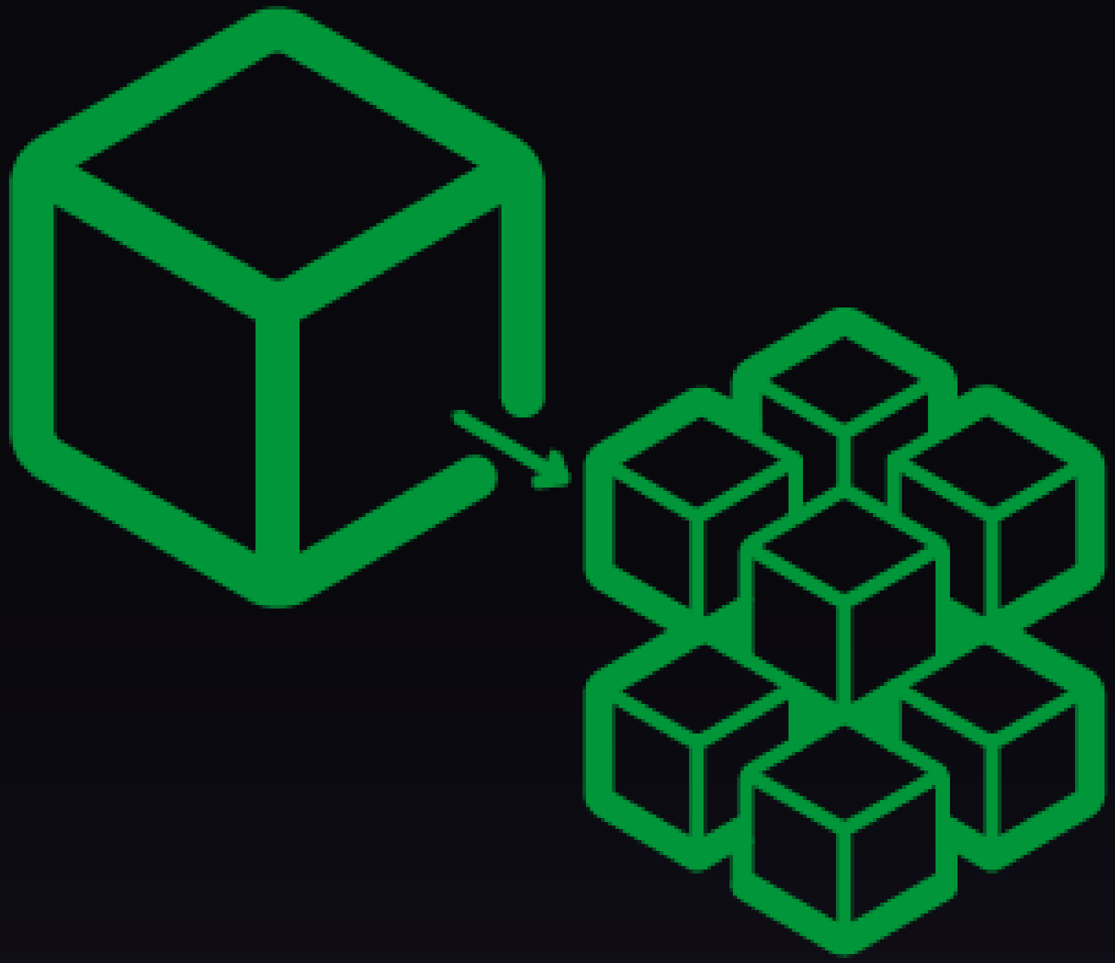
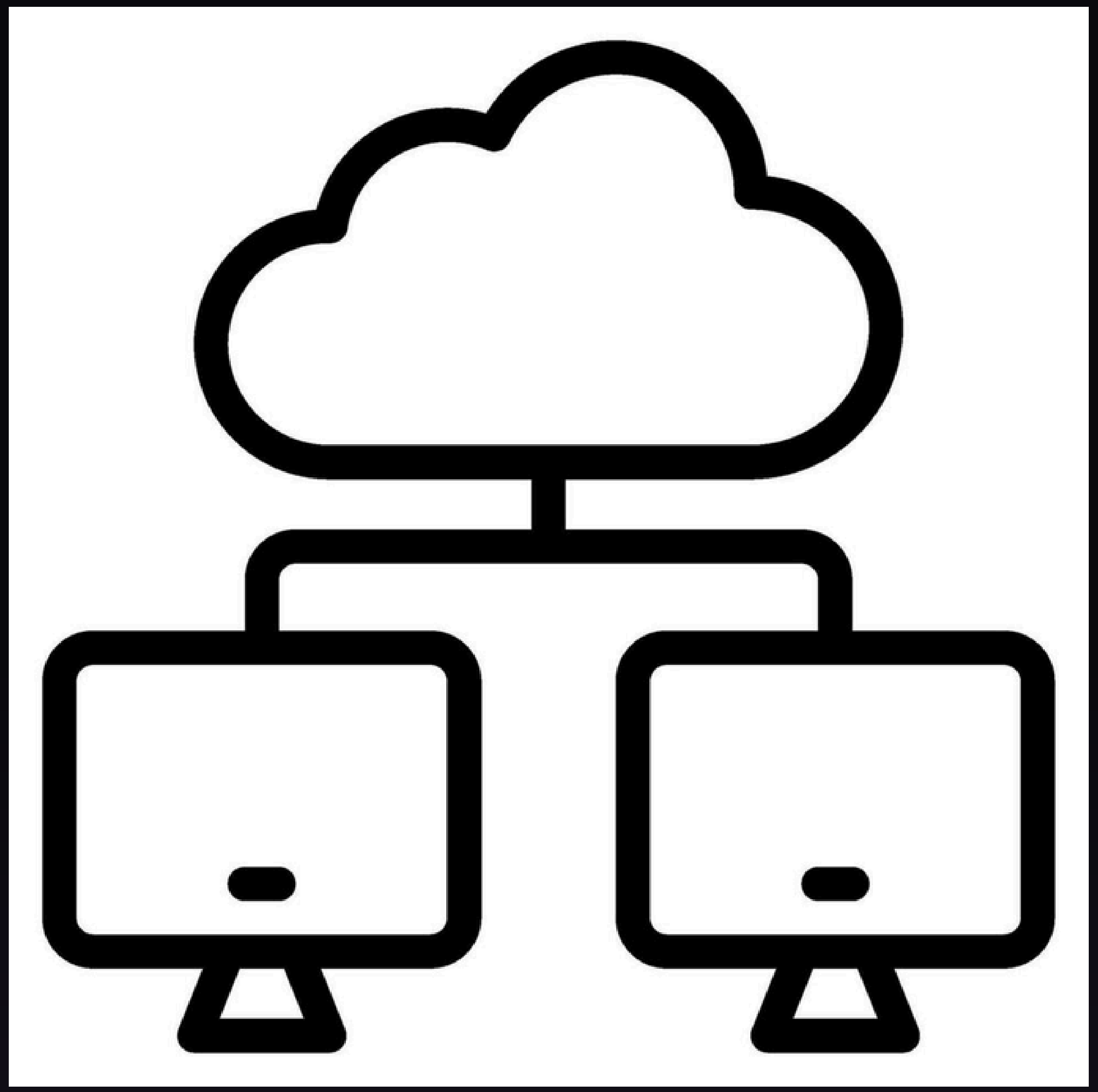
LearnYard

Kubernetes Series

Microservices



Distributed Computing



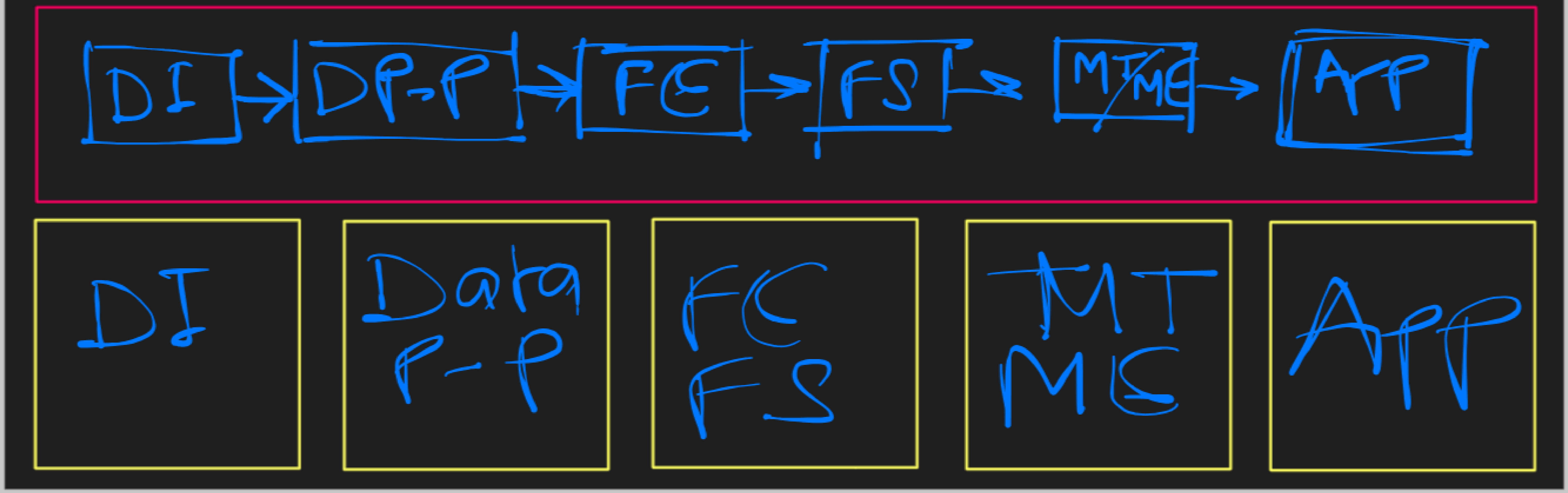
TOPICS TO BE COVERED

- Our playlist so far. (Pre-requisite)
- Distributed Computing
  - Cluster
  - Lead-Node Server
  - Communication
  - Concurrency [Speed, Fault Tolerance]
  - Same as inside Apache Spark internally (Map Reduce)
- Microservices (How related to Docker & Kubernetes?)
- Kubernetes Internals

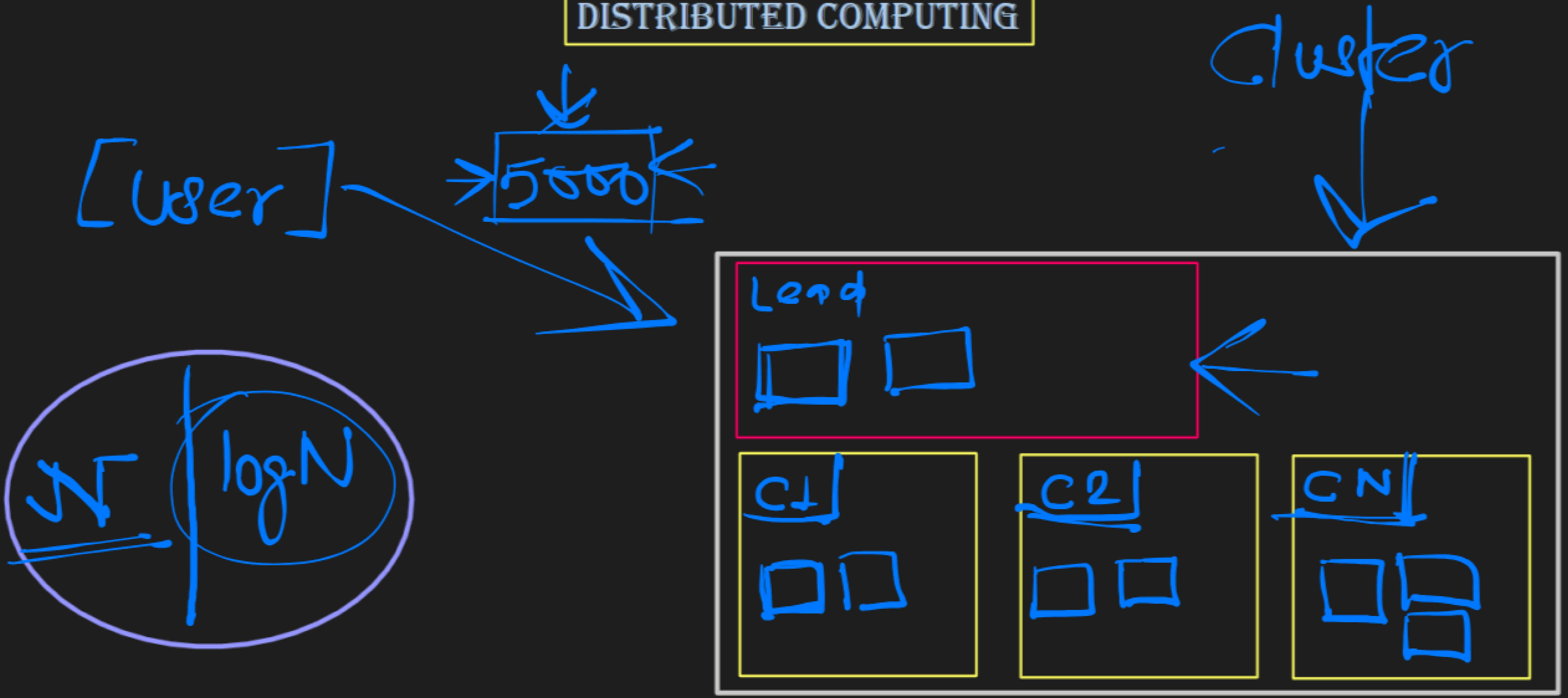
KUBERNETES INTERNALS

- Master Node (Control Plane)
  - Resource Manager
  - Communication (User to API Server)
  - Database (etcd)
- Worker Node
  - Kubelet
  - Kube-Proxy
  - Pods
  - SharedDB (volumes)
  - Kube-manifest (yaml)
  - Service
  - Namespace
  - Scheduler
  - Replica Set
- VS Code - Kubernetes Extension

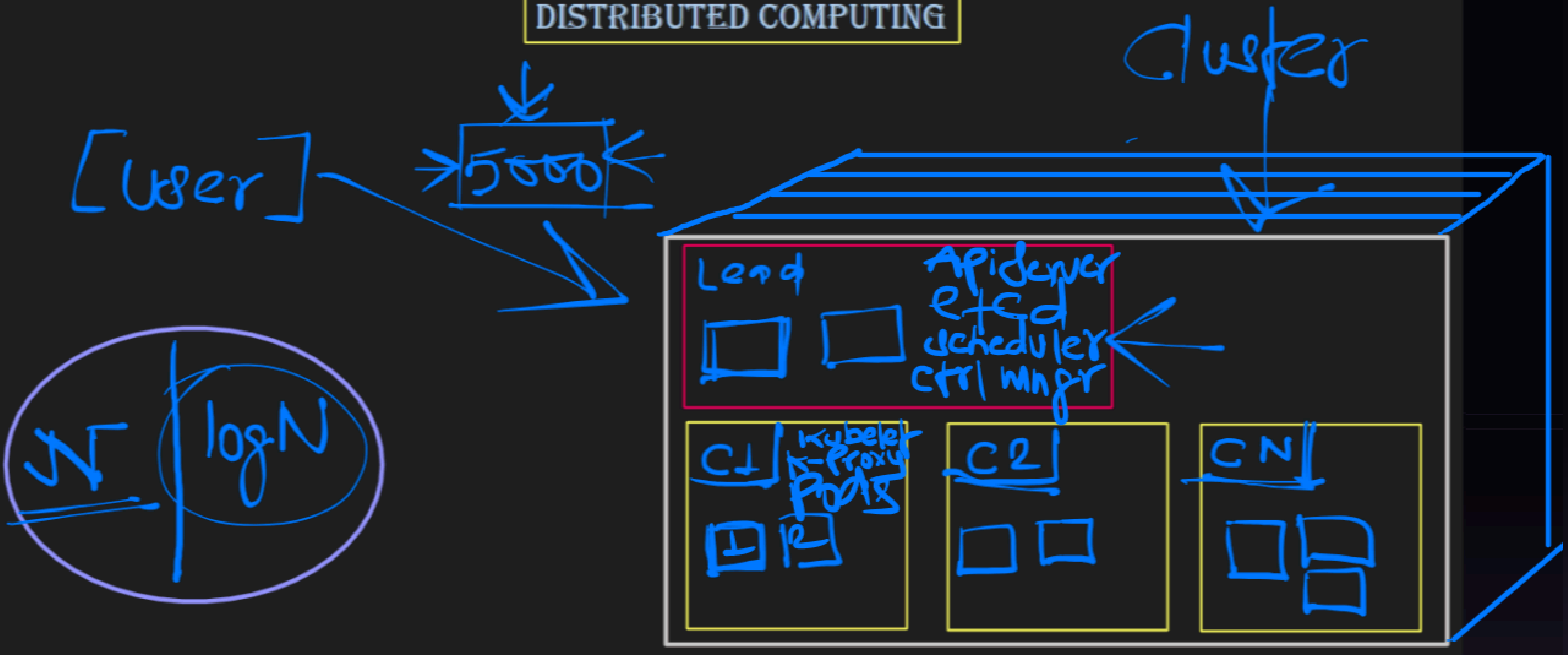
MICROSERVICES EXPLAINED



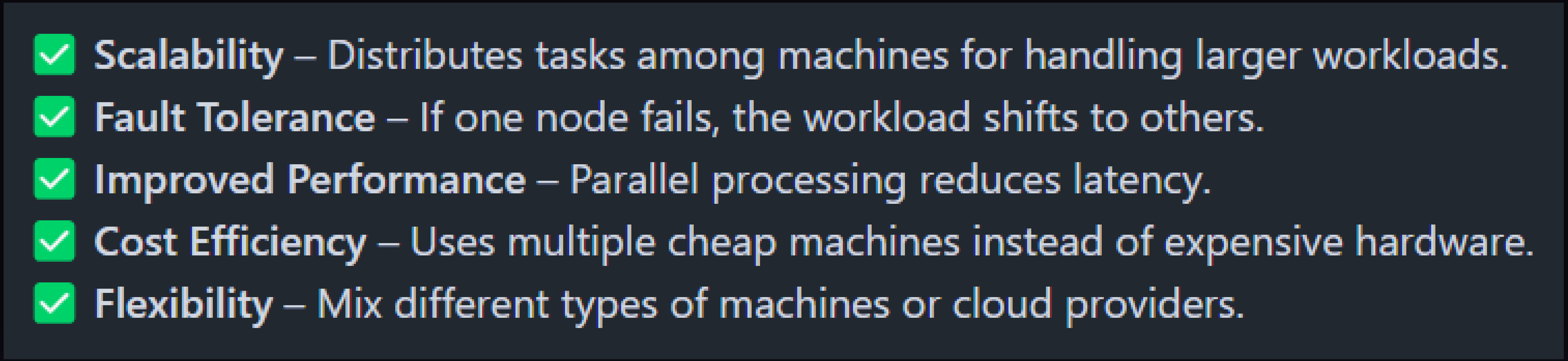
DISTRIBUTED COMPUTING



DISTRIBUTED COMPUTING

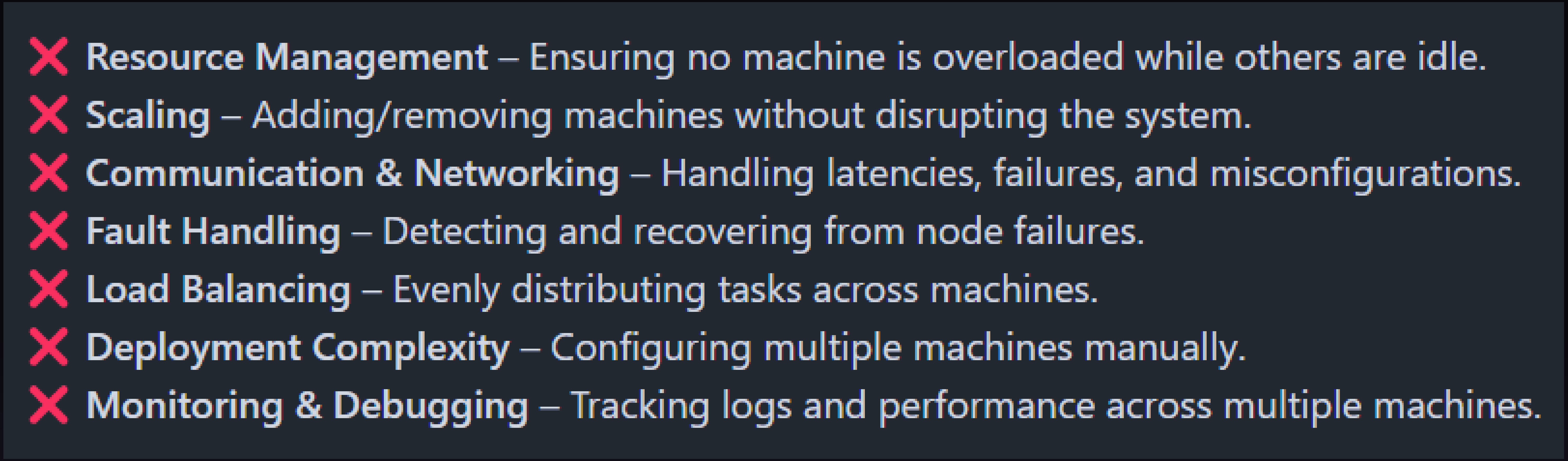


Benefits of Distributed Computing



- ✅ **Scalability** – Distributes tasks among machines for handling larger workloads.
- ✅ **Fault Tolerance** – If one node fails, the workload shifts to others.
- ✅ **Improved Performance** – Parallel processing reduces latency.
- ✅ **Cost Efficiency** – Uses multiple cheap machines instead of expensive hardware.
- ✅ **Flexibility** – Mix different types of machines or cloud providers.

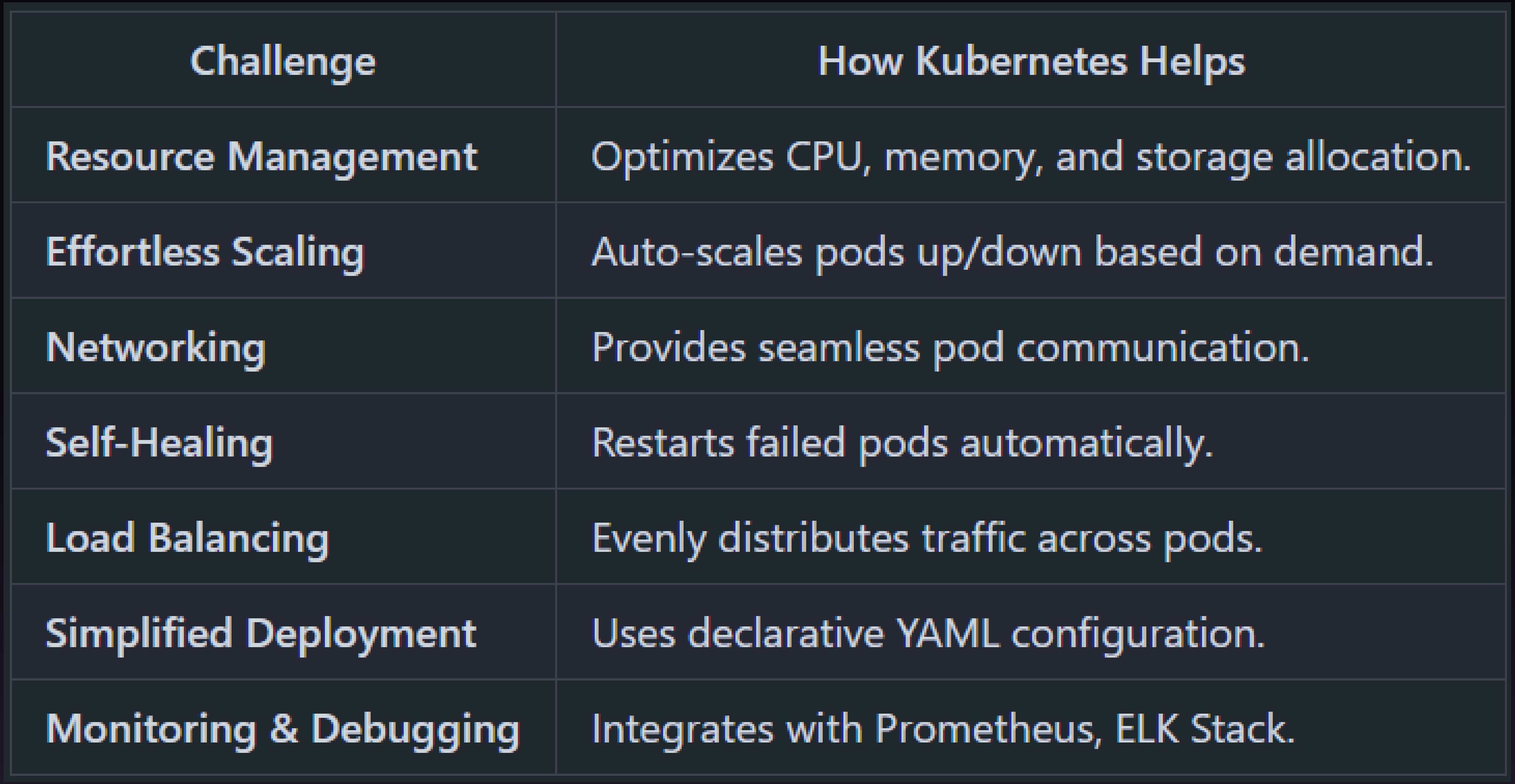
Challenges of Distributed Computing



- ❌ **Resource Management** – Ensuring no machine is overloaded while others are idle.
- ❌ **Scaling** – Adding/removing machines without disrupting the system.
- ❌ **Communication & Networking** – Handling latencies, failures, and misconfigurations.
- ❌ **Fault Handling** – Detecting and recovering from node failures.
- ❌ **Load Balancing** – Evenly distributing tasks across machines.
- ❌ **Deployment Complexity** – Configuring multiple machines manually.
- ❌ **Monitoring & Debugging** – Tracking logs and performance across multiple machines.

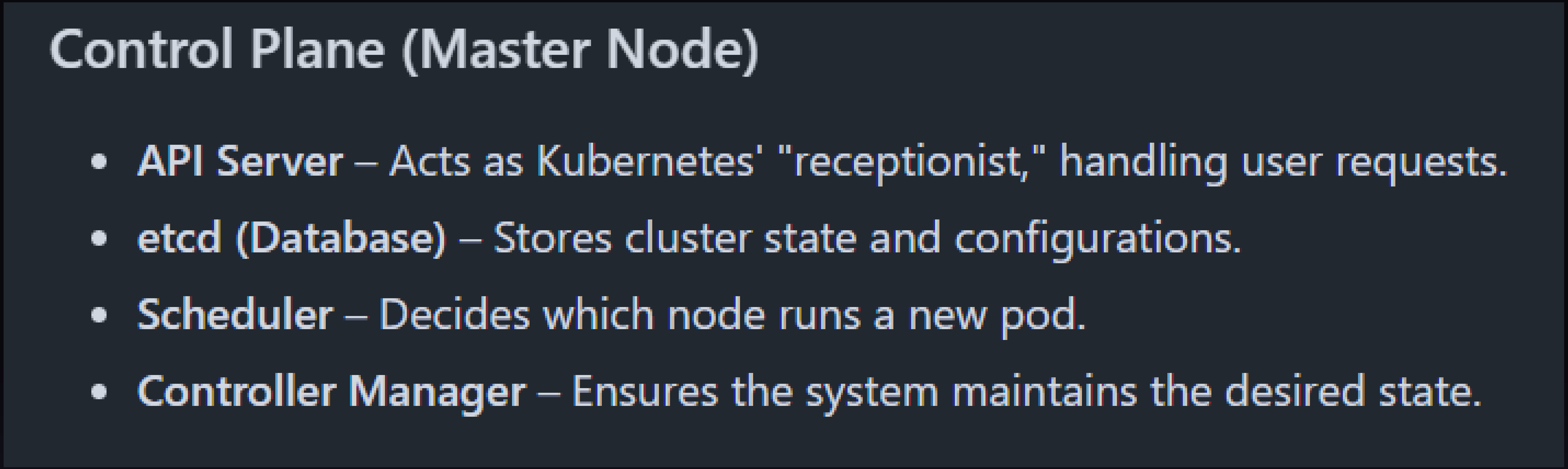
How Kubernetes Solves These Challenges

Kubernetes is a container orchestration platform designed to simplify distributed computing.



Challenge	How Kubernetes Helps
Resource Management	Optimizes CPU, memory, and storage allocation.
Effortless Scaling	Auto-scales pods up/down based on demand.
Networking	Provides seamless pod communication.
Self-Healing	Restarts failed pods automatically.
Load Balancing	Evenly distributes traffic across pods.
Simplified Deployment	Uses declarative YAML configuration.
Monitoring & Debugging	Integrates with Prometheus, ELK Stack.

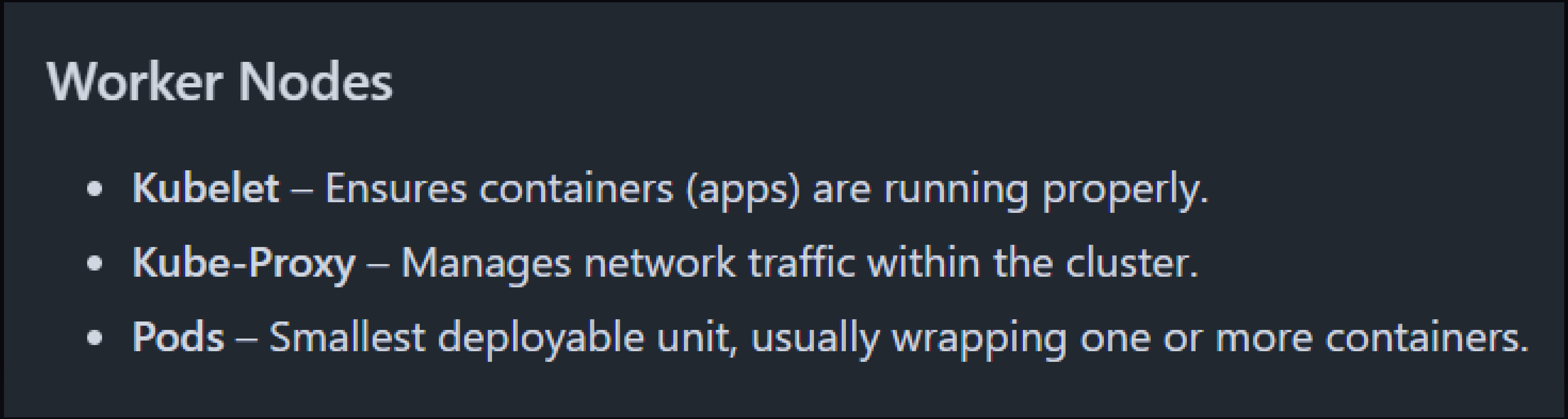
Kubernetes Internals



Control Plane (Master Node)

- **API Server** – Acts as Kubernetes' "receptionist," handling user requests.
- **etcd (Database)** – Stores cluster state and configurations.
- **Scheduler** – Decides which node runs a new pod.
- **Controller Manager** – Ensures the system maintains the desired state.

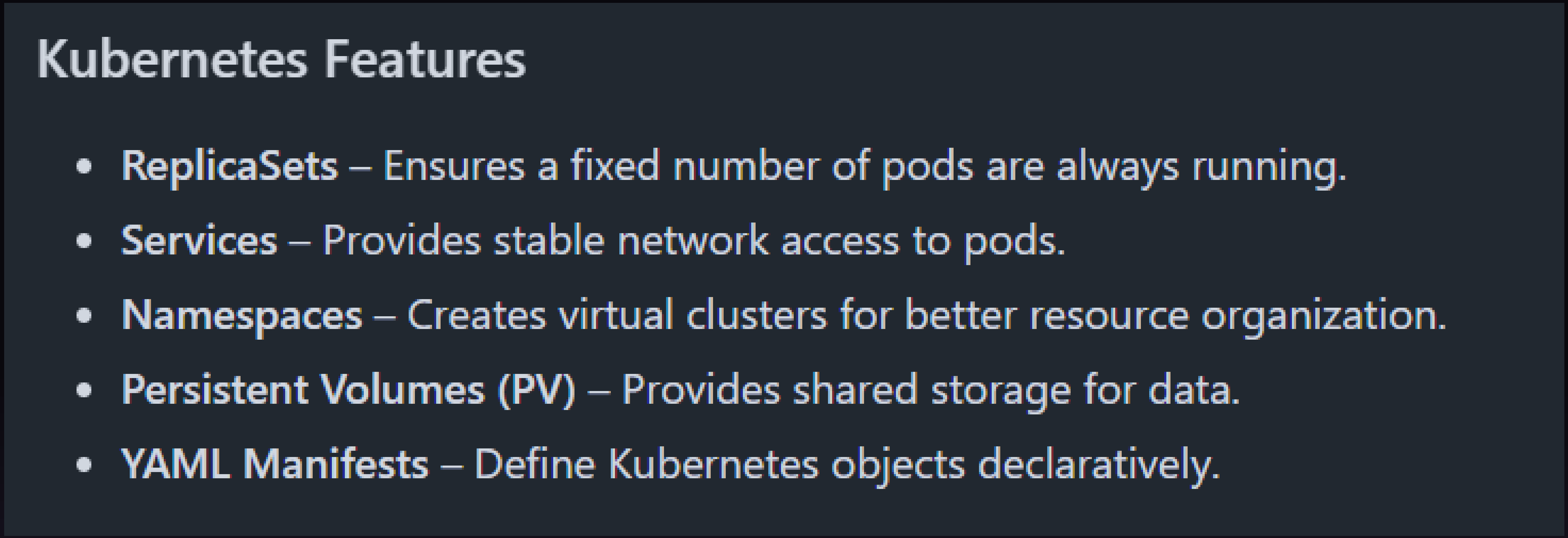
Kubernetes Internals



Worker Nodes

- **Kubelet** – Ensures containers (apps) are running properly.
- **Kube-Proxy** – Manages network traffic within the cluster.
- **Pods** – Smallest deployable unit, usually wrapping one or more containers.

Kubernetes Internals



Kubernetes Features

- **ReplicaSets** – Ensures a fixed number of pods are always running.
- **Services** – Provides stable network access to pods.
- **Namespaces** – Creates virtual clusters for better resource organization.
- **Persistent Volumes (PV)** – Provides shared storage for data.
- **YAML Manifests** – Define Kubernetes objects declaratively.