Notes From Kubernetes Training

- Yekta Yazar / AD-EED
- Notes from Alex Ng, Jerry Katzung, Patrick Nisperos, and Zach Domke

- 10/10/2024

Introduction to Kubernetes

What is Kubernetes?

- Kubernetes provides container Orchestration, with the intent to automate controlling the deployment of containers
  - “Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.” – quote from Kubernetes’ documentation website
  - Fun fact: Kubernetes is the helmsman of the container ship (literal definition)
  - Successor/open-sourced version to Borg, an internal tool for running containerized workloads across Google's data centers (early 2000s)

Why do we need Kubernetes?

- Microservices architecture: allow a larger application to be separated into independent parts
  - Very useful when running application through large storage facilities’
  - For example, when you want to run a web store front
- S3DF – Kubernetes is a solution for long up time applications on S3DF

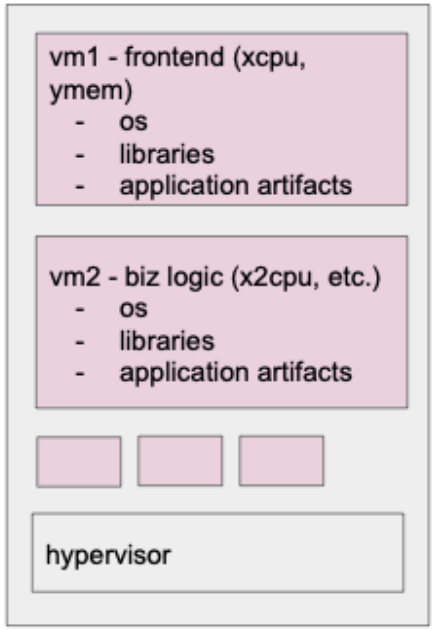
Challenges of using VMs

Couldn't we use Virtual Machines?

- Run multiple OS environments on one computer
- Challenges with VMs, which lead to using containers
  - Using up more resources
  - Slow start up time
  - Single point of failure
  - Scaling
  - Challenging to have automated recovery

Containers:

- Quick to start up, take less resources than VMs
- Still provides some isolation



Challenges

1. Duplication of consumption
   a. Storing the multiple oses takes up disk space
   b. Running multiple oses takes up cpu and mem
2. Boot time for new app server is high
3. SPOF
4. Scaling
   a. Scheduling
5. VM crashes - automate recovery?

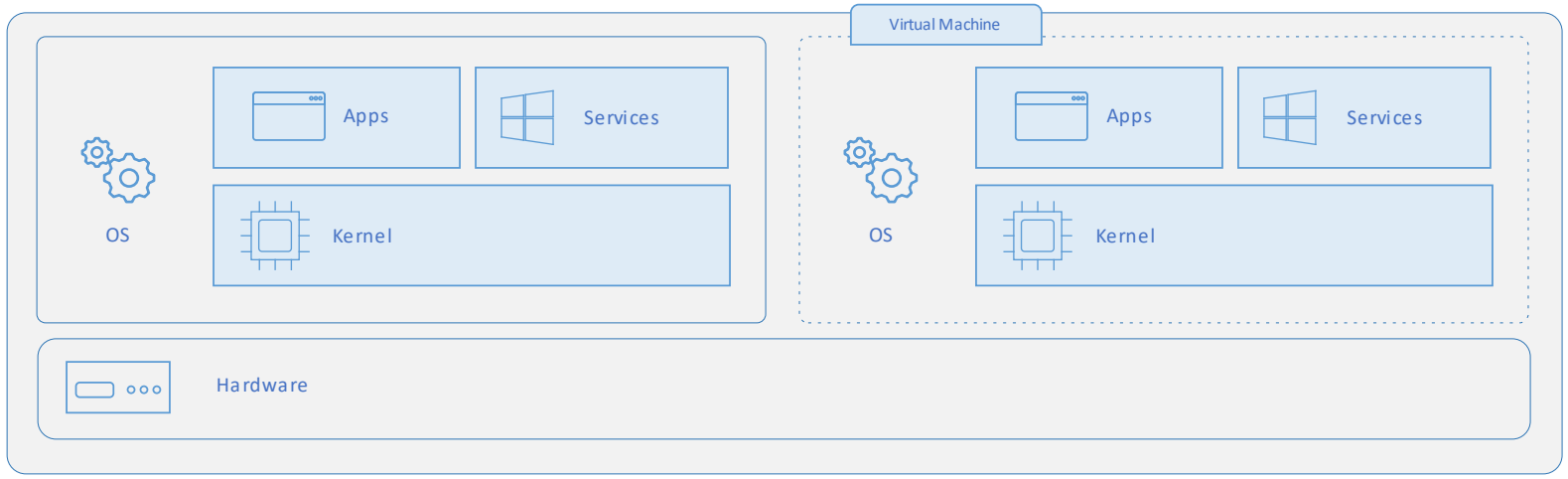
Introduction to Containers

Notes

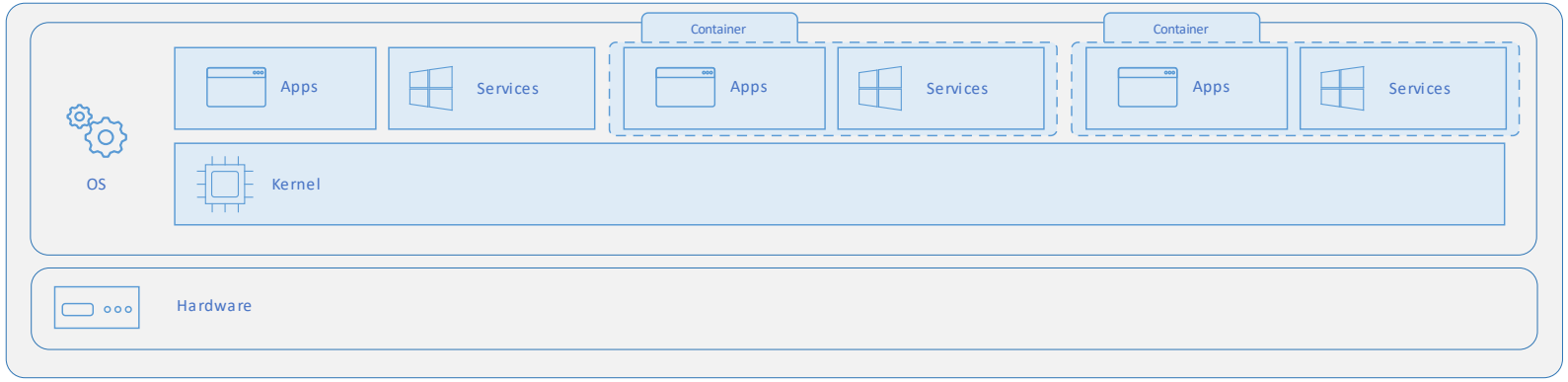
- Containers will use the kernel from the host machine, but has its own user space part of the OS
  - This makes containers a great alternative to VMs for isolation of apps but with less resources utilized
  - Like VMs, containers can only access the memory and processes of the container, not the host system
    - This is done using Cgroups, namespaces, overlay filesystem
  - Containers are faster (in addition to using less resources than VMs) because it doesn't have its own OS kernel
  - Potential drawback: has less isolation than a VM
- Everything needed to run an app baked into a container image
  - Container images are smaller then VM Images

Containers vs VMs

Virtual machine architecture (source)



Container architecture (source)



Some Notes on Kubernetes

Features of K8s:

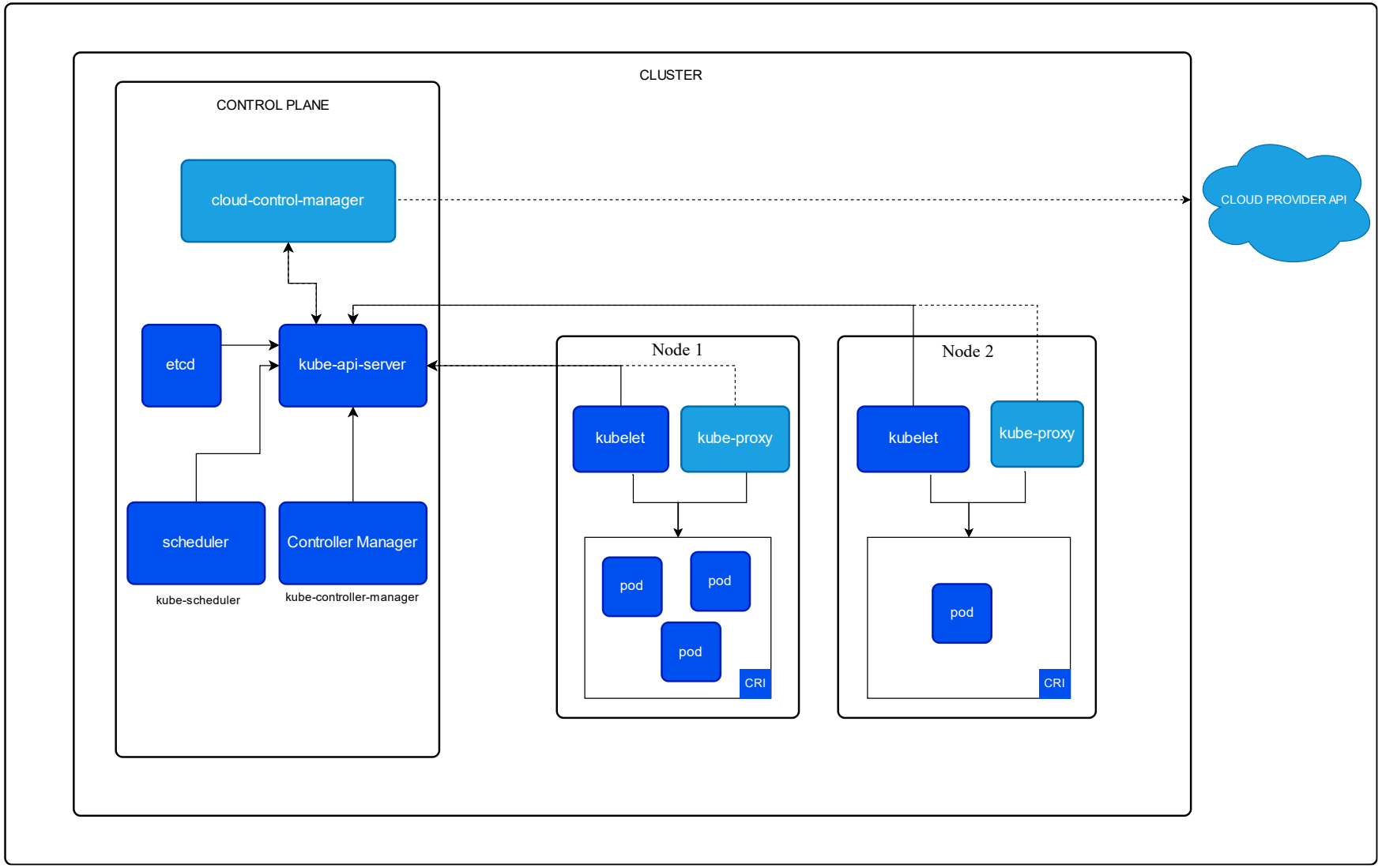
- **Load Balancing:** Distributes network traffic across pods to ensure stability.
- **Self-Healing:** Automatically replaces or restarts failed containers.
- **High Availability**: continuous availability and minimal downtime of applications
- **Rolling updates**: A rolling update in Kubernetes gradually replaces old Pods with new ones to update applications without downtime, ensuring continuous availability and allowing for rollback if issues occur.

Kubernetes Terminology:

- **Clusters:** A group of nodes controlled by Kubernetes .
  - Often is structured as a small number of Control Plane Nodes and a much larger number of Worker Nodes
  - Real world example - never see more than 3 control plane nodes, but might see more than a thousand worker nodes – Patrick's class notes
- **Nodes:** The machines (physical or virtual) that run your containerized applications.
- **Pods:** A Pod is the smallest deployable unit in Kubernetes and is an abstraction that encapsulates one or more containers. Pods provide the environment and configuration for running containers, including shared networking, storage, and metadata.

Kubernetes Architecture

Kubernetes Cluster Architecture (source)



Kubernetes Architecture

Notes on the architecture

- Cluster is split into 2 types of nodes (can be physical or virtual)
    a. control plane machines/nodes
        1. **API server** (listens for https requests from the user to the cluster)
        2. **Database (etcd)** which contains all resource information (like containers, pods, deployments, ingress, etc). And any resource used in the cluster is an added entry to the database
        3. **Scheduler:** responsible for assigning Pods to nodes. It looks at the current state of the cluster and the specifications of the Pods (such as resource requirements) and schedules them on the most suitable nodes.
        4. **Controller Manager:** manages various controllers that regulate different aspects of the cluster, such as ensuring the correct number of Pods are running, managing node states, handling job execution, and more.
        5. **cloud-controller-manager**: When running Kubernetes in a cloud environment, this component interacts with the cloud provider's API, handling tasks like managing load balancers, storage volumes, and networking

Kubernetes Architecture

Notes on the architecture

- Cluster is split into 2 machines (can be physical or virtual)
  a. Control plane (see previous slide)
  b. Worker machines/nodes
     1. **Pods:** The smallest deployable units in Kubernetes, which can contain one or more containers.
        - Have an IP address, which connects them to other pods in the cluster
        - Address is shared by all containers in the pod
     2. **Kubelet:** An agent which runs on each node. It makes sure that containers are running
     3. **kube-proxy:** maintains network rules on nodes to route traffic between services and pods, supporting service discovery and load balancing
- *Summary of some of the parts: Containers run applications, Pods encapsulate containers with shared resources and configurations, and Nodes provide the physical or virtual machines that run Pods within a Kubernetes cluster.*

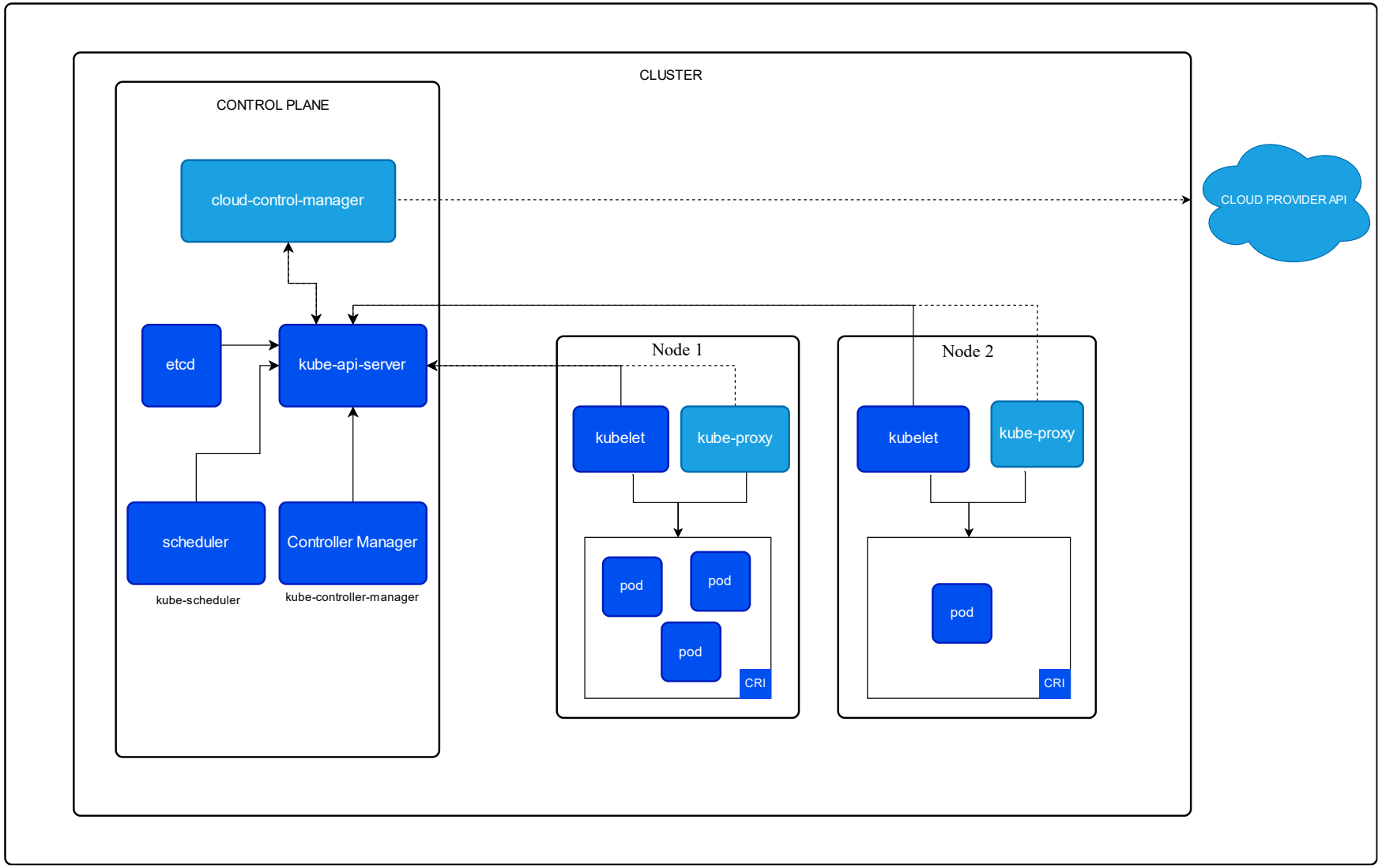
Kubernetes Architecture

More notes on Kubernetes

- **Virtual networking**: Virtual networking in Kubernetes abstracts network communication between pods, services, and nodes, enabling seamless, scalable, and secure connectivity across the cluster using overlay networks, network policies, and service discovery mechanisms.
  - Lots of fun topics here like DNS, Network Policies, Overlay Networks, etc.
- **Storage volumes**: Storage volumes in Kubernetes provide persistent or temporary storage to Pods, allowing data to survive beyond the Pod's lifecycle, with various types like emptyDir, hostPath, PersistentVolumes, NFS, cloud-specific volumes, and CSI-based solutions for flexibility and scalability.
- **Stateful vs. stateless pods**: Stateless Pods do not retain data between restarts and are interchangeable, ideal for scalable web services, while Stateful Pods retain data and have unique identities, making them suitable for stateful applications like databases or distributed systems.

Kubernetes Architecture

Kubernetes Cluster Architecture (source)



Kubernetes at SLAC

Where and how Kubernetes is being used here at SLAC

Build system:

- The database support is implemented in three long-lived pods, and the artifact storage service is a fourth long-lived pod.
  - Artifact is the binary, DB file, etc, anything to run the app
- Temporary pods are spun up to run containers in the course of building/testing/deploying software and will vary based on load/activity.
  - The build system will start a pod, pod/container clones app from github, then looks at repo's config file for dependencies, grabs them from the artifact storage service pod. Starts build in the container.
  - This solution only has stateless pods
- The users of the build system will not be interacting with the Kubernetes

Lume-services-Deployment

- GPUs on K8s nodes
- Train ML Models

Resources for learning more about Kubernetes

Useful links for more information on these topics:

Kubernetes:

- https://kubernetes.io/docs/home/
  - https://kubernetes.io/docs/concepts/architecture

VMs and Containers:

- https://learn.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm