A BACKUP SOLUTION

Troubleshooting On-Prem and Cloud Networks

1

Ensure that the host networking is working as expected

2

Ensure kube-proxy (or eBPF) is working

3

Ensure the CNI is working

Container Logging

Logging

Stderr

1

Typically used to output error messages

Stdoutt

2

Command normal output

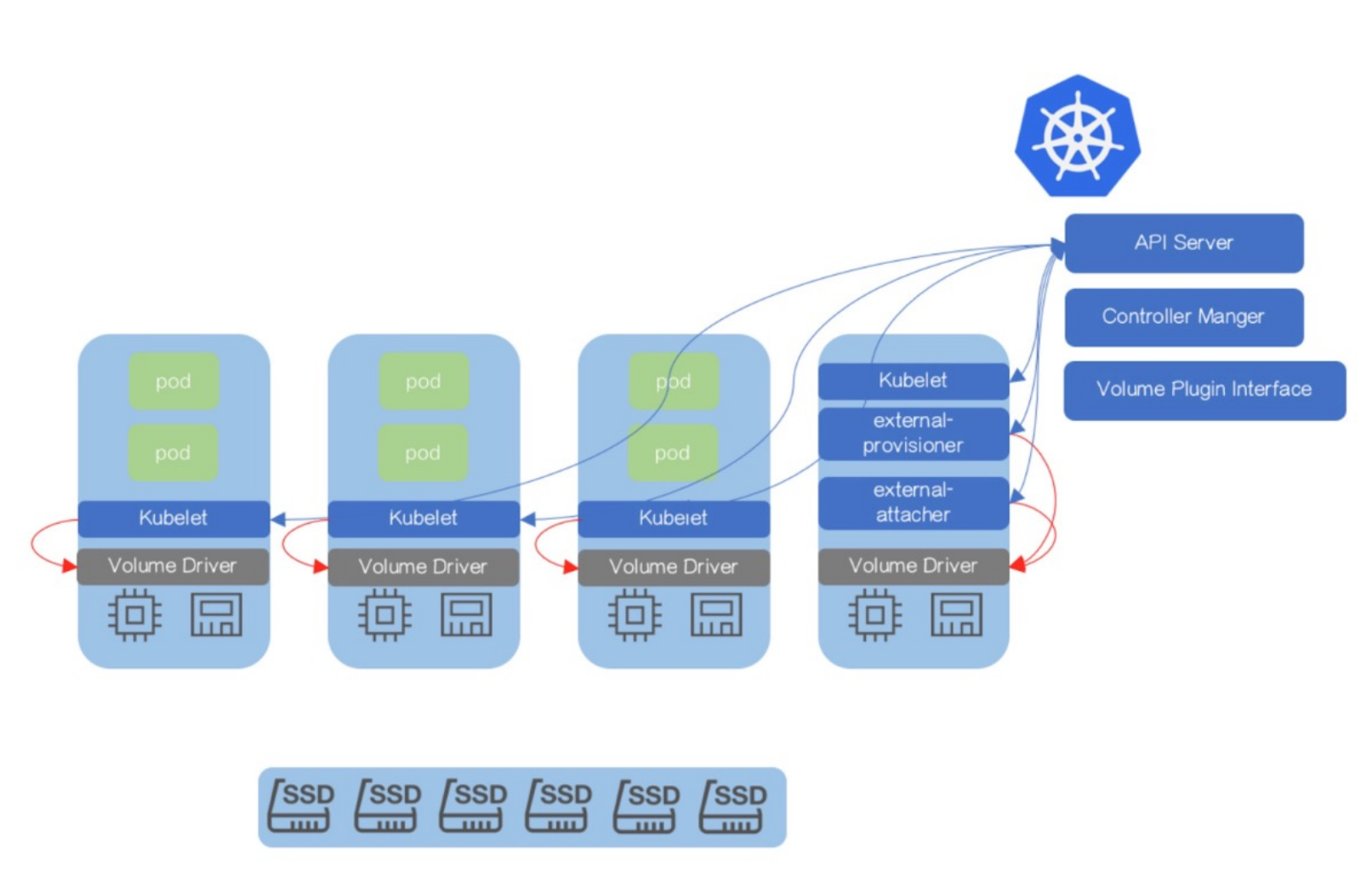
CSI

A way to use storage (like a hard drive) on Kubernetes

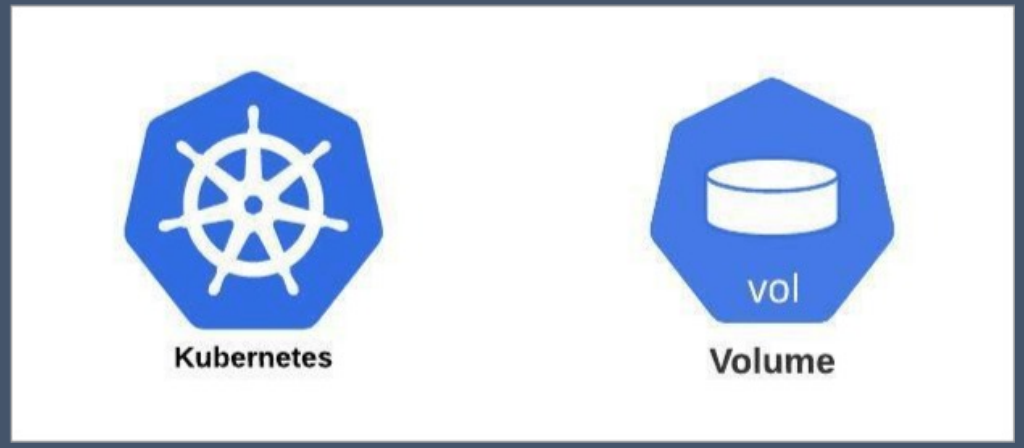
CSI Breakdown

- Gives Kubernetes the ability to use storage, which is called Volumes.
- Various storage interfaces available
- It’s a plugin much like the CNI and CRI

CSI Breakdown



Volume Reclaim Policies



Retain

- If a user deletes a PersistentVolumeClaim, the corresponding PersistentVolume will not be deleted
- It is moved to the Released phase, where all of its data can be manually recovered.

Recycle

- This policy is deprecated
- **Performs a scrub on the volume and makes it available for re-use**

Delete

- Default action for dynamically provisioned PersistentVolumes
- **A dynamically provisioned volume is automatically deleted when a user deletes the corresponding PersistentVolumeClaim**
- **This automatic behavior might be no good if the volume contains data that you need**

Kubernetes Secrets

- Stored in Etcd
- **Stored as Base64 Plain Text**

What You Should Know

1. Linux
2. Networking
3. Storage
4. Lite programming/automation
5. Application support
6. Infrastructure/scaling

Home Lab With Cash

Synology DS223 Diskstation NAS

Intel NUC 12 Extreme Kit NUC12DCMi7

DigitalCloud TRAINING

Home Lab With Less Cash

Intel NUC 9 Business Mini PC Desktop

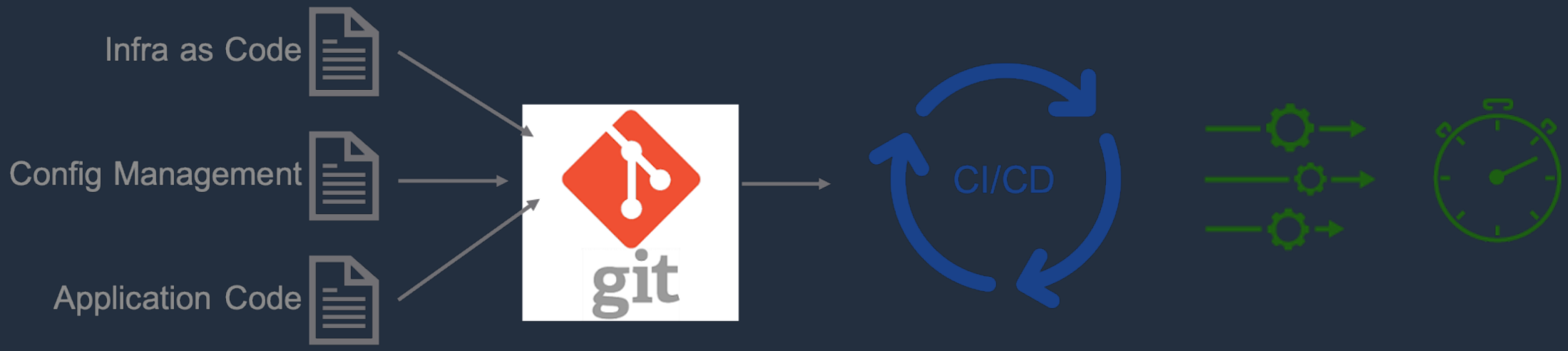
Synology DS120j 1 bay NAS

DigitalCloud TRAINING

GitOps

- Deploy our code automatically
- Confirm current state is the desired state
- Git-based (like the name suggests)

GitOps-in-a-nutshell



"Source of Truth" for declarative code

Update to code source triggers a pipeline

Pipeline runs a series of tasks, resulting in the update of the runtime environment to match the source

Tools

- Have there been any tools requested by teams?
- Do you know the business value of what you're implementing?
- Can the engineers in-house get familiar with the tools in a timely fashion?
- What impact would bringing on a new tool create?

Container Network Interface (CNI)

- Dynamically configure network resources
- A framework/plugin to make Networking a separate entity for Kubernetes resources

Cluster Networking

Outside of the CNI

Kubernetes Service

- “frontend” of Pods
- A way to access Pods in a more production-level fashion
- Definition: A Service is a method for exposing a network application that is running as one or more Pods in your cluster
- Services get DNS Names, which come from CoreDNS

LoadBalancer vs ClusterIP vs NodePort

1

LoadBalancer

Creates a service that has a load balancer attached to it

2

ClusterIP

Default type. Used to create a connection using the clusters IP and a port. Used for communication within the cluster

3

NodePort

Exposes a port on each node. Like a ClusterIP, but used for communication within and outside of the cluster

Network Policies

- Firewall rules for Pods
- Controls Ingress and Egress traffic

If you want to control traffic flow at the IP address or port level (OSI layer 3 or 4), then you might consider using Kubernetes NetworkPolicies for particular applications in your cluster. NetworkPolicies are an application-centric construct which allow you to specify how a pod is allowed to communicate with various network "entities" (we use the word "entity" here to avoid overloading the more common terms such as "endpoints" and "services", which have specific Kubernetes connotations) over the network. NetworkPolicies apply to a connection with a pod on one or both ends, and are not relevant to other connections.

Ingress and Egress

Traffic in

Traffic out

Kubernetes Controllers

Confirm that the current state is the desired state

Ingress Controller