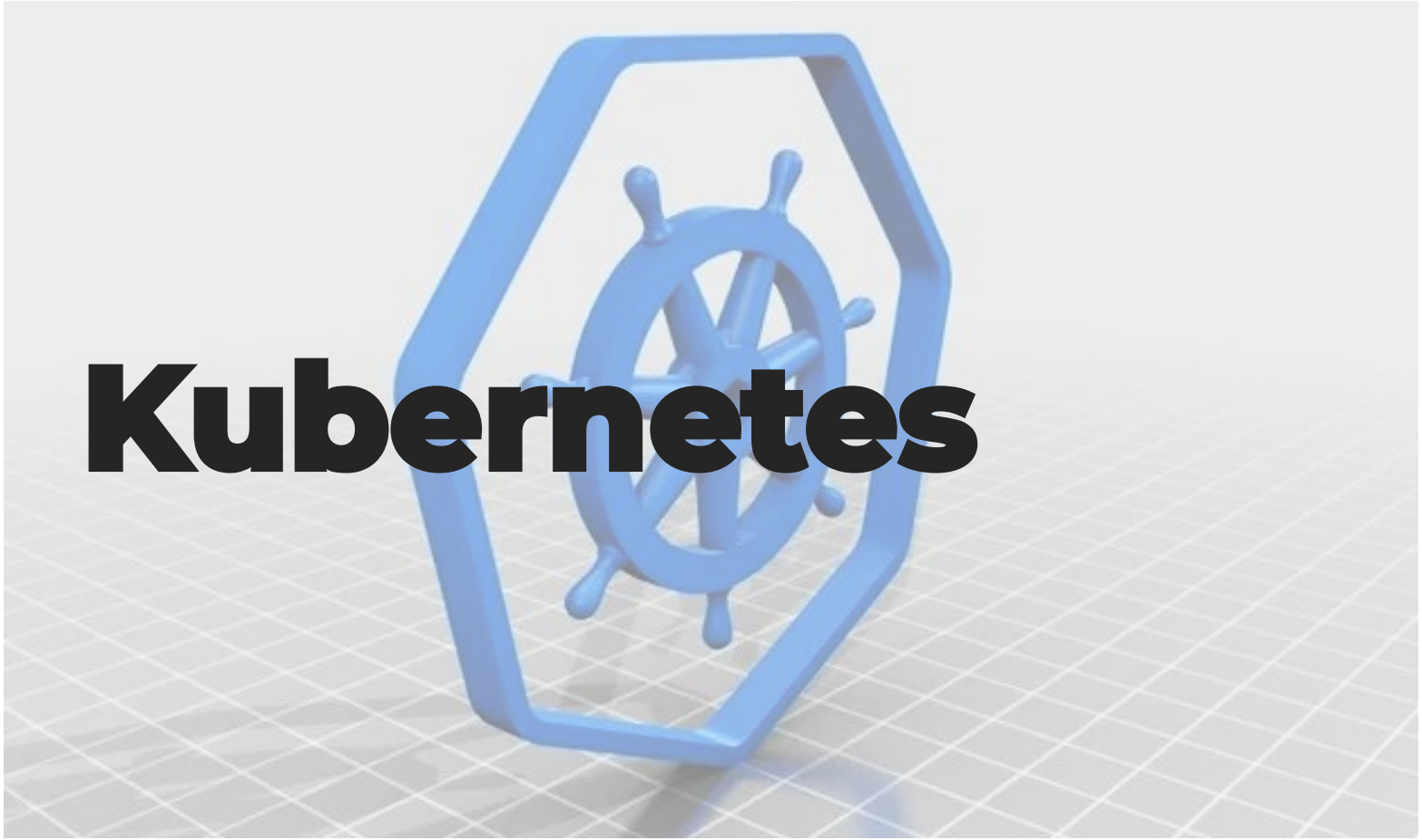
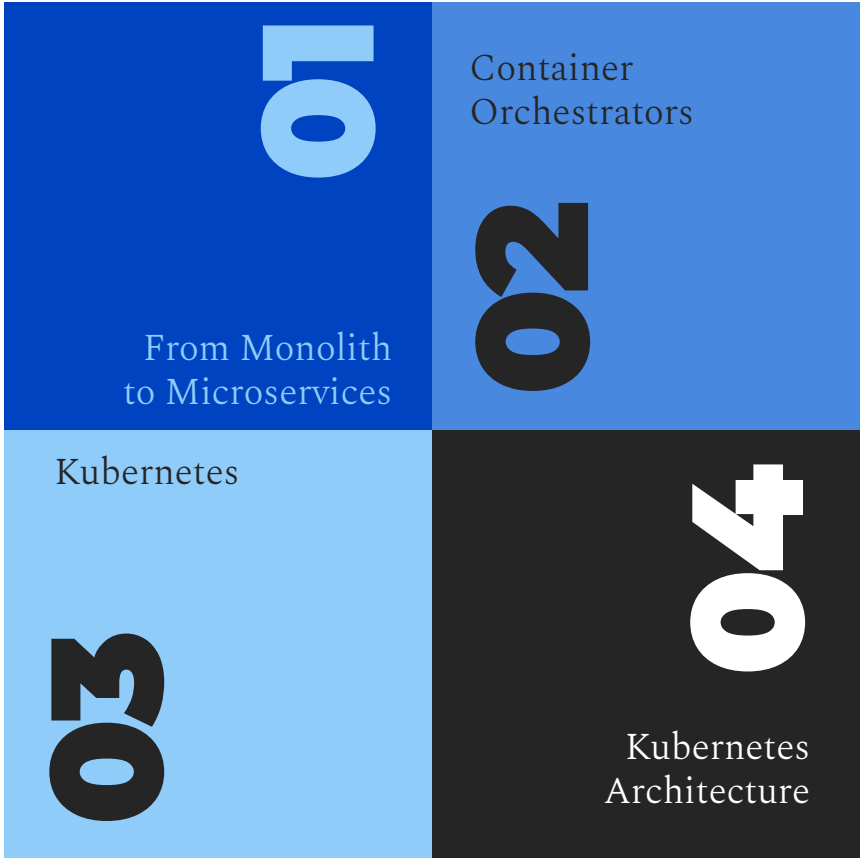

Kubernetes


01
From Monolith
to Microservices
Container
Orchestrators
02
Kubernetes
03
04
Kubernetes
Architecture

Agenda

1. From Monolith to Microservices

Describe the transformation path from a monolith to microservices.

The Legacy Monolith to Microservices!

In our fast-changing world, business success depends on being able to react quickly to threats, opportunities and change. As a result, businesses have shifted from cumbersome monolithic applications to smaller, loosely coupled services

—microservices

2. Container Orchestrators

Define the concept of container orchestration.

Containers

Containers are application-centric methods to deliver high-performing, scalable applications on any infrastructure of your choice. Containers are best suited to deliver microservices by providing portable, isolated virtual environments for applications to run without interference from other running applications.

Microservices

Microservices are lightweight applications written in various modern programming languages, with specific dependencies, libraries and environmental requirements. To ensure that an application has everything it needs to run successfully it is packaged together with its dependencies.

Container Image

A **container image** bundles the application along with its runtime and dependencies, and a container is deployed from the container image offering an isolated executable environment for the application. Containers can be deployed from a specific image on many platforms, such as workstations, Virtual Machines, public cloud, etc.

3. Kubernetes

Define Kubernetes.

Kubernetes is an **open-source system** for automating deployment, scaling, and management of containerized applications.

Kubernetes is highly inspired by the Google Borg system

Kubernetes Features I

Automatic bin packing

Kubernetes automatically schedules containers based on resource needs and constraints, to maximize utilization without sacrificing availability.

Self-healing

Kubernetes automatically replaces and reschedules containers from failed nodes. It kills and restarts containers unresponsive to health checks, based on existing rules/policy. It also prevents traffic from being routed to unresponsive containers.

Horizontal scaling

With Kubernetes applications are scaled manually or automatically based on CPU or custom metrics utilization.

Service discovery and Load balancing

Containers receive their own IP addresses from Kubernetes, while it assigns a single Domain Name System (DNS) name to a set of containers to aid in load-balancing requests across the containers of the set.

Kubernetes Features II

Automated rollouts and rollbacks

Kubernetes seamlessly rolls out and rolls back application updates and configuration changes, constantly monitoring the application's health to prevent any downtime.

Secret and configuration management

Kubernetes manages secrets and configuration details for an application separately from the container image, in order to avoid a re-build of the respective image.

Storage orchestration

Kubernetes automatically mounts software-defined storage (SDS) solutions to containers from local storage, external cloud providers, or network storage systems.

Batch execution

Kubernetes supports batch execution, long-running jobs, and replaces failed containers.

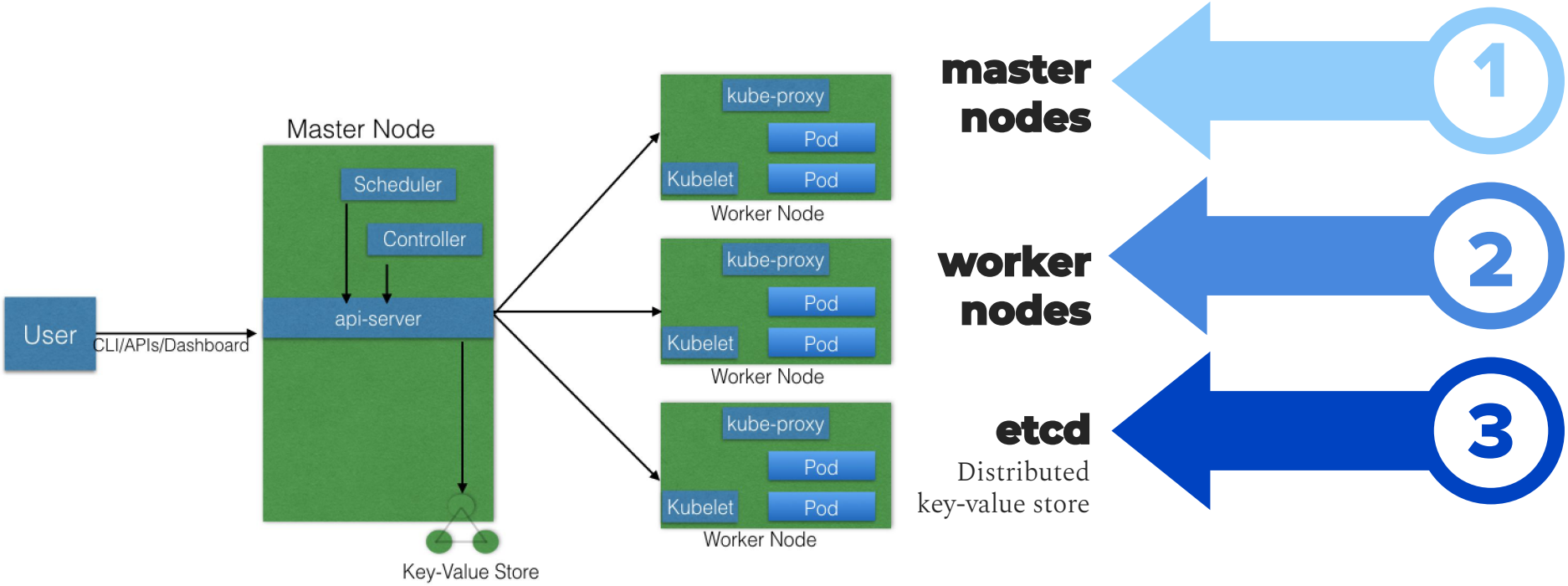
Why use **Kubernetes** ?

- *Kubernetes can be deployed in many environments such as local or remote Virtual Machines, bare metal, or in public/private/hybrid/multi-cloud setups.*
- *Kubernetes' architecture is modular and pluggable.*
- *Kubernetes' functionality can be extended by writing custom resources, operators, custom APIs, scheduling rules or plugins.*

4. Kubernetes Architecture

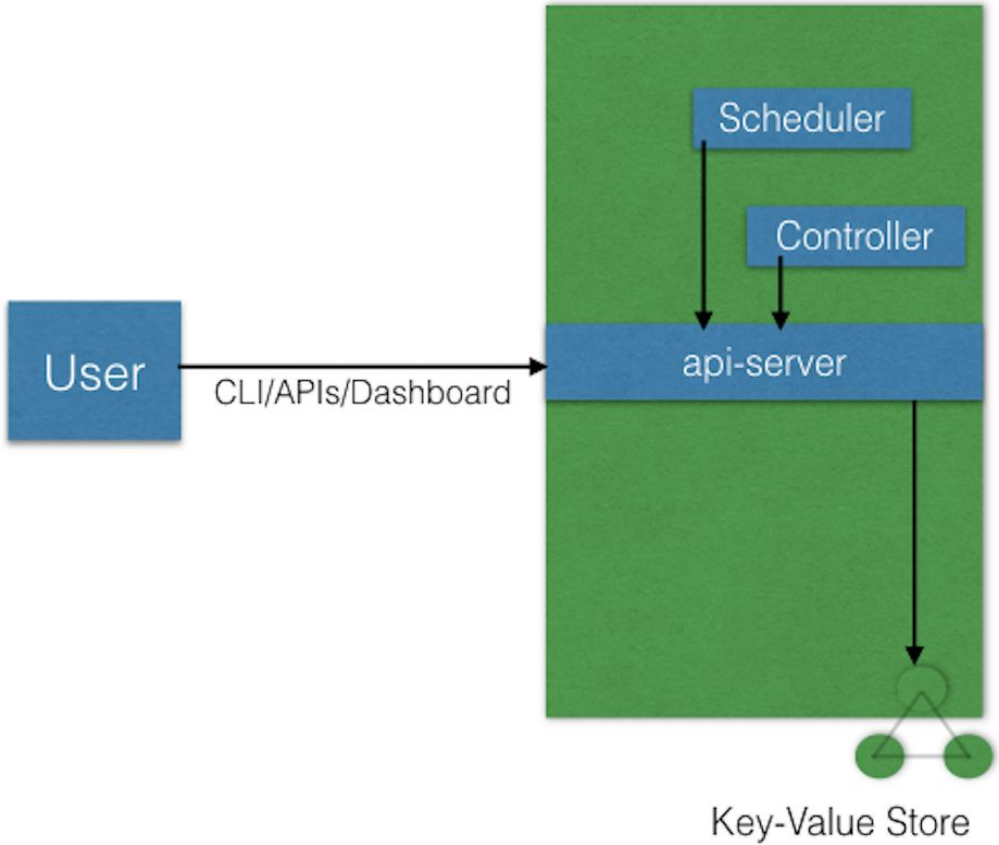
Explain the different components for master and worker nodes

KUbernates has the following main components:

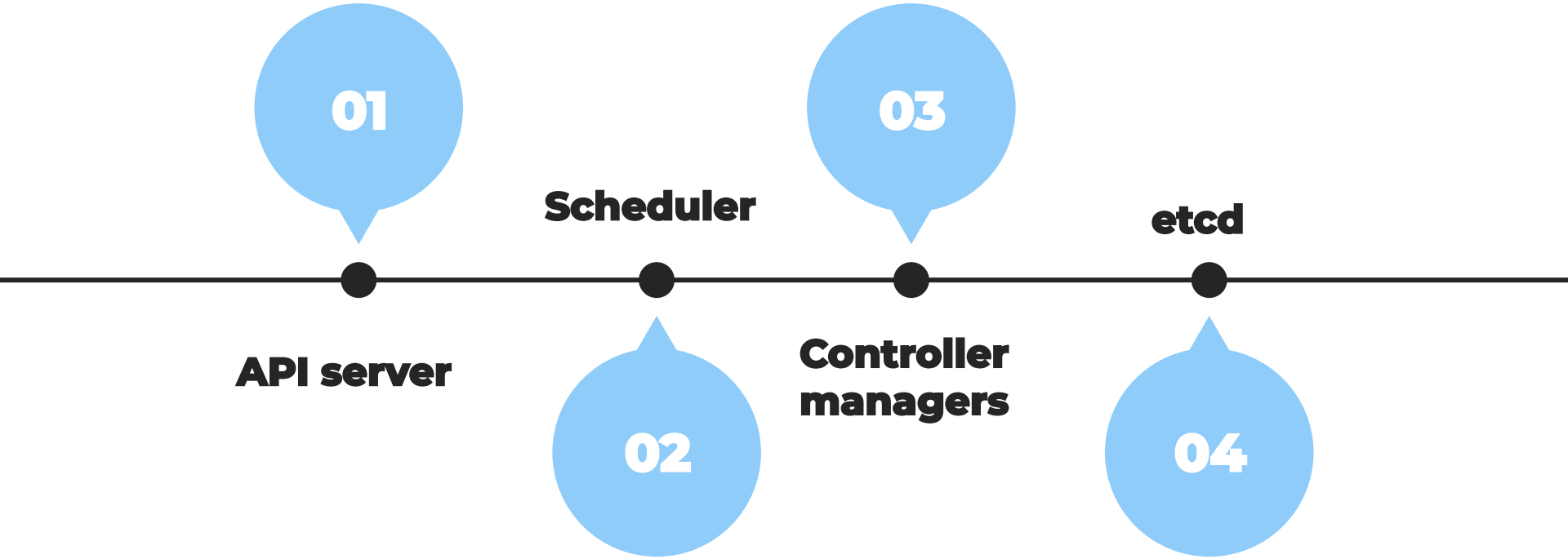


Master Node

The **master node** provides a running environment for the control plane responsible for managing the state of a Kubernetes cluster, and it is the brain behind all operations inside the cluster.

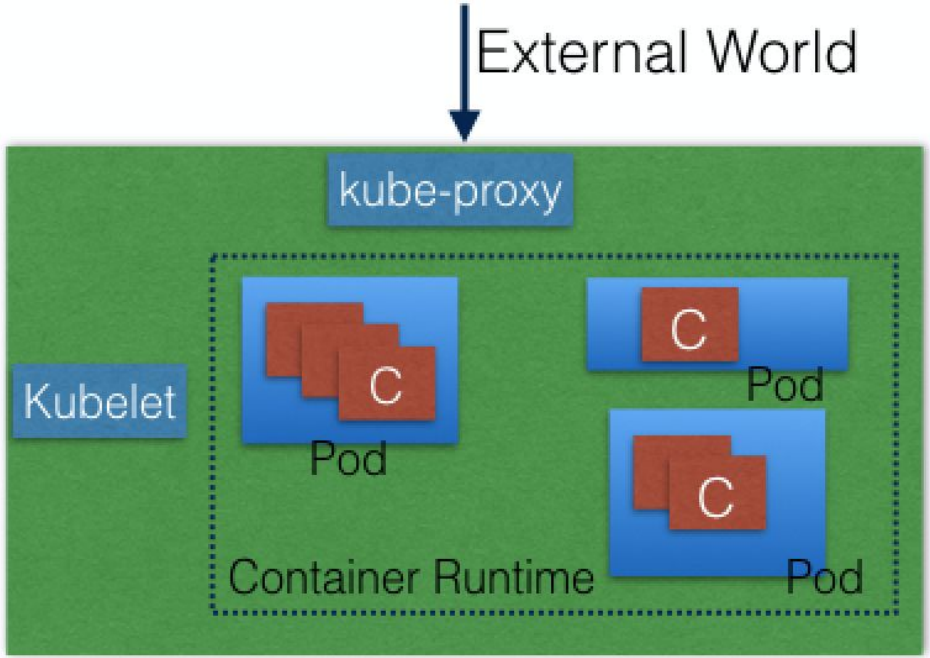


Master Node
Components

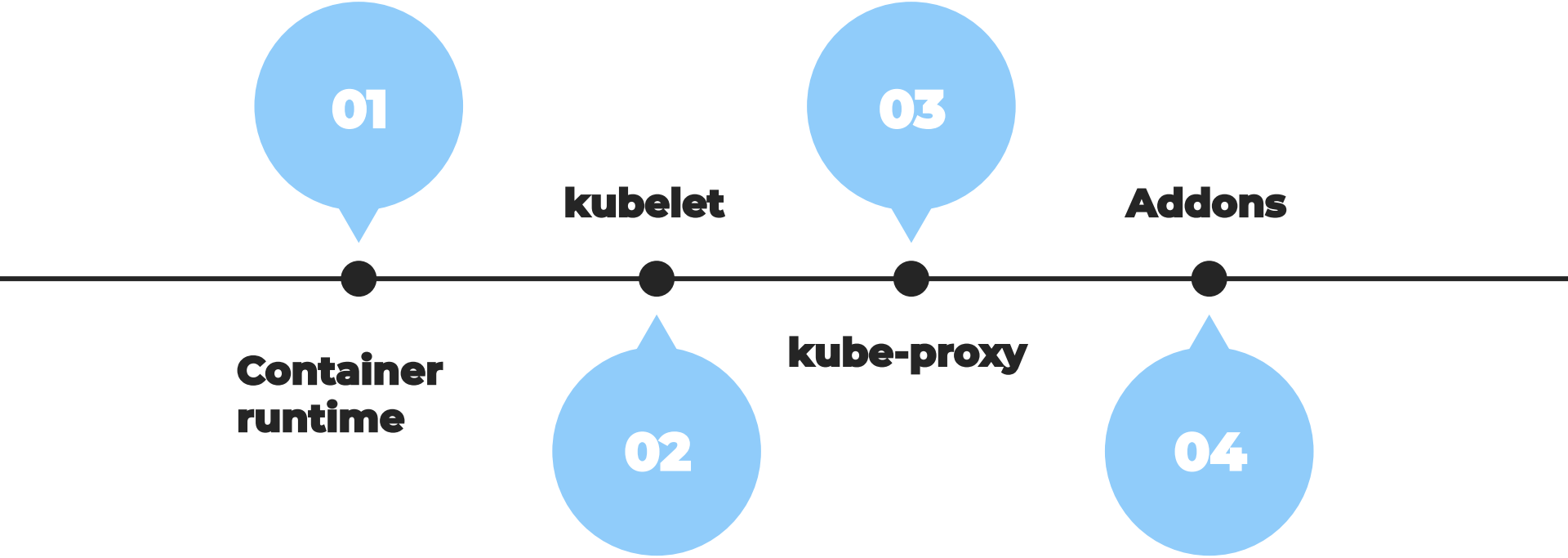


Worker Node

A **worker node** provides a running environment for client applications. Though containerized microservices, these applications are encapsulated in Pods, controlled by the cluster control plane agents running on the master node



Worker Node Components




Thanks!