kubernetes

Software Architecture

Matteo Marchesini

A.A. 2018/2019

INTRODUCTION

- A **container** is a technology for the deployment of distributed systems, a sort of virtualized server within an operating system
- **Container orchestration** is a process that allows to distribute a multiplicity of containers in order to implement an application

KUBERNETES

- An open source system for managing containerized applications between multiple hosts
- Microsoft, IBM and Google offer Container as a Service (CaaS) platforms based on Kubernetes
- Initially developed by Google in 2015 and currently supported by **Cloud Native Computing Foundation (CNCF)**

KUBERNETES FEATURES

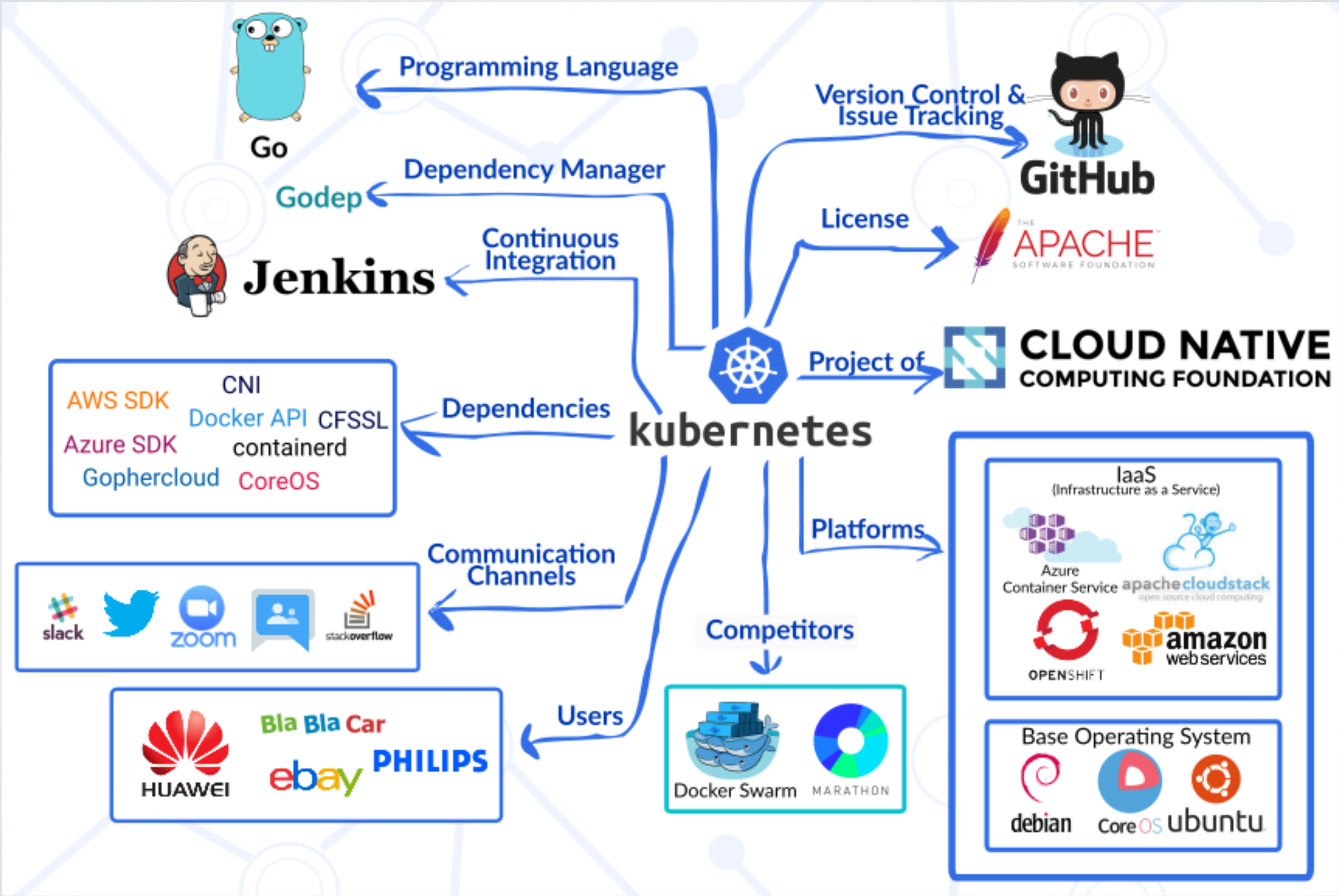
- **Deployment:** it can be performed in a variety of environments with different patterns

- **Scaling:** allows to resize the application according to your needs, replicating the *pods*.

- **Management:** provides an interface for managing cluster and containerized applications. e.g. *Google Container Engine (GKE)*

Google Cloud

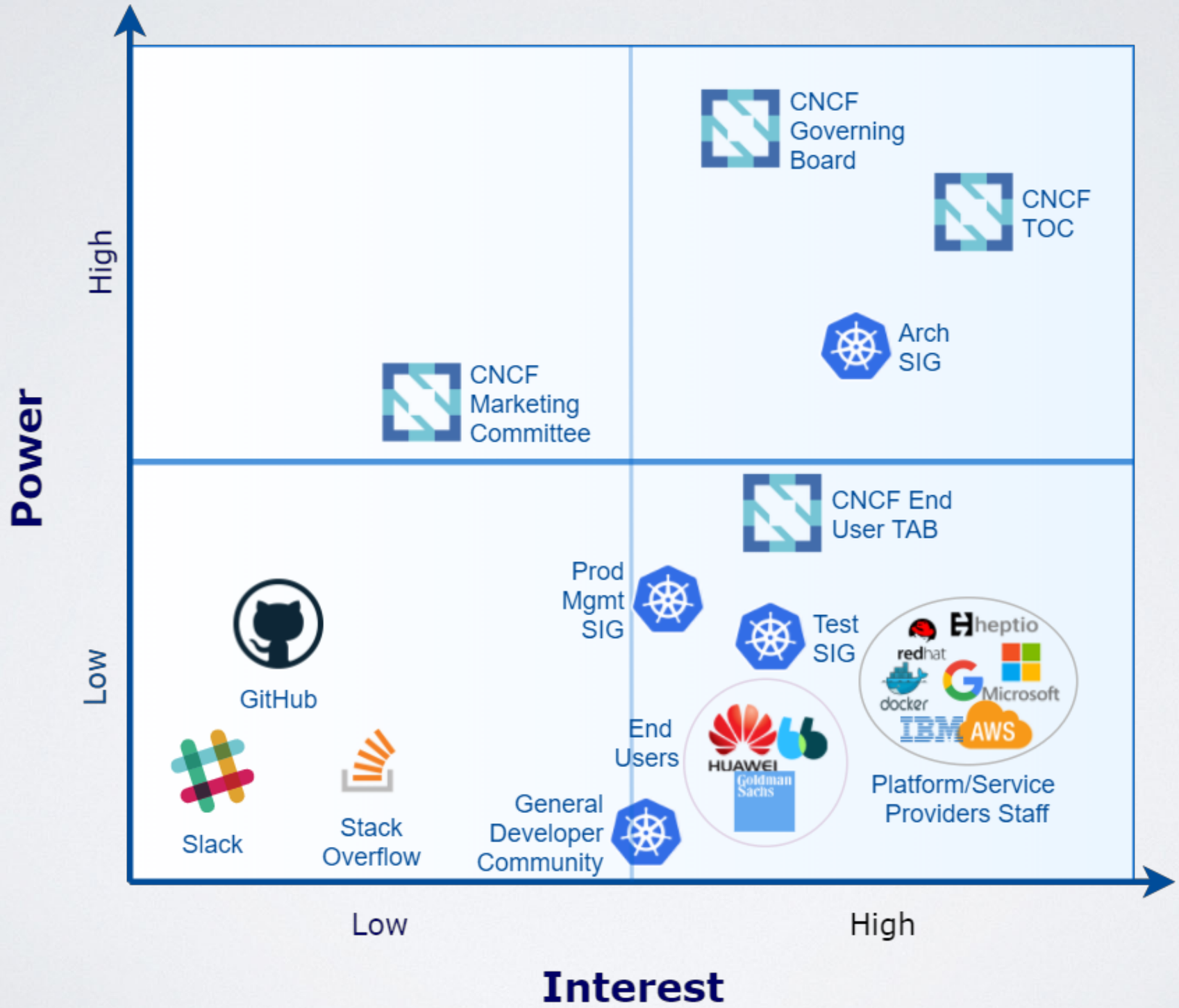
SYSTEM CONTEXT



STAKEHOLDERS

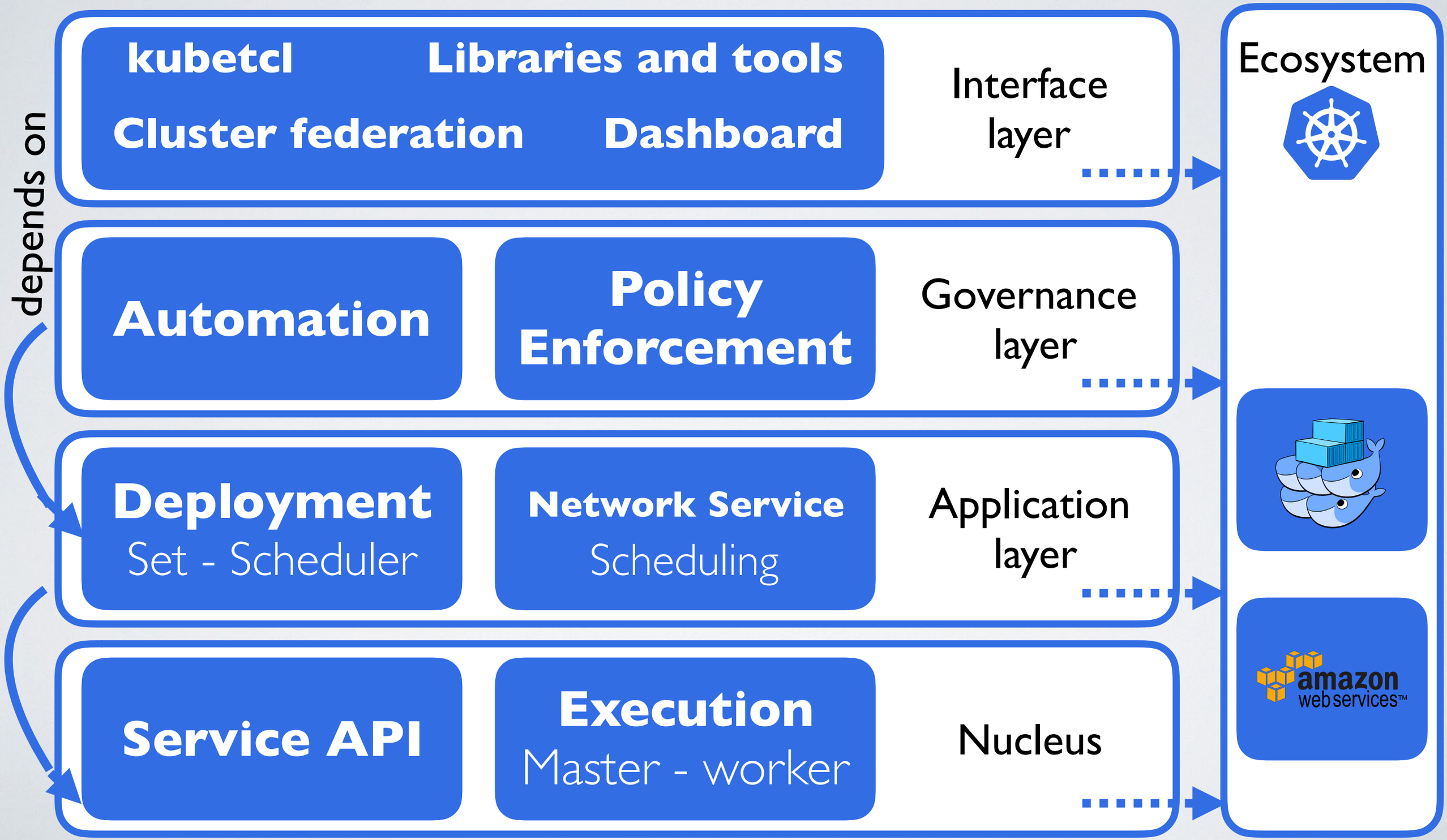
- **Cloud Native Computing Foundation** members are distributed in in *Platinum*, *Gold*, *Silver*, *End-User, Academic* and *Non-profit* membership levels.
- There are **3 high-level divisions** in CNCF:
  - Governing Board (Platinum CNCF members)
  - Technical Oversight Committee (TOC)
  - End User Technical Advisory Board (TAB)
- **Suppliers** (*Github*, *Slack*, and *StackOverflow)*
- **Special Interest Group** (SIG) for Kubernetes' development.

POWER/INTEREST GRID



DEVELOPMENT

Kubernetes Module Organization

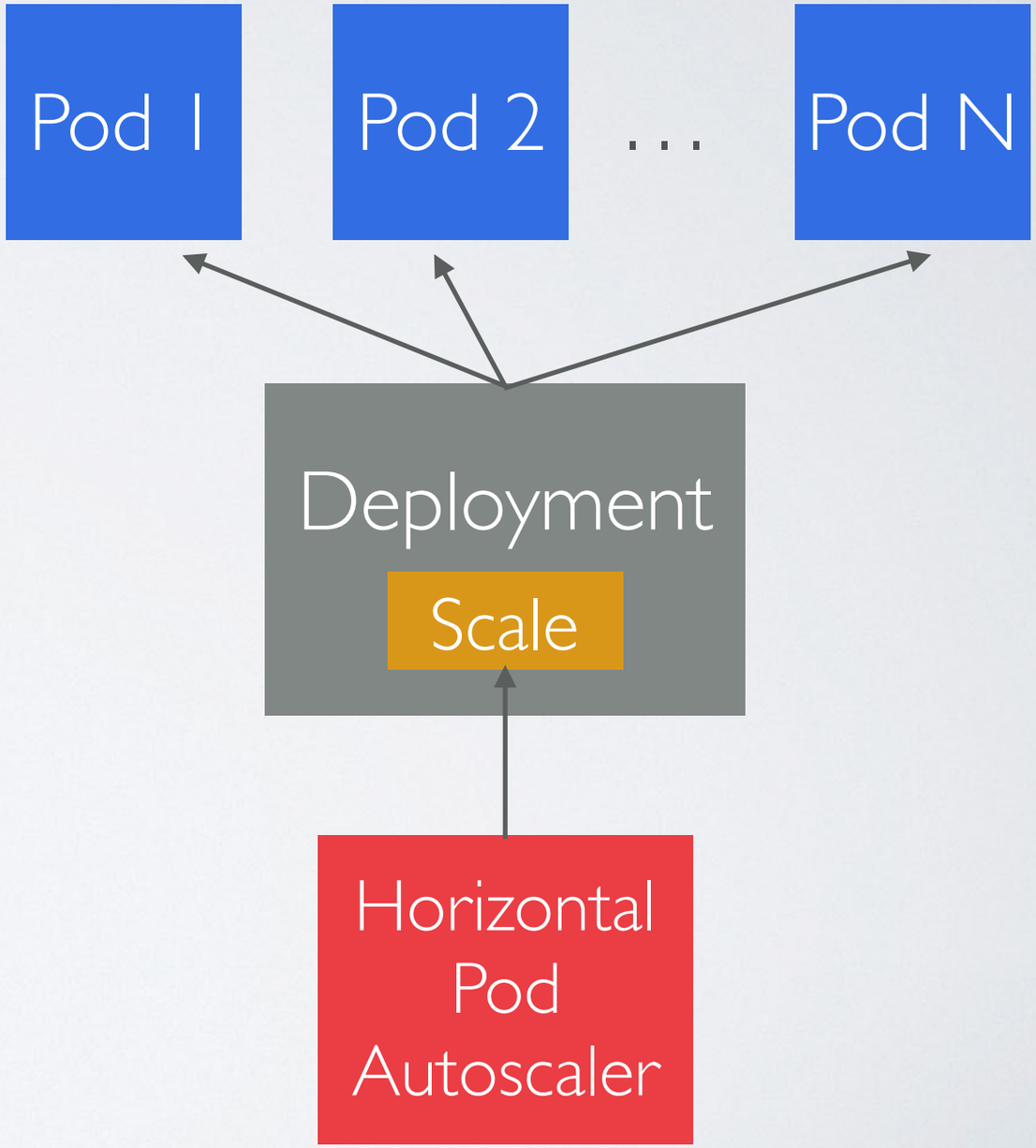


PLATFORMS

- Minikube is a **local**-machine solution
- AppsCode, Google Kubernetes Engine, Amazon EKS, Azure AKS are **hosted solutions**
- Agile Stacks, Google Compute Engine (GCE), Giant Swarm, IBM Cloud for **Cloud IaaS**
- **Custom solutions**, VMs, Bare Metal

ARCHITECTURAL DRIVERS

- **Scalability**
  - Horizontal auto-scaling
  - It automatically scales the number of pods in controlled replicas based on the CPU usage
  - Stateless applications scaling (e.g. NoSQL and RDBMS)



ARCHITECTURAL DRIVERS

- **High Availability**
  - Workloads require availability at both the infrastructure and application levels.
  - **ReplicaSet** guarantees that a specified number of pod replicas are running at any time
  - **Replication Controller** is the same as ReplicaSet, but only supports the equality selector, while ReplicaSet supports the set-based selector.

ARCHITECTURAL DRIVERS

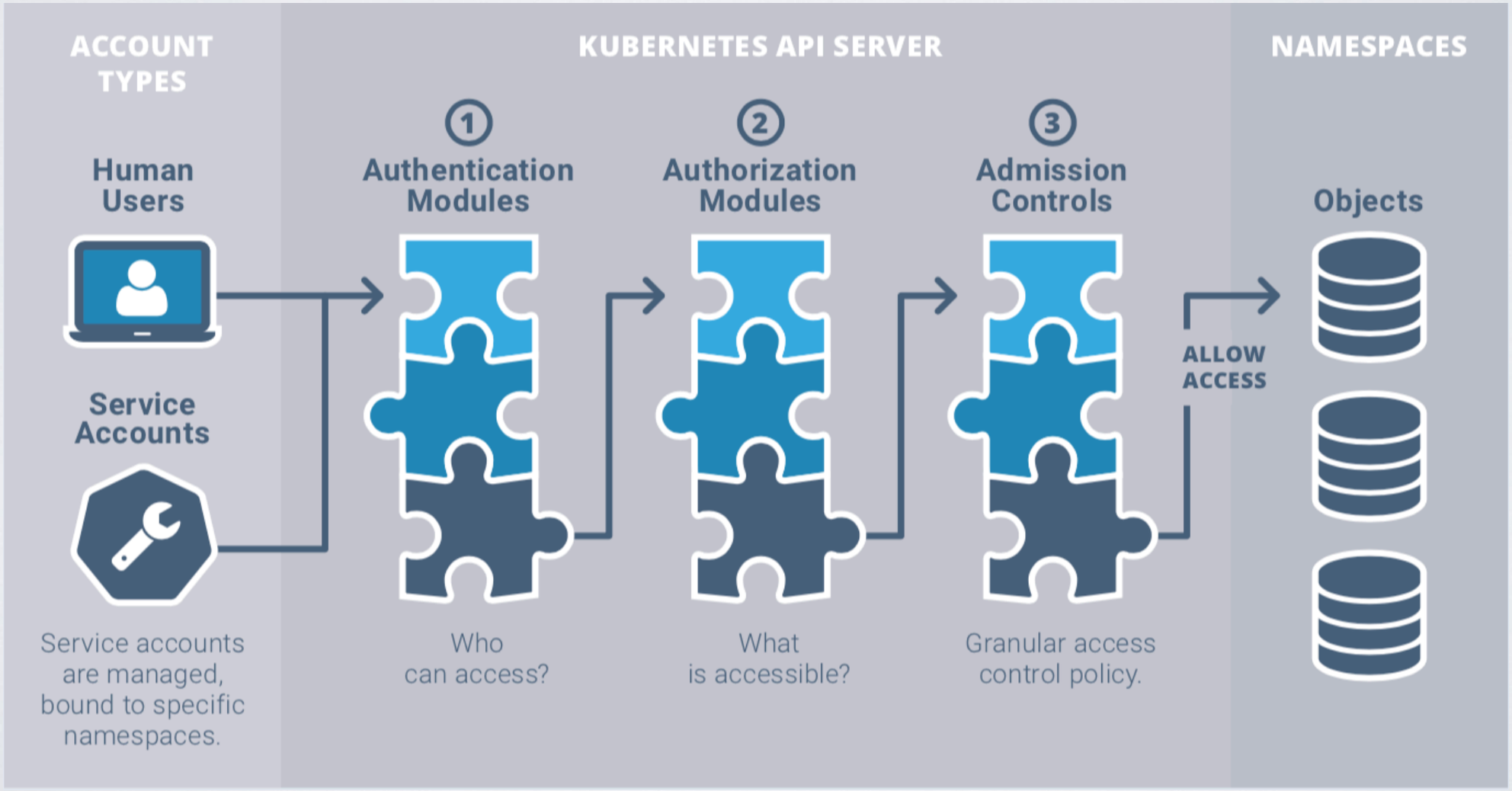
- **Portability**
  - Kubernetes is designed to offer freedom of choice over the choice of operating system to use, container execution, processor architecture, cloud platforms and PaaS.
  - **Federation v1** is an API that have Hybrid cloud capabilities: includes clusters running in different cloud providers

ARCHITECTURAL DRIVERS

- **Security**
  - Threats
    - External attacks
    - Compromised containers
    - Compromised credentials
    - Misuse of legitimate privileges

ARCHITECTURAL DRIVERS

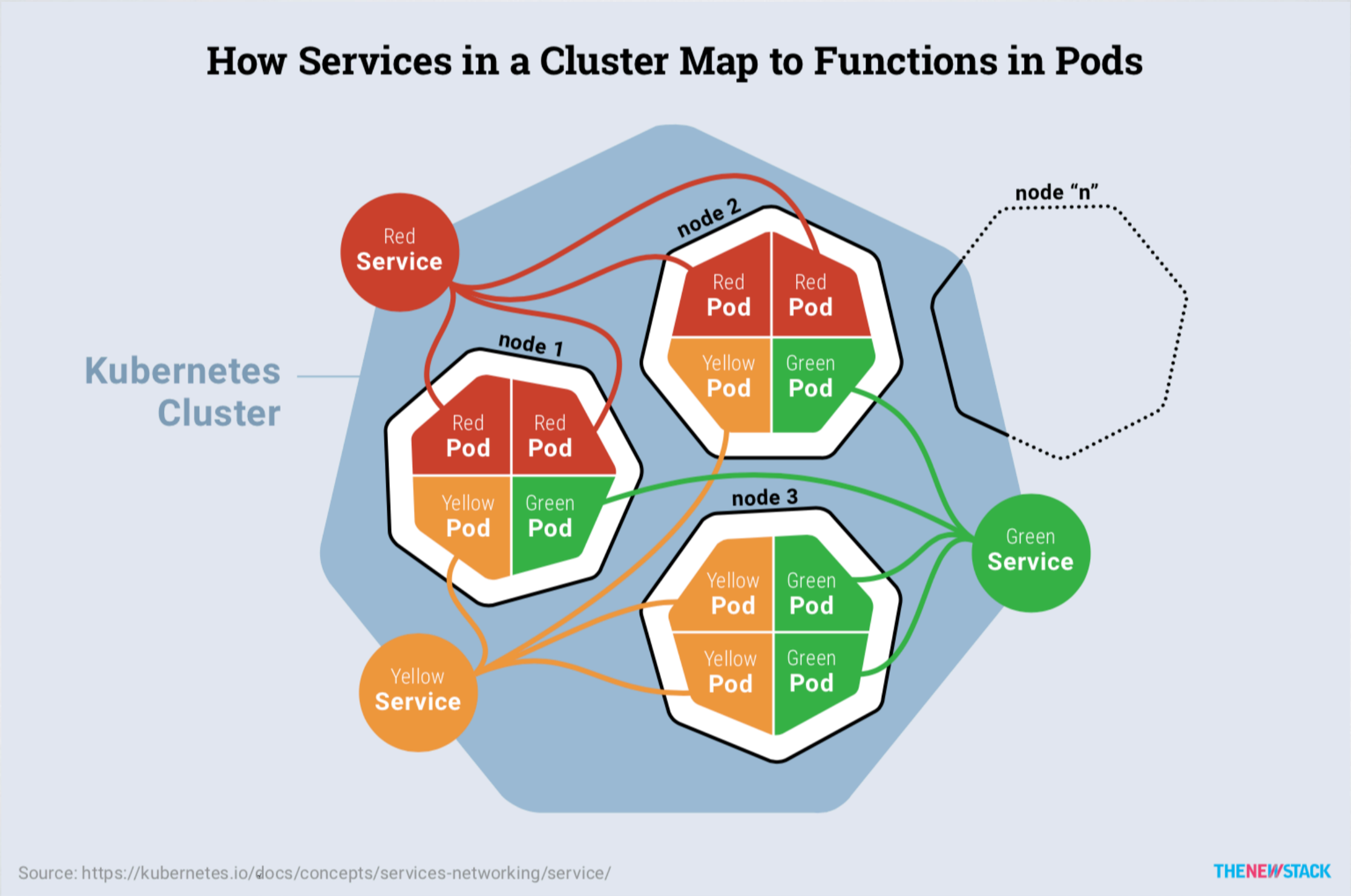
- **Security**
  - Authentication and authorization architecture



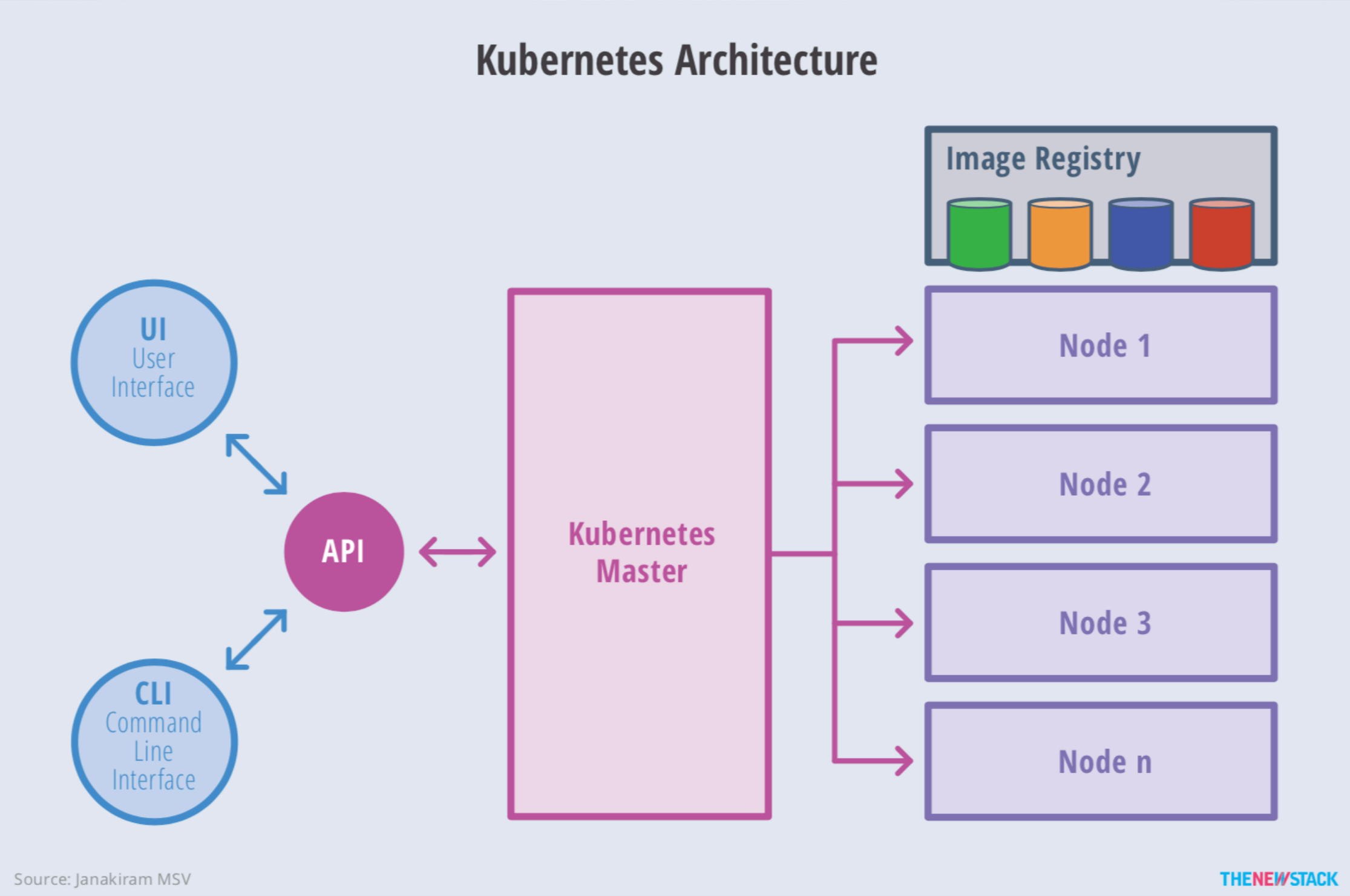
ARCHITECTURAL STRUCTURE

- **Service Oriented Architecture**
- A **pod** is a collection of one or more containers that serves as the main unit of Kubernetes for workload management
- **Labels** are key-value pairs associated with any object, including a node or pod, to identify and group objects that share a common attribute or property.
- A **selector** is a type of criterio used to query Kubernetes objects that correspond to a label value. This powerful technique allows to have a low coupling between objects
- **ReplicaSets** relies on labels and selectors to determine which pods will be scaled

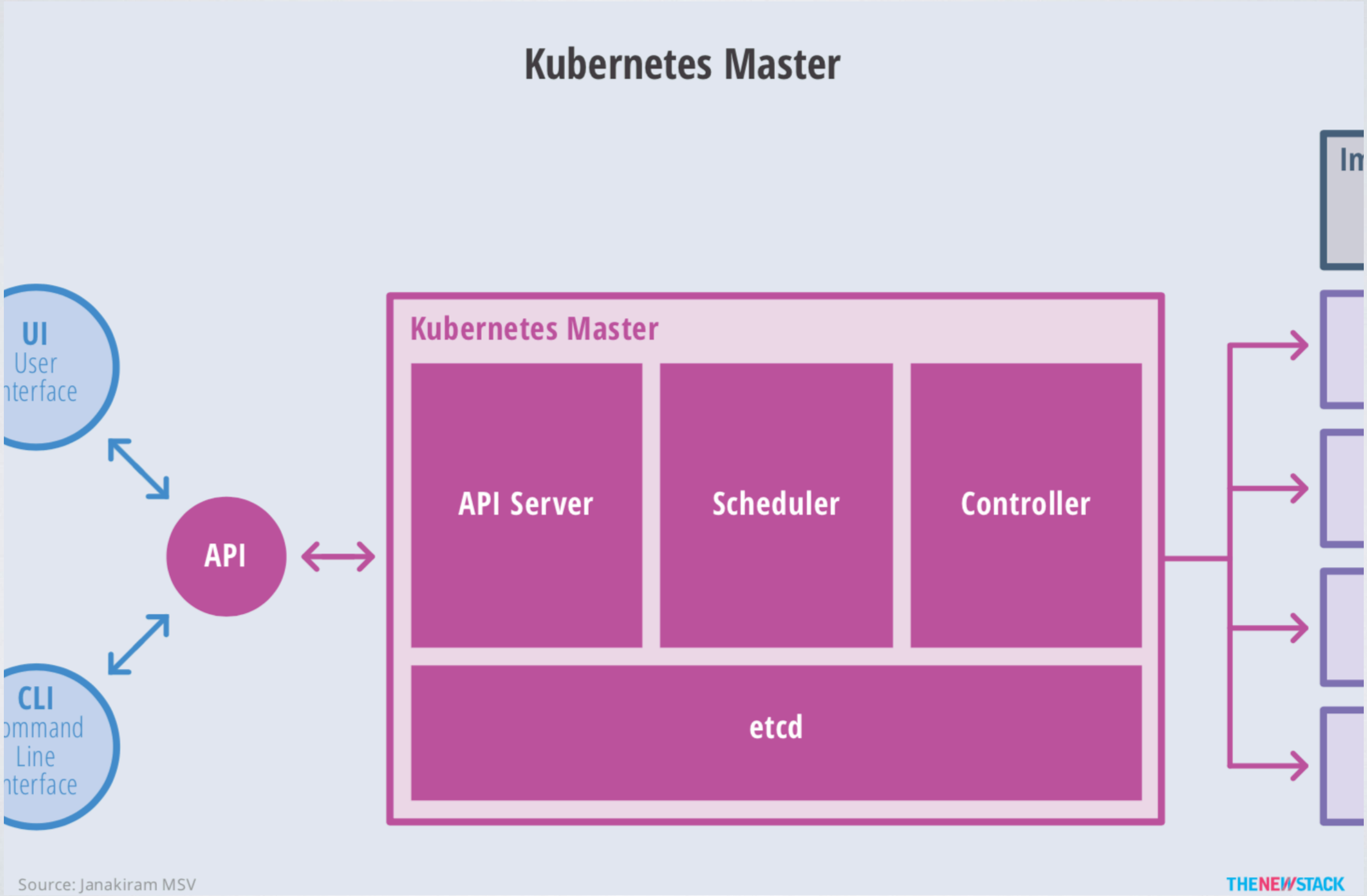
ARCHITECTURAL STRUCTURE



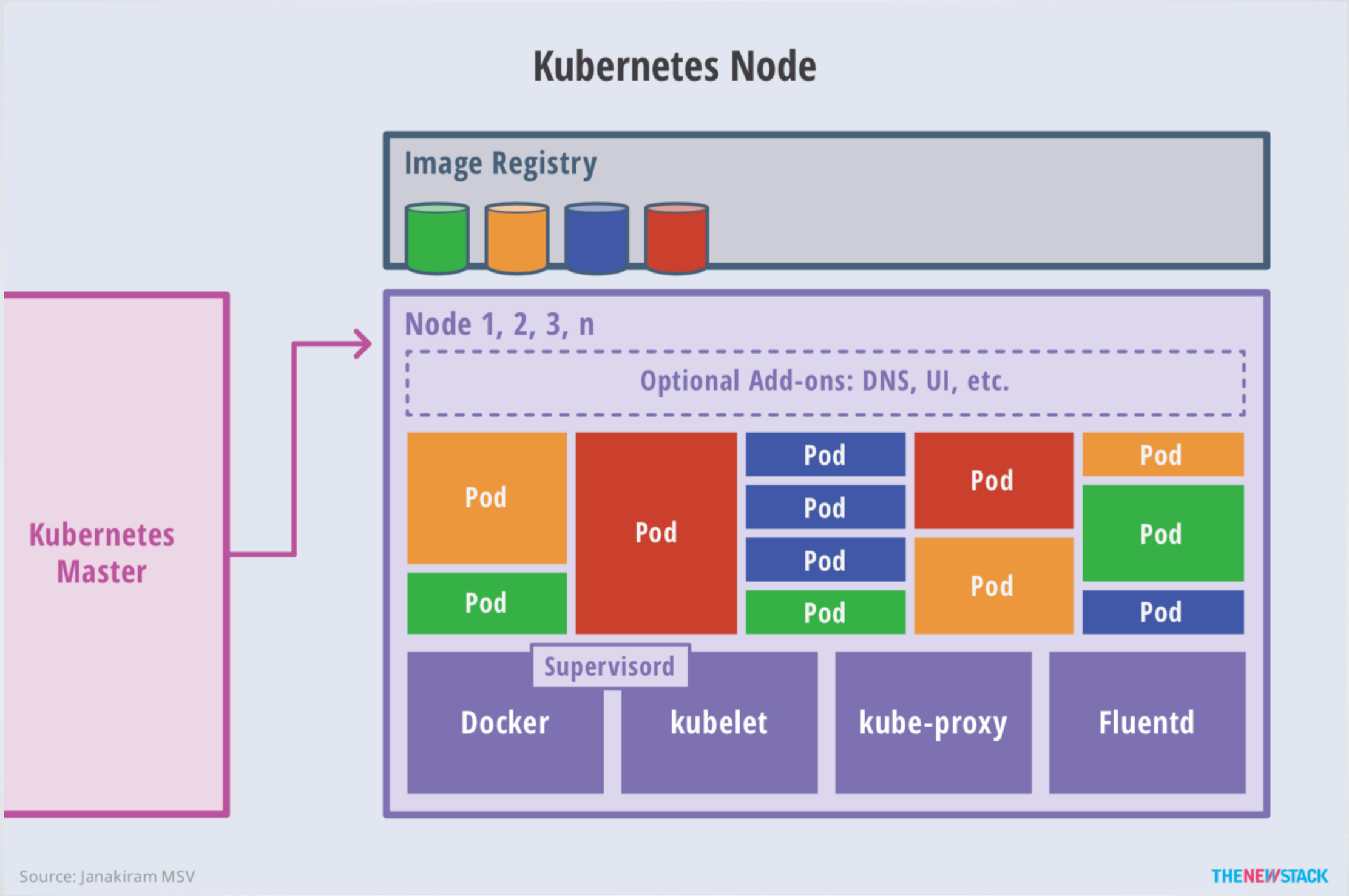
ARCHITECTURAL STRUCTURE



ARCHITECTURAL STRUCTURE



ARCHITECTURAL STRUCTURE



DEPLOYMENT PATTERNS

Features	**Self Hosted, Custom deployment**	**Managed Kubernetes Cluster**	**Container as a Service (CaaS)**	***Platform as a Service (PaaS)***
Ability to customize	🛠️🛠️🛠️ High	🛠️🛠️ Medium/High	🛠️🔧 Medium/Low	🛠️ Low
Overall costs (SW e Infrastructure)	💶💶💶 High	💶💶 Medium	💶💶💶 High	💶💶💶 High
Staffing and support costs	💶💶💶 High	💶💶 Medium	💶 Low	💶 Low
Admin skills	🛠️🛠️🛠️ High	🛠️🛠️ Medium	🛠️ Low	🛠️ Low
Portability around cloud	🛠️ Low	🛠️🛠️🛠️ High	🛠️🛠️ Medium (varies by providers)	🛠️🛠️ Medium (varies by providers)
Container image registry	❌ Not included	❌✅ Varies by provider	✅ Included	✅ Included
Security and monitoring services	❌ Not included	✅ Included	✅ Included	✅ Included
High-availability features	❌ Not included	❌ Not included	✅ Included	✅ Included
Built-in infrastructure auto-scaling	❌ Not included	❌ Not included	✅ Included	✅ Included
Complete app lifecycle management	❌ Not included	❌ Not included	❌ Not included	✅ Included

CODE QUALITY

- **Code duplication:** The developer used the same lines of code in several functions. This means that they have to change all code in corresponding functions whenever they need to change logic.
- **Total complexity:** The high number of complexity indicates that the code contains too much logic and should probably be broken down into smaller elements

Too many function arguments `kubelet.Kubelet.AttachContainer`

critical 8 arguments

Mute Report He

pkg/kubelet/kubelet_pods.go

```
1653 func (kl *Kubelet) AttachContainer(podFullName string, podUID types.UID, containerName string, stdin
1654     streamingRuntime, ok := kl.containerRuntime.(kubecontainer.DirectStreamingRuntime)
1655     if !ok {
1656         return fmt.Errorf("streaming methods not supported by runtime")
1657     }
1658
1659     container, err := kl.findContainer(podFullName, podUID, containerName)
1660     if err != nil {
1661         return err
1662     }
1663     if container == nil {
1664         return fmt.Errorf("container not found (%q)", containerName)
1665     }
1666     return streamingRuntime.AttachContainer(container.ID, stdin, stdout, stderr, tty, resize)
1667 }
```

UTILITY TREE

- **Utility**
  - **High Availability**
    - Application Fault tolerance
      - Il sistema deve garantire una replica completa del componente guasto in tempo < 3s **(M,H)**
      - I componenti guasti/malfunzionanti devono essere rilevati automaticamente dal sistema con tempo < 500ms **(M,H)**
    - Network Fault tolerance
      - Utilizzo di back-end di archiviazione per ripristinare il sistema allo stato precedente in < 3s **(H,M)**
  - **Security**
    - Data confidentiality
      - Le transazioni sono sicure al 99% **(H,L)**
      - L'autenticazione dell'utente è garantita il 99,99% delle volte **(H,L)**
    - Data integrity
      - Il sistema rileva in automatico la presenza di minacce di accesso a credenziali o container **(M,M)**
  - **Scalability**
    - Load balancing
      - In base all'utilizzo di risorse il sistema deve auto bilanciare il carico di lavoro 100% indipendente senza overloads **(M,H)**
    - Auto-scaling
      - Il sistema è 100% indipendente e non necessita dell'intervento dell'utente per la gestione delle risorse **(H,H)**
  - **Portability**
    - Adaptability
      - Il sistema deve potere adattarsi ad ogni esigenza supportando risorse complementari locali/cloud **(H,H)**
    - Installability
      - Il sistema è facilmente installabile e configurabile in tutti i SO **(L,H)**

KUBERNETES VS DOCKER SWARM

Scalability	Supporta cluster con un massimo di 100 nodi. Il 99% di tutte le chiamate API viene restituito in meno di 1 secondo Il 99% dei pod e i relativi container hanno un tempo d'avvio entro 5 secondi	Testato per supportare 1000 nodi e 30.000 container senza alcuna differenza evidente tra il tempo di avvio del 1 ° o 30.000°.
Availability	Altamente disponibile. I nodi guasti vengono rilevati e gestiti in automatico	Altamente disponibile. Lo Swarm Manager gestisce le risorse su larga scala.
Load-balancing	Bilanciamento del carico definendo pod come servizi	Bilanciamento del carico interno, ed esterno solo per servizi visibili
Auto-scaling	Eccellente.	Non supporta auto-scaling
Performance	L'architettura complessa ne rallenta le prestazioni	È 5x più veloce in termini di start di un cluster e scalabilità.
Easy to use	Difficile da configurare ed utilizzare	Indicato per implementazioni rapide e semplici

BIBLIOGRAPHY

- Alex Williams, The State of the Kubernetes Ecosystem, TheNewStack, 2017.
- *Alex Williams, Kubernetes Deployment e Security Patterns, TheNewStack, 2018.*
- *Alex Williams, Benjamin Ball, Gabriel Hoang Dinh, Lawrence Hecht, Use Cases for Kubernetes, TheNewStack, pp 7-13.*
- *Enreina Annisa Rizkiasri, Francisco Morales, Haris Suwignyo, Mohammad Riftadi, Kubernetes - Container Orchestration, Delft Students on Software Architecture, 2018*
- *Arsh Modak, Chaudhary, Paygude, Idate ,Techniques to Secure Data on Cloud: Docker Swarm or Kubernetes?, 2018 2nd International Conference, IEEE.*
- *Kate Matsudaira, Scalable Web Architecture and Distributed Systems, The architecture of Open Source Applications, vol 2.*