Kubernetes Cluster Operations

Version 2.0.1

About Strigo

- Strigo is a web-based platform that provides the classroom environment for our courses.
- Let's walk through the features of the platform.
- We'll also get your lab environment initialized

Introductions

- About your instructor(s)

- Tell us about yourself
  - Would you classify yourself as a
    - developer
    - systems administrator
    - architect
    - It depends on the day/hour
  - What are your goals for learning and adopting Kubernetes?

Agenda

1. Cluster Architecture
2. Logging
3. Monitoring
4. Cluster Troubleshooting
5. Onboarding Applications and Teams
6. Cluster Maintenance

Course Format

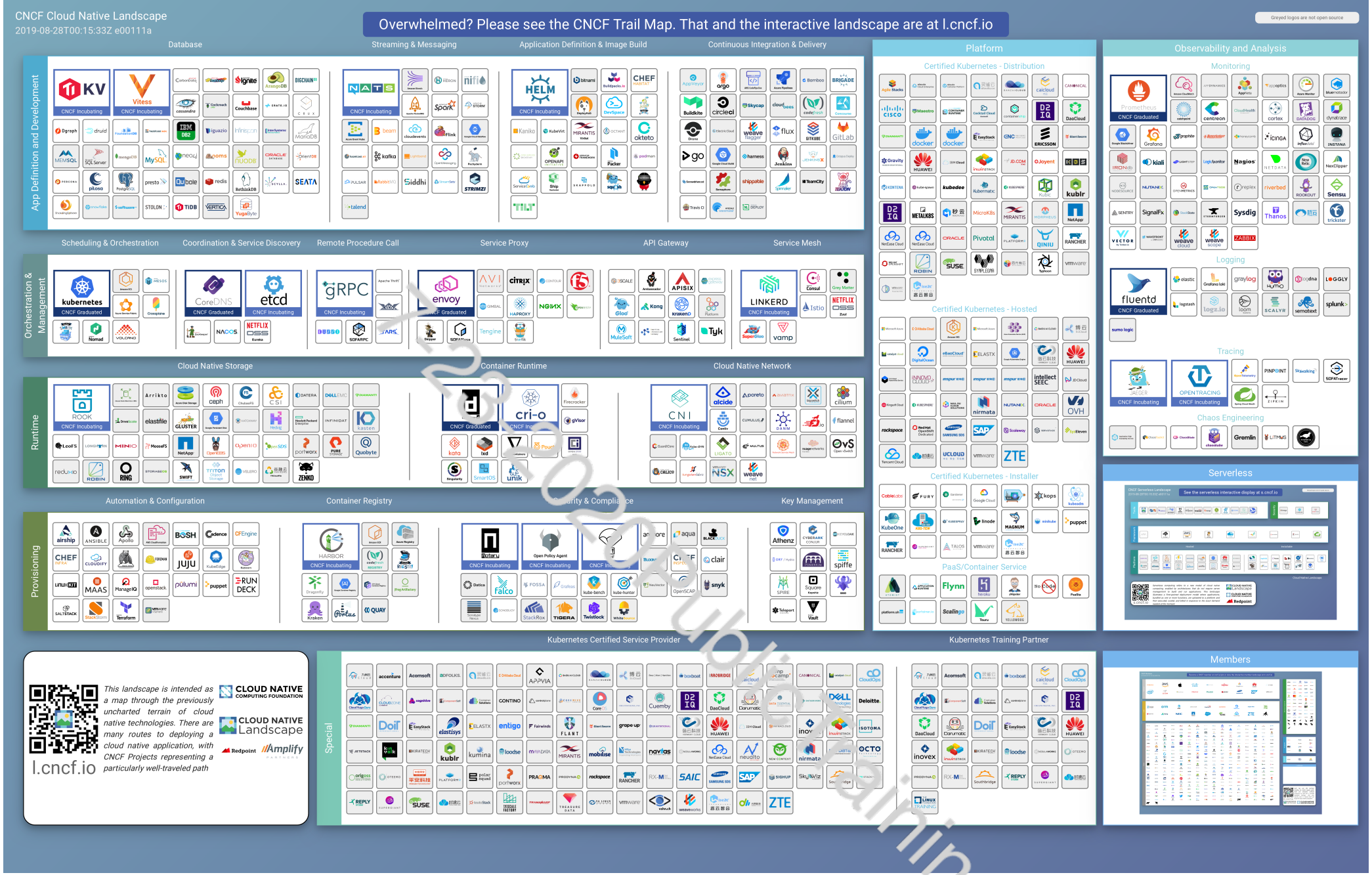
- This is a lab-intensive, hands-on course
- Each section will begin with the introduction of a new concept
- Each section contains a lab exercise where you will explore each new concept

Cluster Architecture

Chapter 01

Agenda

1. Cluster Architecture
2. Logging
3. Monitoring
4. Cluster Troubleshooting
5. Onboarding Applications and Teams
6. Cluster Maintenance



Ecosystem

- Kubernetes ecosystem is large
- Quick overview to help navigate
- Future course in progress will dive deep in these areas

Distributions and Service Offerings

Distributions

- Upstream Kubernetes
  - access to latest versions
  - highest level of interoperability

- Distributions
  - based on upstream but with additions
  - user experience tends to be the same
  - Installation/support/setup differs
  - vendor lock-in considerations
  - convenience at a cost

RANCHER

Red Hat OpenShift

Service Offerings

- Benefits
  - managed and maintained
  - turnkey
- Considerations
  - patches / fixes / versions
  - upgrade control
  - extensibility (ie. admission controllers)

Azure Kubernetes Service (AKS)

Amazon EKS

Google Kubernetes Engine

Storage

Storage – Supported PV types

- GCEPersistentDisk
- AWSElasticBlockStore
- AzureFile
- AzureDisk
- FC (Fibre Channel)
- Flexvolume
- Flocker
- NFS
- iSCSI
- RBD (Ceph Block Device)
- CephFS
- Cinder (OpenStack block storage)
- Glusterfs
- VsphereVolume
- Quobyte Volumes
- Portworx Volumes
- ScaleIO Volumes
- StorageOS

Storage – Container Storage Interface (CSI)

- Out of tree development
  - vendor can add storage support without committing to Kubernetes
- Drivers
  - https://kubernetes-csi.github.io/docs/drivers.html

Storage – Local Persistent Volumes

- Use Cases
  - applications that provide data replication at software layer
  - ie. MongoDB, Cassandra, Elasticsearch
- Provisioning
  - manual – admin creates PV at node creation time
  - static provisioner – provisioner makes PVs for each directory
- Considerations
  - pods are pinned to nodes
  - error scenarios are still manually handled

Networking

Networking - Considerations

- Kubernetes provides basic networking framework
- Plan addressing ahead of time
  - choose IP ranges carefully to not conflict with your datacenter
  - difficult or not possible to change after cluster is running
- Choose your CNI provider…

Networking – Container Network Interface (CNI)

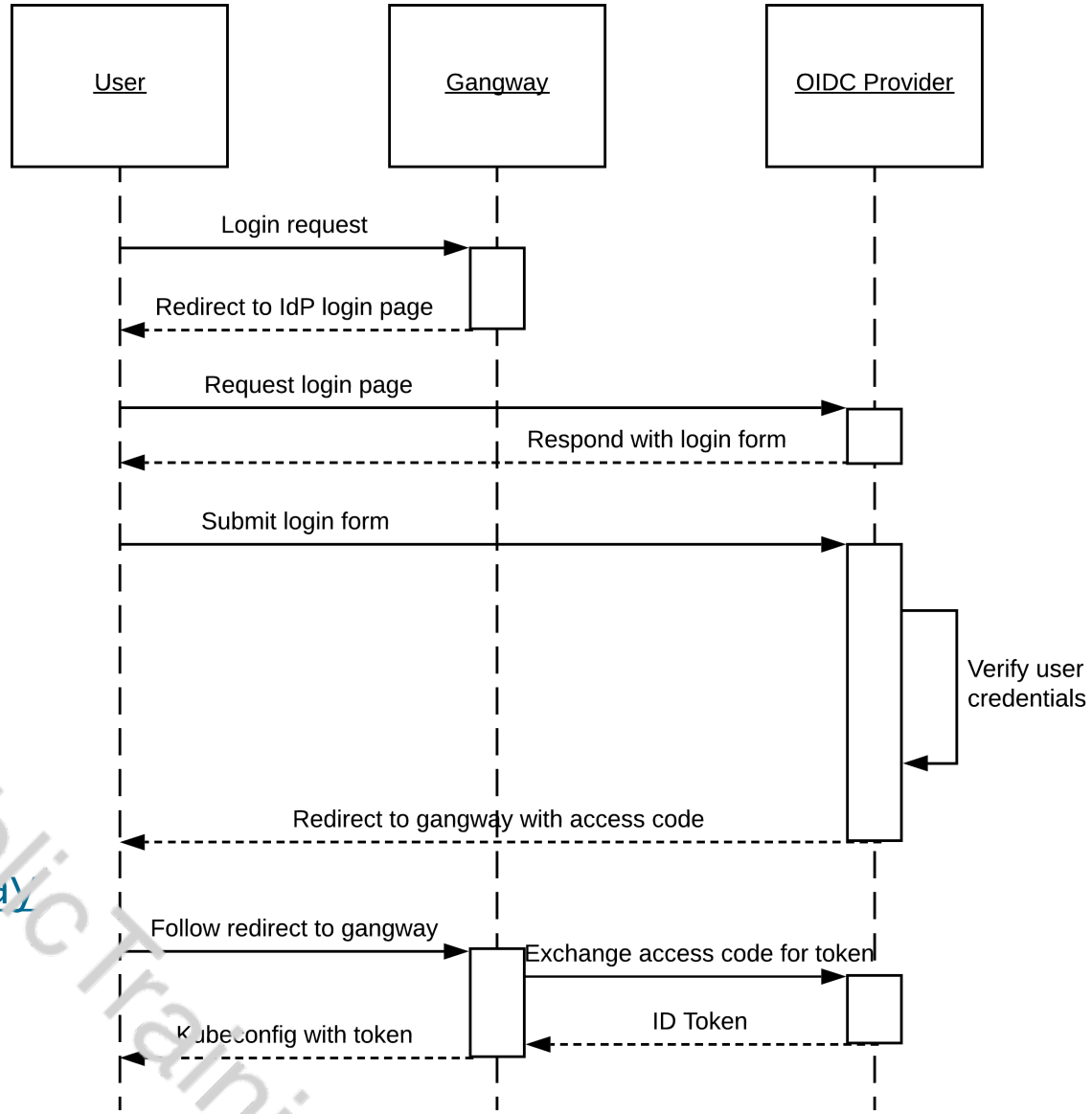
- Interface for third party providers
  - integrate external network stack without committing to Kubernetes

- Example CNI providers
  - Project Calico
  - NSX-T
  - Flannel
  - Weave

External Authentication

- Users are not stored in Kubernetes

- Typical Integration
  - **dex** – OpenID Connect provider
    - https://github.com/dexidp/dex
  - **gangway** -
    - https://github.com/heptiolabs/gangway



kubeadm

- Bootstrap minimum viable cluster
  - control plane creation/setup
  - joining nodes
  - certificate creation/management
  - initial cluster-admin account setup
- Nongoals
  - provisioning of machines
  - Installing nice-to-have addons

Lab 01

Build a Kubernetes cluster

Logging

Chapter 02

Agenda

1. Cluster Architecture
2. Logging
3. Monitoring
4. Cluster Troubleshooting
5. Onboarding Applications and Teams
6. Cluster Maintenance

Logging - requirements

- What are the kinds of logs we want to capture?

Logging - requirements

- What are the kinds of logs we want to capture?
  - container logs
  - host OS logs
  - control plane logs
  - event messages

Logging – Container Logs

- What information should each log entry have?

Logging – Container Logs

- What information should each log entry have?
  - timestamp
  - log message

Logging – Container Logs

- What information should each log entry have?
  - timestamp
  - log message

  - namespace name
  - pod name
  - container name
  - labels
  - image
  - node name

Logging – Container Logs

- Ultimately want to be able to query and logs like…
  - Show all logs from all pods with label X=Y
  - Show all logs from pods on node X
  - Show all logs from nodes with label X
  - Show all logs from container X on pods Y
  - Show all logs from namespace X

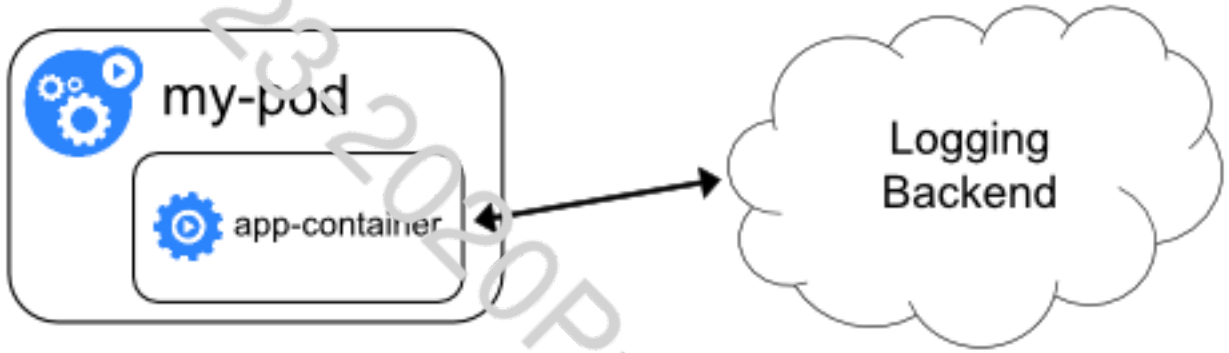
Logging in Kubernetes and Containers

- Containers ideally log to stdout/stderr
- stdout/stderr logs are found in /var/log/containers
- helpful commands
  - kubectl logs <pod_name> [-f] [-c <container_name>]
  - kubectl exec -it <pod_name> [-c <container_name>] <command>

App Logging - architectures

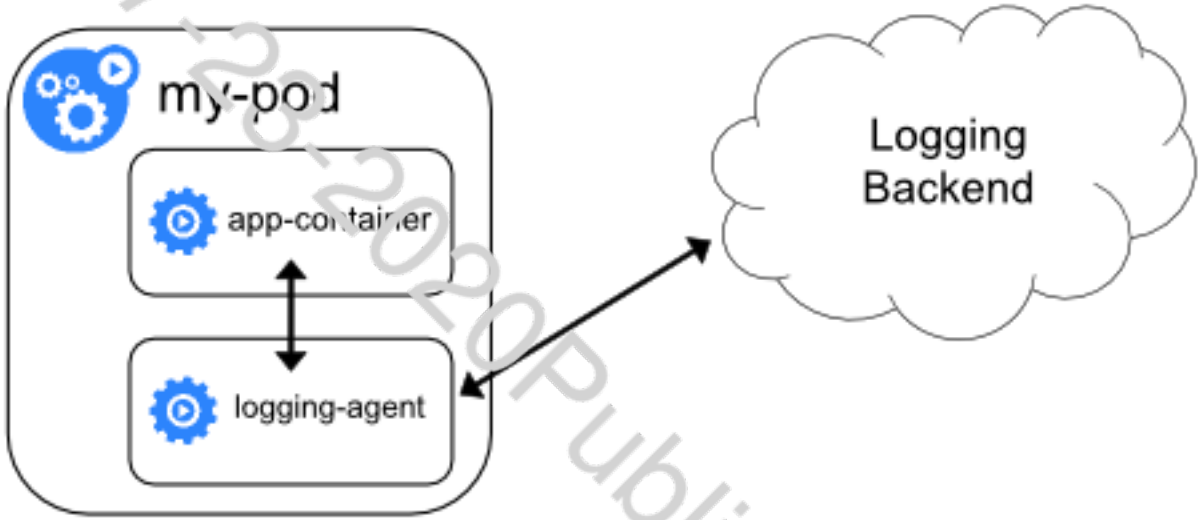
Application -> Logging Backend

- Application has built-in logic to log to backend



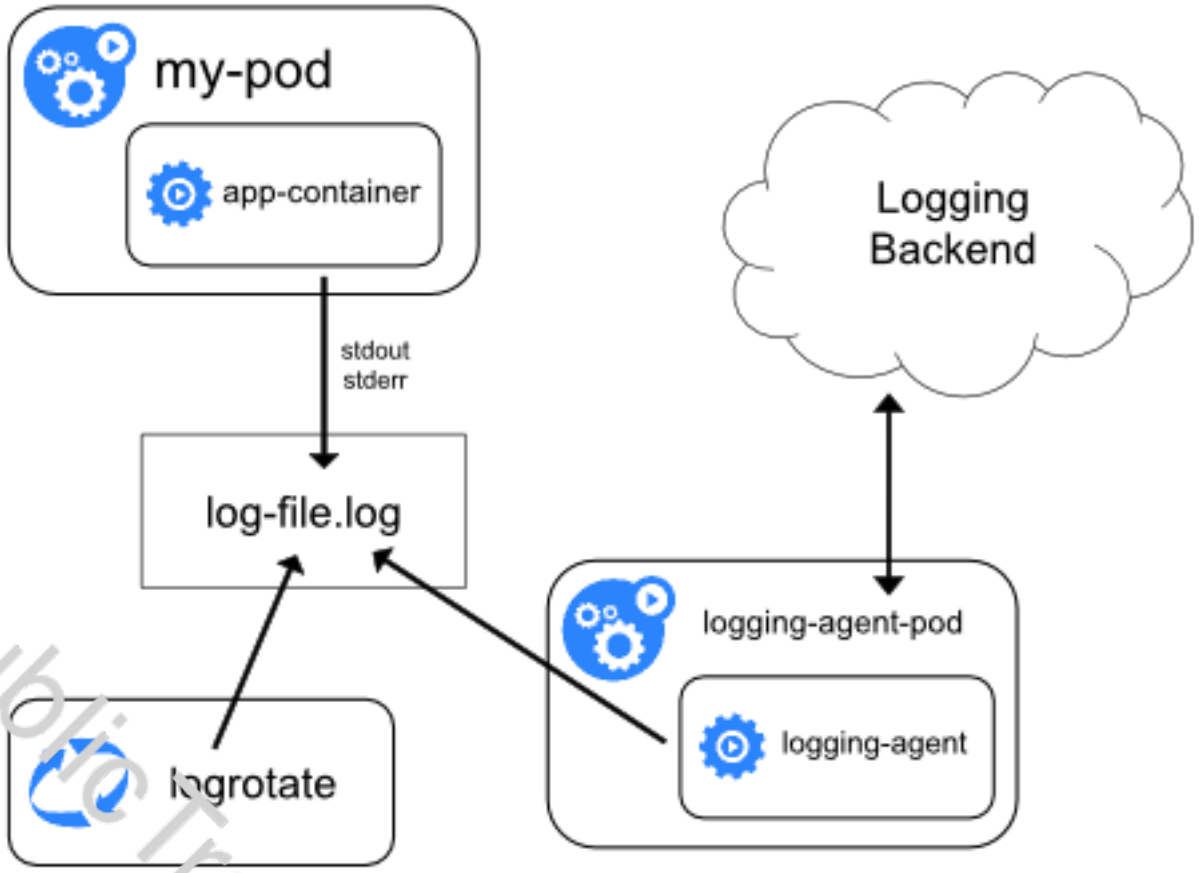
Sidecar Container -> Logging Backend

- Logging agent built-in to deployment of application
  - second container in the same pod as the application



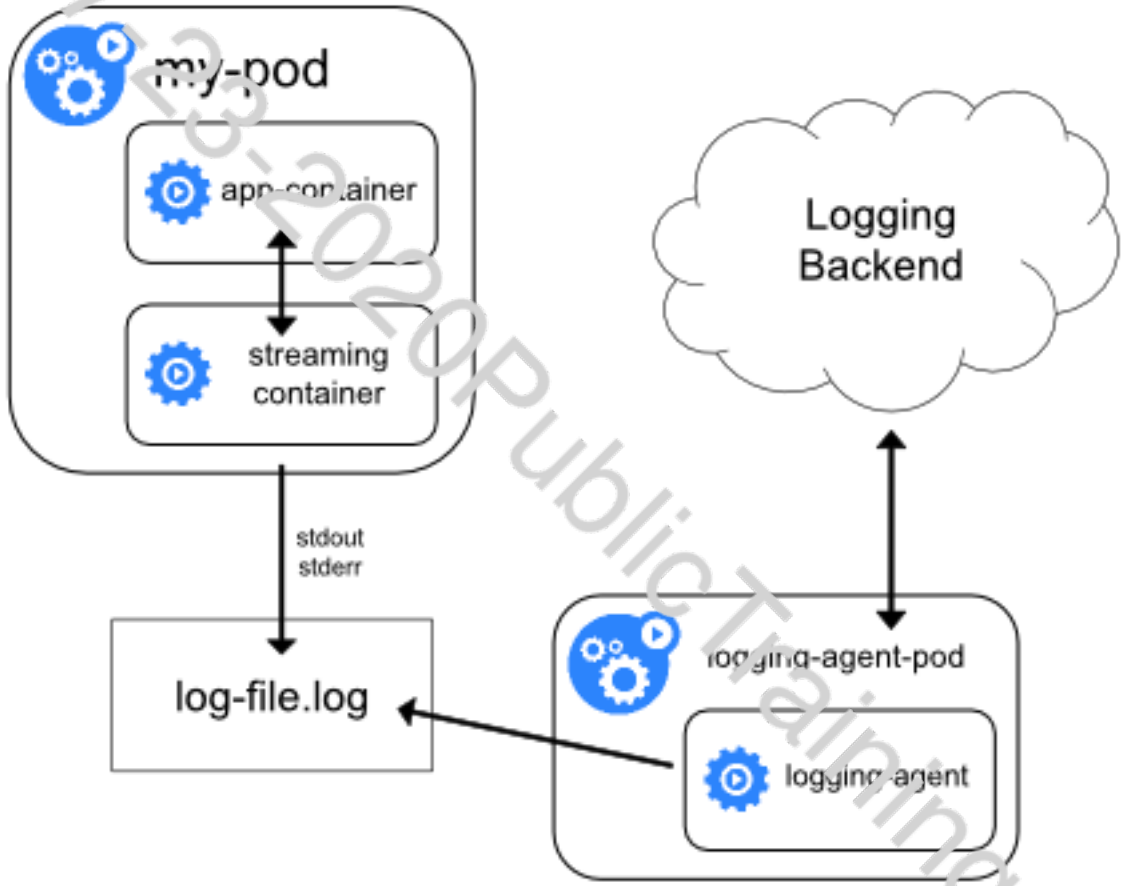
Node Logging Agent

- Run single agent on node
  - Leverage DaemonSet
- Ideal solution if possible
  - only works if all containers log via stdout/stderr

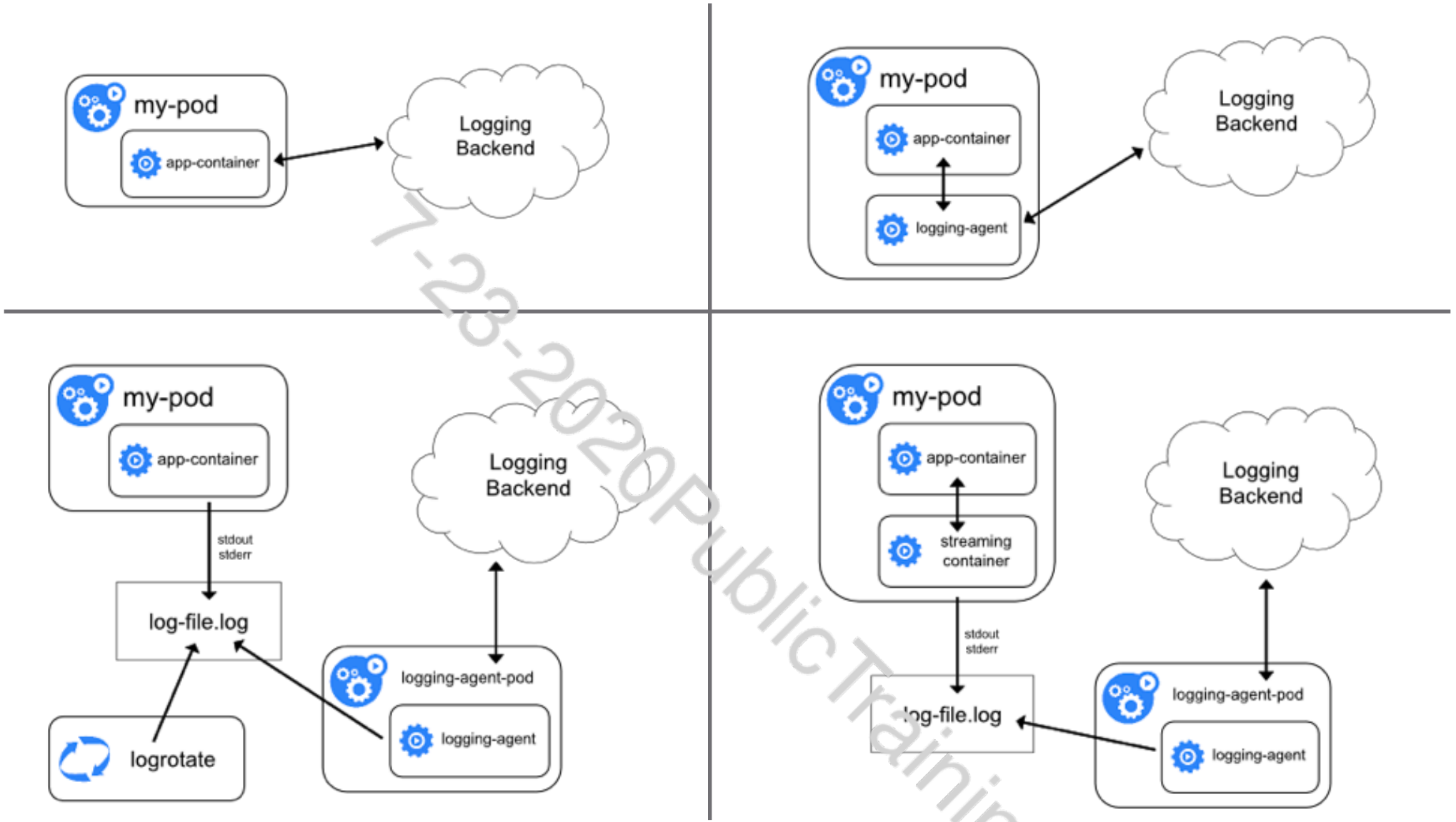


Sidecar Container + Node Agent

- Run N sidecars for how many logs there are
  - streaming container tails logs and sends to stdout



Which one to use?



Logging Backend

- Where should your logging backend live?

Logging Backend

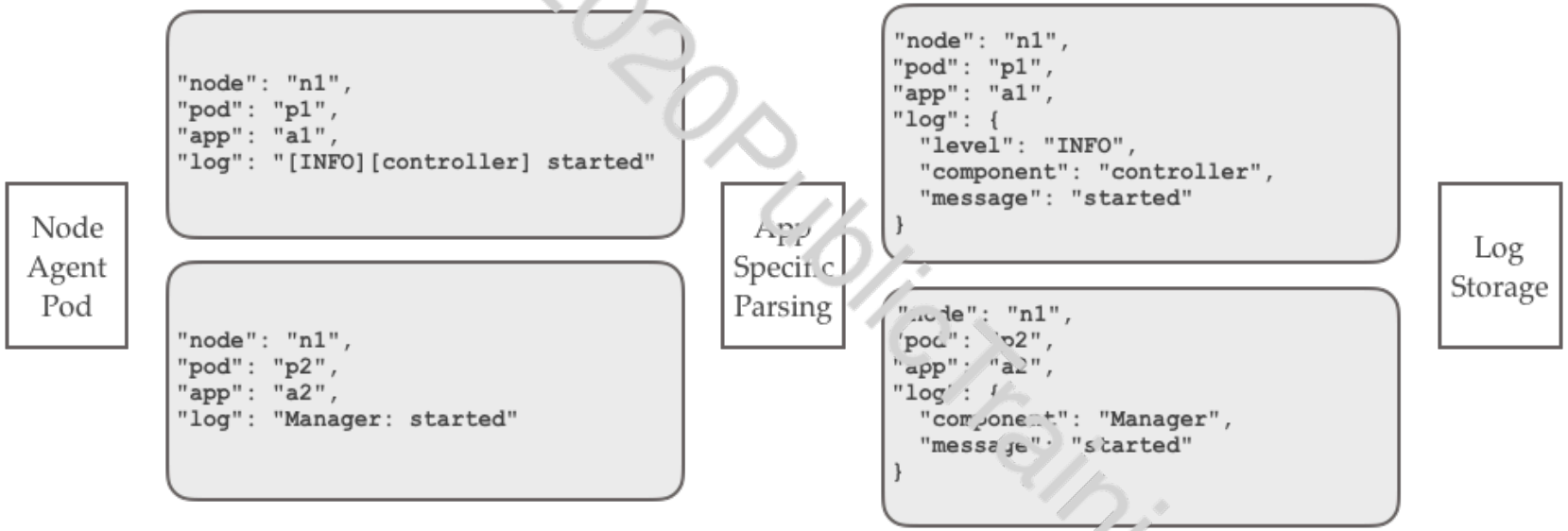
- Where should your logging backend live?
  - Outside of the cluster
    - Standalone
    - On a different Kubernetes cluster
  - Inside of the cluster
    - Dedicated namespace
  - Third party service

Direct Agent Shipping Concerns

- What problem arises with node agents sending logs directly to logging backend?

Direct Agent Shipping Concern – Middle Layer

- What problem arises with node agents sending logs directly to logging backend?
  - node agent is generic and log format agnostic
  - middle layer allows ability to further parse logs for each app specific format



Direct Agent Shipping Concerns – Filebeat Hints

- Filebeat - log agent/shipper with *hints* feature
  - allows agent to be dynamically configured with parse patterns
  - configuration is placed in PodSpec putting control with developers
  - https://www.elastic.co/blog/docker-and-kubernetes-hints-based-autodiscover-with-beats

```
metadata:
  annotations:
    co.elastic.logs/multiline.pattern: '^\['
    co.elastic.logs/multiline.negate: true
    co.elastic.logs/multiline.match: after
    co.elastic.logs.sidecar/exclude_lines: '^DBG'
```

System Component Logging

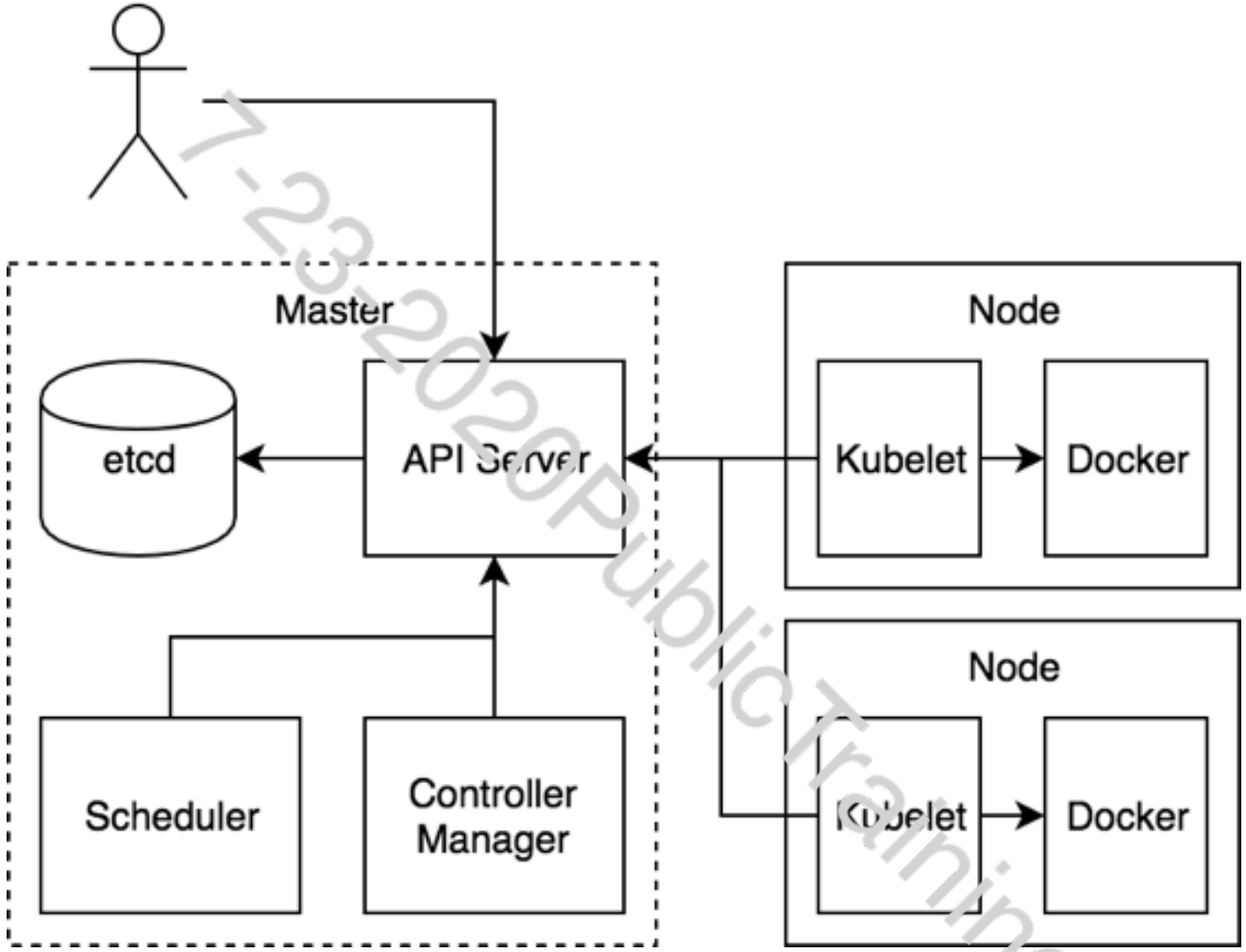
- Components not running as pods
  - Kubelet
  - container engine
  - potentially etcd
- Use your favorite agent and add to your machine setup
- Log rotation
  - if installed with kube-up.sh then logrotate added automatically
  - if not, then add log rotation yourself manually

Event Messages

- Characteristics
  - Written by system components and stored in etcd
  - Removed after 60 minutes by default
  - *kubectl get events*
- Stream copies into external system
  - enable log term storage and reports to be run
  - Example solutions
    - eventrouter
    - metricbeat

Audit Log

- Track all user and system component access



Audit Log

- API server configuration
  - --audit-log-path
  - --audit-policy-file

- Ideally setup log agent and ship to logging backend
  - enables log term reporting and auditor requirements
  - create alerts when certain activity detected

Lab 02

Deploy a cluster logging solution

Monitoring

Chapter 03

Agenda

1. Cluster Architecture
2. Logging
3. Monitoring
4. Cluster Troubleshooting
5. Onboarding Applications and Teams
6. Cluster Maintenance

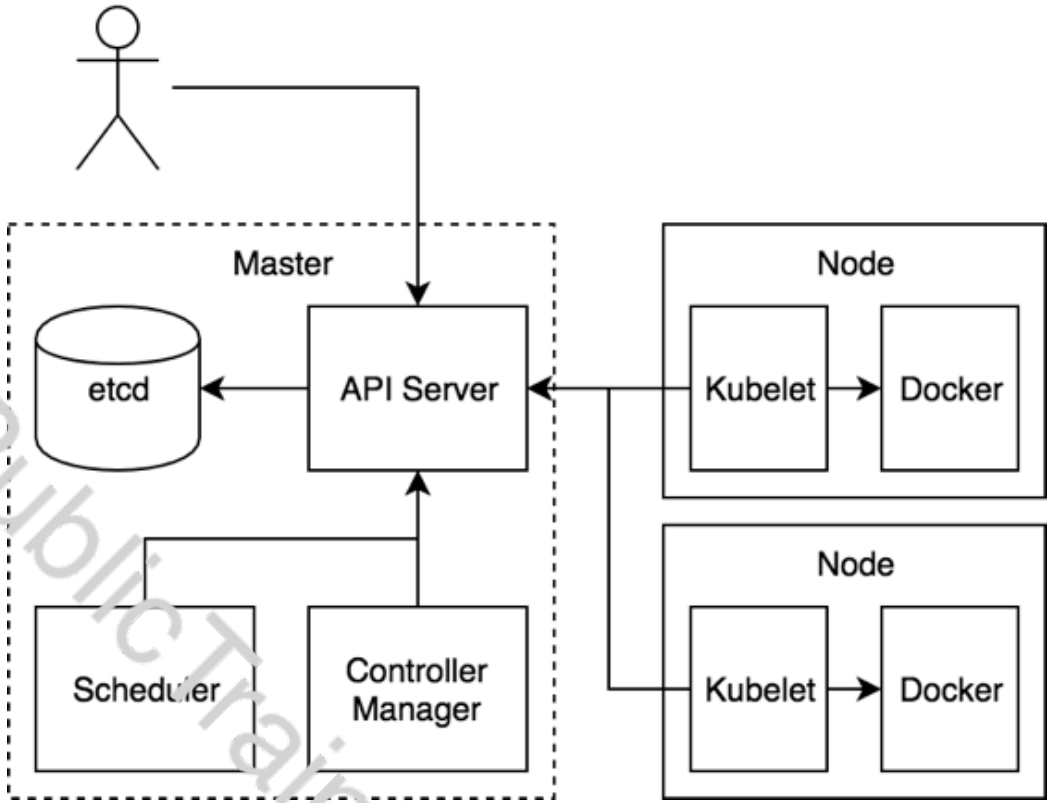
Monitoring - Goals

- Determine normal vs abnormal
  - definition normal will potentially be different from cluster to cluster
- Alerting vs Monitoring
- Monitoring + Logs

Monitoring - Tooling

Monitoring

- Time interval collection of data/state
  - cluster
  - worker nodes
  - control plane components
  - container engine



Monitoring - Agent Options

- DIY - Free
  - kube-state-metrics – collects metrics from Kubernetes API
  - metricbeat - standard metric monitoring that is kubernetes aware
  - ganglia
  - and more…
- Third Party Services - Paid
  - datadog
  - honeycomb
  - stackdriver with Google Monitoring
  - and more…

Monitoring - Interpreting

- Determine normal vs abnormal
  - definition normal will most likely be different from cluster to cluster
  - absolute values not as important as trends

- Examples
  - average time for deployment of application X
  - average disk utilization of worker nodes, etcd node, etc.
  - average response time for a `kubectl get pods` for example

Alerts - Monitoring + Logs

- Alerts
  - use your favorite alerting tool
  - fine tune with your team over time

- logs + monitoring
  - deployment taking longer than normal + ERROR level logs for the same app
  - more than X 500 response codes + higher than normal load average on nodes

Lab 03

Deploy a cluster monitoring solution

Cluster Troubleshooting

Chapter 04

Agenda

1. Cluster Architecture
2. Logging
3. Monitoring
4. Cluster Troubleshooting
5. Onboarding Applications and Teams
6. Cluster Maintenance

Troubleshooting - Cluster vs Application

- Application
  - pod crashing problems and similar where can be resolved by user
- Cluster
  - resolution requires administrator privileges
  - usually affects multiple applications
- How is the problem discovered?
  - user detected
  - monitoring/alert detected

General Guidance

- Before
  - establish as many alerts and dashboards as possible
- During
  - be familiar with where to look (kubectl and dashboards)
  - if node specific, remove workloads from node and then debug
- After
  - determine if new alerts/dashboards should be created/modified

Determine Impact

Determine Impact

- Master node is completely offline/down

Determine Impact

- Master node is completely offline/down
- Working
  - applications
  - logging platform and searching
- Broken
  - deploying and changing applications
  - recover from a node failure

Determine Impact

- Docker engine unreachable by kubelet

Determine Impact

- Docker engine unreachable by kubelet
- Working
  - all workloads with more than one pod
  - overall cluster still healthy
- Broken
  - any workloads with only one pod that was on the problem node

Determine Impact

- If the kubelet cannot reach Docker, what action(s) should the kubelet take?

Determine Impact

- If the kubelet cannot reach Docker, can the kubernetes safely reschedule the pods?

Lab 04

Cluster Troubleshooting

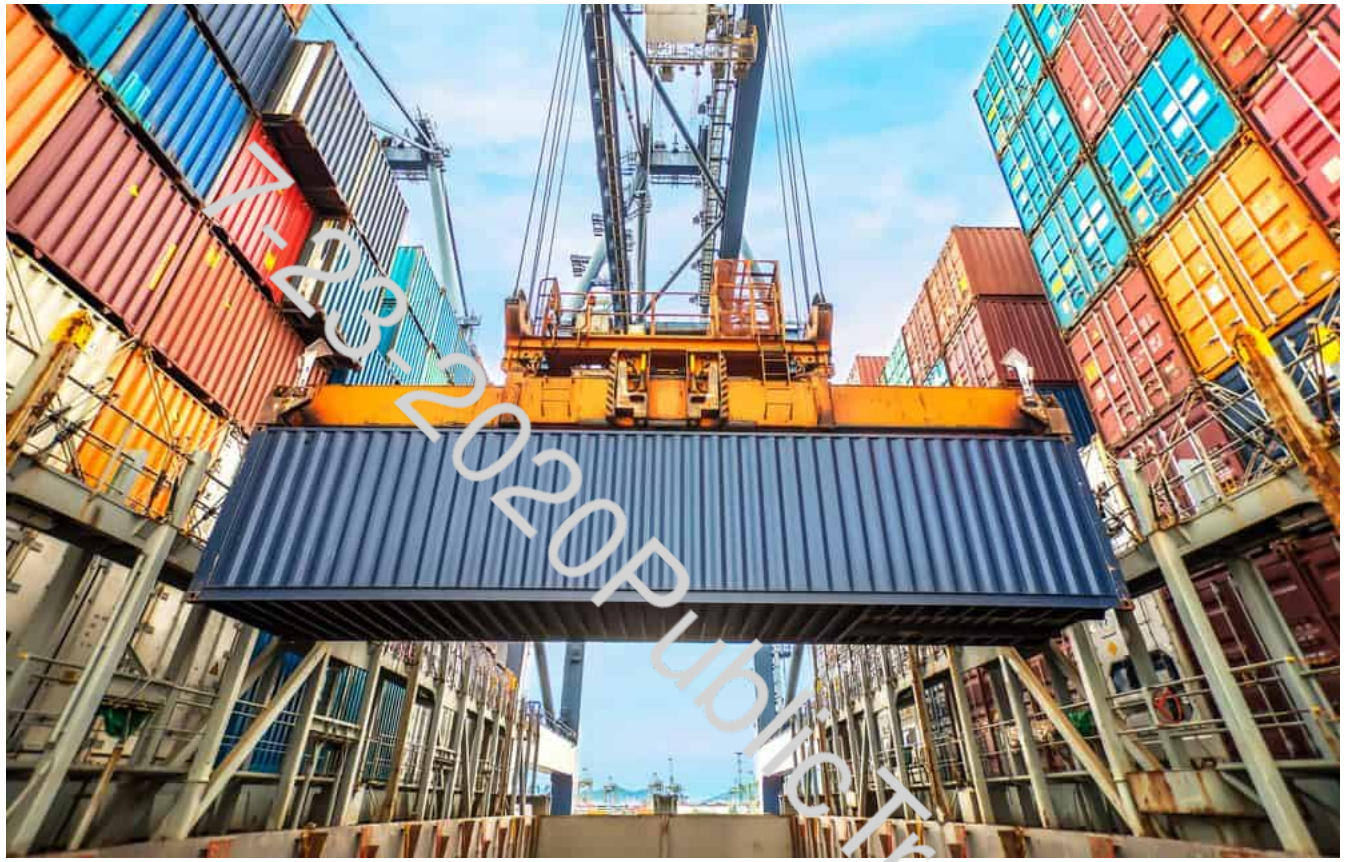
Onboarding Applications and Teams

Chapter 05

Agenda

1. Cluster Architecture
2. Logging
3. Monitoring
4. Cluster Troubleshooting
5. Onboarding Applications and Teams
6. Cluster Maintenance

Onboarding



Onboarding – Teams / People

- What information do we need to collect?

Onboarding – Teams / People

- What information do we need to collect?
  - account creation
  - what groups to be a member of
  - mirror access of existing person
  - do new groups need to be created?

Onboarding – Applications / Projects

- What information do we need to collect?

Onboarding – Applications / Projects

- What information do we need to collect?
  - new or modification to existing
  - network access
  - elevated privileges (ie. run as root, etc.)
  - resource requirements
    - CPU / RAM – min
    - CPU / RAM – max
    - storage
    - number of instances

Mapping to Kubernetes – Application Specification

Goal	Kubernetes
New / Existing application	
Application CPU / RAM min	
Application CPU / RAM max	
Storage	
Escalated privileges	
Number of instances	

Mapping to Kubernetes – Application Specification

Goal	Kubernetes
New / Existing application	Namespace
Application CPU / RAM min	
Application CPU / RAM max	
Storage	
Escalated privileges	
Number of instances	

Mapping to Kubernetes – Application Specification

Goal	Kubernetes
New / Existing application	Namespace
Application CPU / RAM min	Resource request
Application CPU / RAM max	
Storage	
Escalated privileges	
Number of instances	

Mapping to Kubernetes – Application Specification

Goal	Kubernetes
New / Existing application	Namespace
Application CPU / RAM min	Resource request
Application CPU / RAM max	Resource limit
Storage	
Escalated privileges	
Number of instances	

Mapping to Kubernetes – Application Specification

Goal	Kubernetes
New / Existing application	Namespace
Application CPU / RAM min	Resource request
Application CPU / RAM max	Resource limit
Storage	PersistentVolumeClaim (PVC)
Escalated privileges	
Number of instances	

Mapping to Kubernetes – Application Specification

Goal	Kubernetes
New / Existing application	Namespace
Application CPU / RAM min	Resource request
Application CPU / RAM max	Resource limit
Storage	PersistentVolumeClaim (PVC)
Escalated privileges	SecurityContext
Number of instances	

Mapping to Kubernetes – Application Specification

Goal	Kubernetes
New / Existing application	Namespace
Application CPU / RAM min	Resource request
Application CPU / RAM max	Resource limit
Storage	PersistentVolumeClaim (PVC)
Escalated privileges	SecurityContext
Number of instances	Number of Pods

Restriction Mechanisms / Policies

Restricting Resources

Restricting Resources - Resource Quota

- Restrict total resources a namespace can consume
  - compute resources
  - object counts

- When ResourceQuota is specified, requires requests to include specifications

Resource Quota - Compute

- If specified on quota, then required on inbound specs

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    pods: "4"
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

Resource Quota - Counts

- Set max limits for counts

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: object-counts
spec:
  hard:
    configmaps: "10"
    persistentvolumeclaims: "4"
    replicationcontrollers: "20"
    secrets: "10"
    services: "10"
    services.loadbalancers: "2"
```

Restricting Elevated Privileges (Security Context)

Pod Security Policy

- Optional but recommended
- Restrict what users can request in their pods
- Users configure their pod via *SecurityContext* and it is validated against policies

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: my-default-psp
spec:
  privileged: false
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  runAsUser:
    rule: MustRunAsNonRoot
  fsGroup:
    rule: RunAsAny
  volumes:
  - '/tmp'
```

Mapping to Kubernetes – Restriction Mechanisms

Goal	Kubernetes
Network access	
User/Group access permissions	
Namespace CPU max / min	
Namespace RAM max / min	
Namespace number of pods	
Elevated privilege control	

Mapping to Kubernetes – Restriction Mechanisms

Goal	Kubernetes
Network access	NetworkPolicy
User/Group access permissions	
Namespace CPU max / min	
Namespace RAM max / min	
Namespace number of pods	
Elevated privilege control	

Mapping to Kubernetes – Restriction Mechanisms

Goal	Kubernetes
Network access	NetworkPolicy
User/Group access permissions	RoleBinding/ClusterRoleBinding
Namespace CPU max / min	
Namespace RAM max / min	
Namespace number of pods	
Elevated privilege control	

Mapping to Kubernetes – Restriction Mechanisms

Goal	Kubernetes
Network access	NetworkPolicy
User/Group access permissions	RoleBinding/ClusterRoleBinding
Namespace CPU max / min	ResourceQuota
Namespace RAM max / min	ResourceQuota
Namespace number of pods	ResourceQuota
Elevated privilege control	

Mapping to Kubernetes – Restriction Mechanisms

Goal	Kubernetes
Network access	NetworkPolicy
User/Group access permissions	RoleBinding/ClusterRoleBinding
Namespace CPU max / min	ResourceQuota
Namespace RAM max / min	ResourceQuota
Namespace number of pods	ResourceQuota
Elevated privilege control	PodSecurityPolicy

Namespace Templating

- What should be done at namespace creation time?

Namespace Templating

- What should be done at namespace creation time?
  - labels on the namespace
  - NetworkPolicy
  - RoleBinding
  - PodSecurityPolicy permissions
  - ResourceQuota
- Ideally script this or use tool/framework

Lab 05

Onboarding Applications and Teams

Cluster Maintenance

Chapter 06

Agenda

1. Cluster Architecture
2. Logging
3. Monitoring
4. Cluster Troubleshooting
5. Onboarding Applications and Teams
6. Cluster Maintenance

Backups

- What are the things that need backed up?

Backups

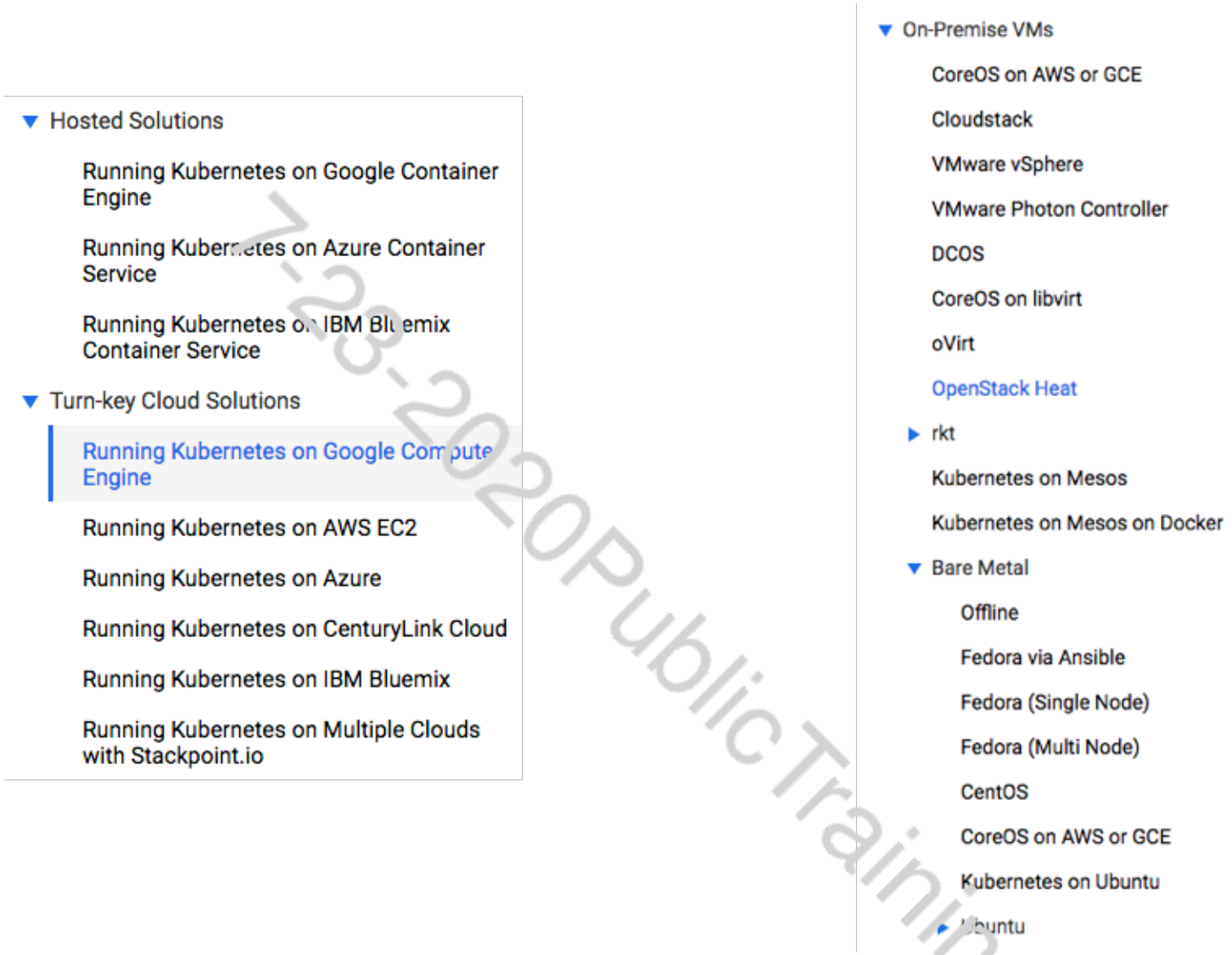
- What are the things that need backed up?
  - Resource definitions (Deployments, Services, ConfigMaps, etc.)
  - Generated Resources (Events, Controller augmented fields)
  - Persistent Volumes
  - Secrets

- How much backup is needed?
  - Stateless vs. Stateful applications
  - Using declarative and all yaml is checked into source control

Backups - Options

- etcd snapshots/exporting
  - requires access to etcd server and is low level/manual
  - restoring is not controlled/modifiable
- ReShifter
  - requires access to etcd server but is more automated and has a UI
- Velero
  - leverages API service only; no dependency upon etcd
  - can run in cluster and at automated intervals
  - output format is JSON based and can be used for other uses

Upgrading



Upgrading – Forklift vs In Place

- In Place
  - traditional approach; leverages existing install and machines
  - if problems arise can be difficult to mitigate

- Forklift
  - create brand new cluster with new version and migrate workloads
  - promotes multi-cluster deployments
  - validation done prior to placing workloads new cluster/version

Upgrading – In Place

- Versioning
  - OK - 1.13 --> 1.14
  - BAD - 1.13 --> 1.15
  - downgrades not possible
- Order of Components
  - upgrade etcd
  - upgrade master node(s)
  - upgrade worker nodes

Node Maintenance

- Reasons for maintenance
  - OS patches/upgrades
  - kubernetes or container engine upgrade/patching
  - hardware replacement
- Mark node as schedulable/unschedulable
  - `kubectl [cordon | uncordon]`
- Evict/drain pods from a node
  - `kubectl drain`

Node Maintenance - Tips

- Leverage node labels
  - racks, OS versions, subnets, etc.
  - all help make using a selector easy for bulk operations

- Build in checks and progress
  - `cordon` first
  - `drain` only one node at time
  - move to next once all deployments have been scheduled

Lab 06

Cluster Maintenance

Conclusion

Thank You