




Curso de **Kubernetes**

Marcos Nils Lilljedahl



Marcos Nils Lilljedahl (@marcosnils)
VP of Eng @ iúnigo
(<https://iunigo.com.ar>)

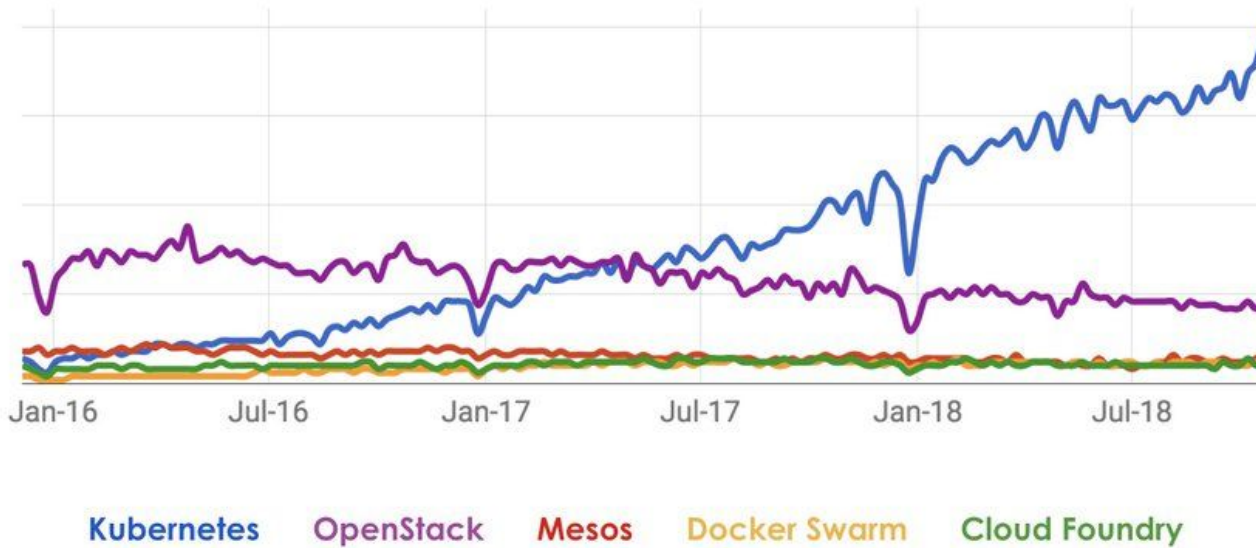
- Ing. en sistemas / MoIT
- OSS & Golang ❤️
- Docker Captain 
- Docker global hackday #3
(<https://blog.docker.com/2015/09/docker-global-hack-day-3-winners/>)
- Keynote cierre DockerCon EU 2015
(<https://www.youtube.com/watch?v=ZBcMy-xuYk>)
- Co creador de play-with-docker.com y play-with-kubernetes.com junto a @xetorthio
- Crossfitter

Objetivos del curso.

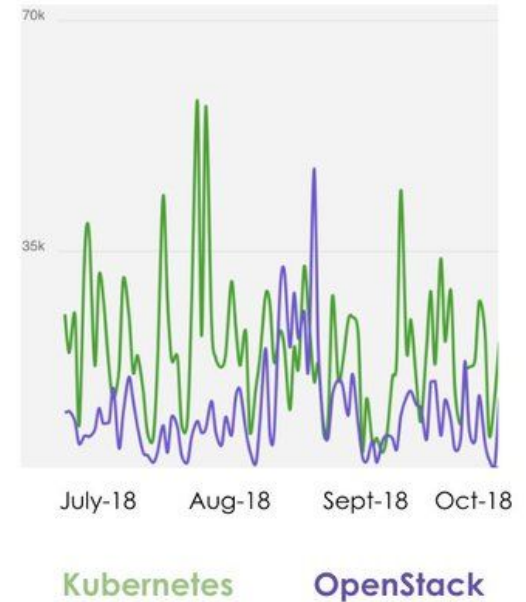
- Visión y entendimiento de la plataforma de Kubernetes
- Uso y buenas prácticas de las distintas herramientas
- Workflow aplicativo sobre la plataforma de Kubernetes

Kubernetes in Search Trends

Google Trends



WeChat



Notas

- El hashtag oficial del curso es #PlatziK8s
- Utiliza las herramientas de la plataforma para enviar tus consultas
- Haz el curso tuyo, el objetivo es que puedan poner en práctica todo lo aprendido de manera **inmediata**

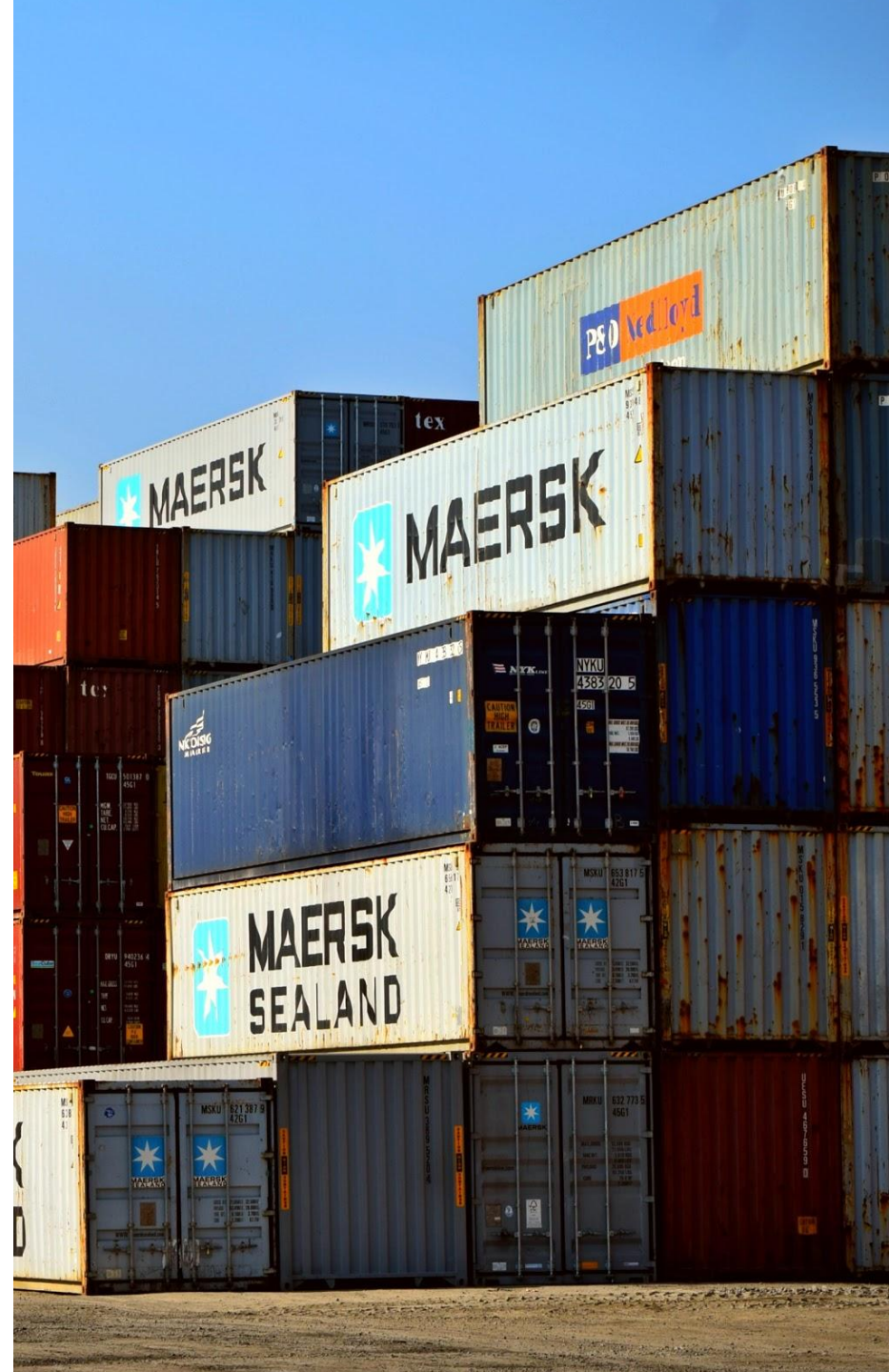
Pre-requisitos

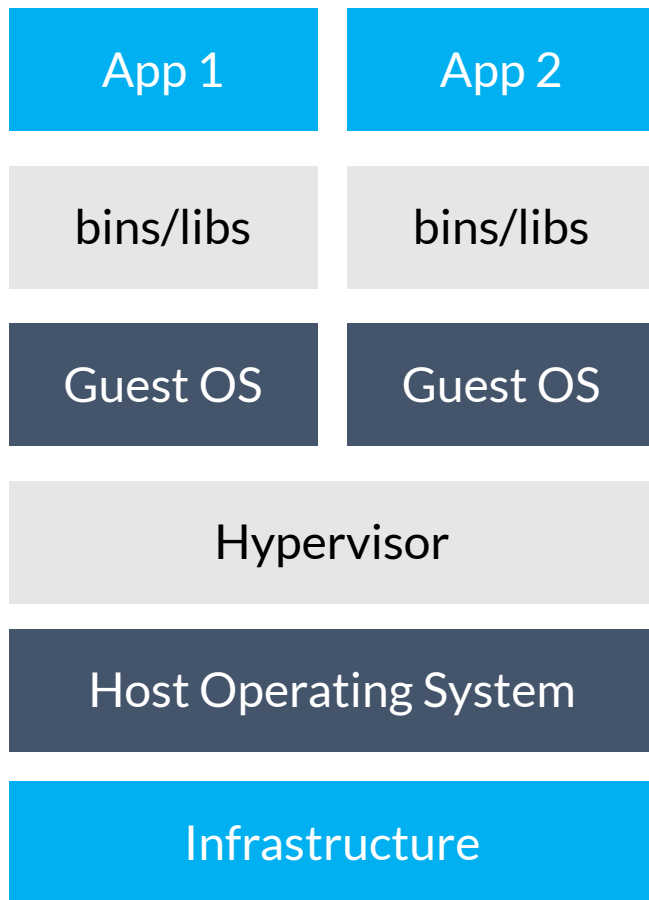
- Estar cómodos con UNIX command line
- Conocimientos básicos de Docker (no te preocupes, vamos a repasar los conceptos básicos)

Contenedores

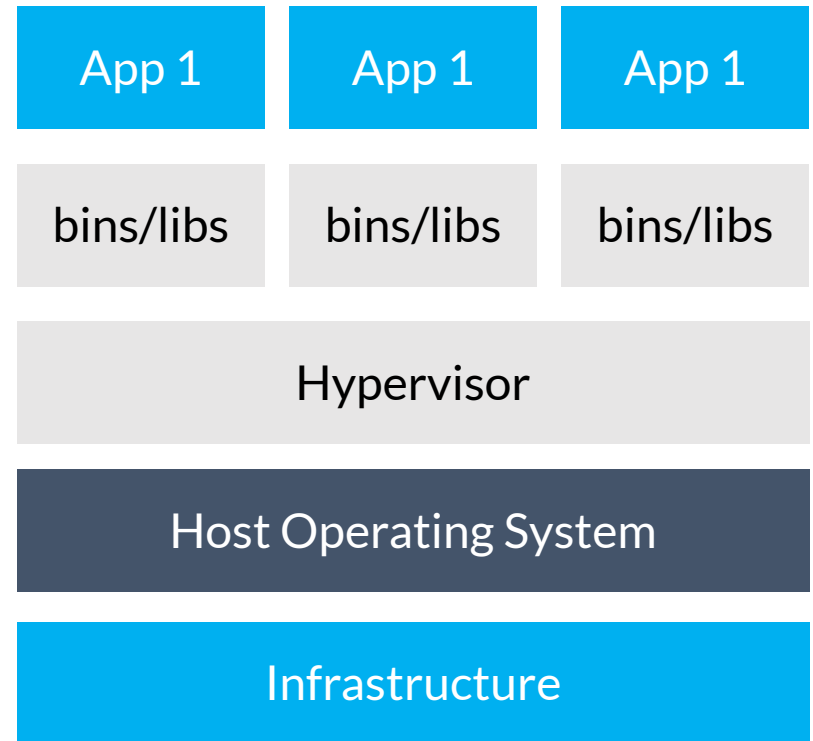
Contenedor = namespaces
+ cgroups + chroot + ...

- Namespaces: Vistas de los recursos del SO
- Cgroups: Limitan y miden los recursos del SO
- Chroot: Cambia el root directory de un proceso





Virtual Machines



Containers

Docker & Kubernetes

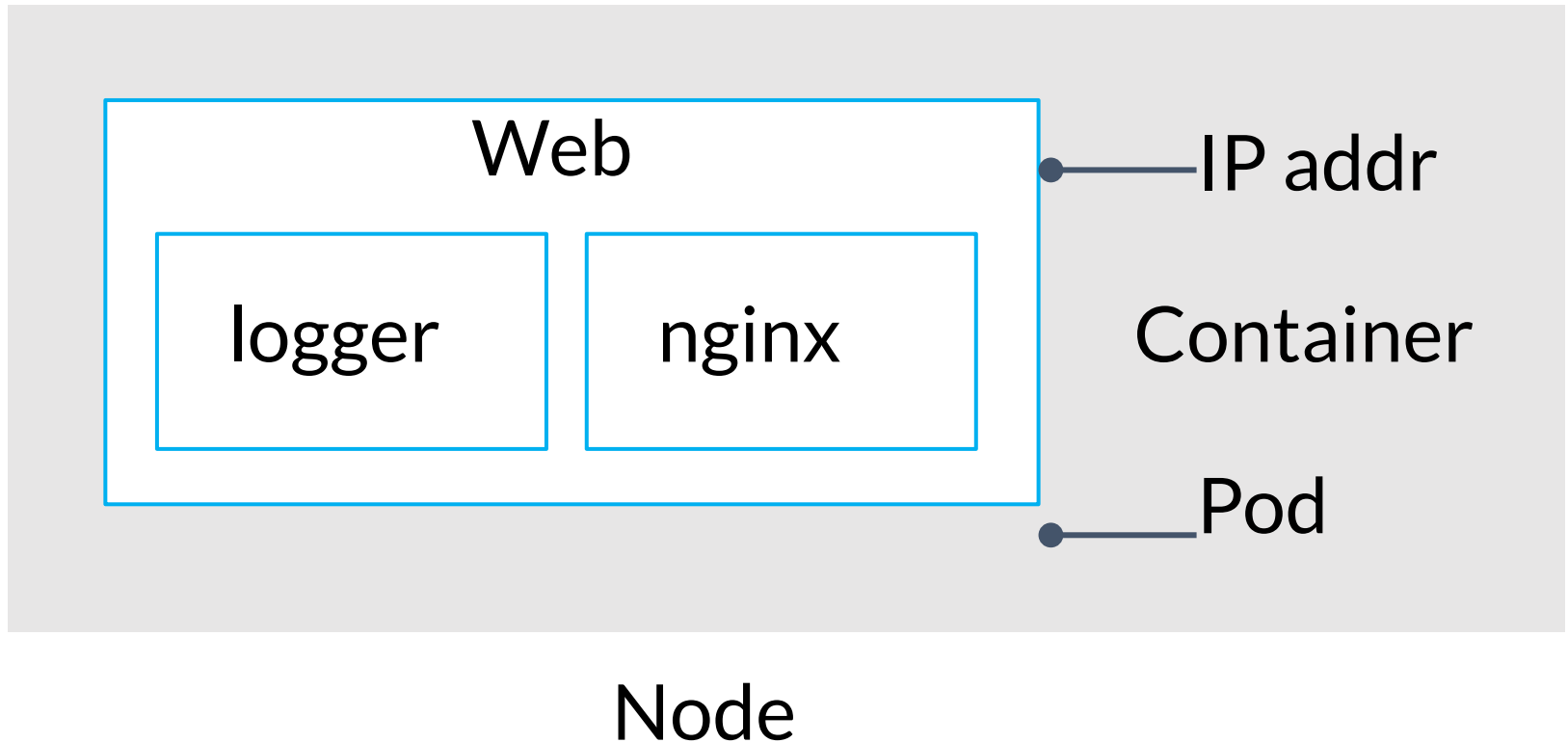
- Docker se encarga principalmente de gestionar los contenedores
- Kubernetes es una evolución de los proyectos de Google Borg & Omega
- Kubernetes pertenece a la CNCF (Cloud Native Computing Foundation)
- Todos los cloud providers (GCP / AWS / Azure / DO) ofrecen servicios de managed k8s utilizando Docker como su container runtime
- Es la plataforma más extensiva para orquestación de servicios e infraestructura

Kubernetes en práctica (in-practice)

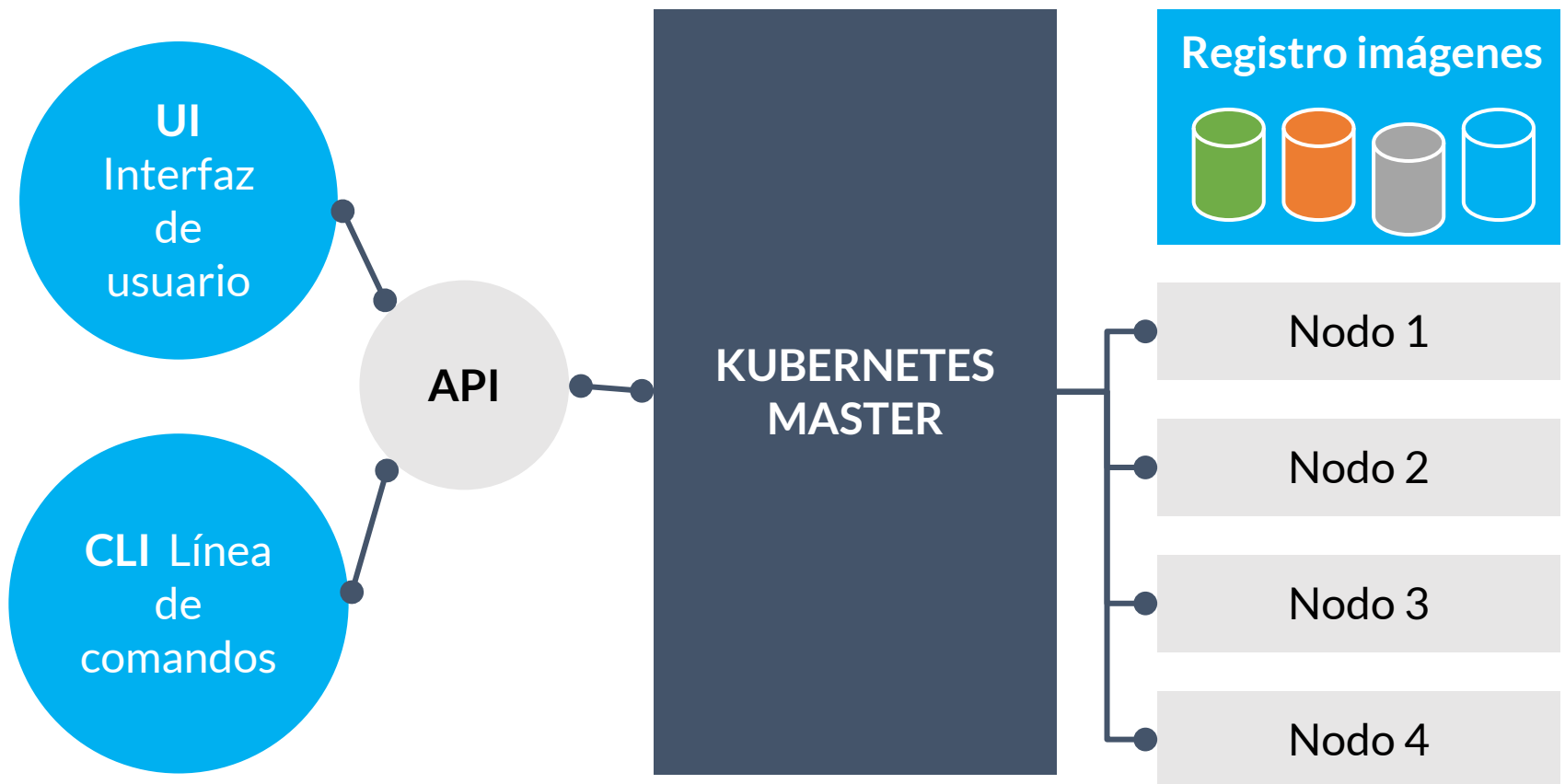
- Correr varias réplicas y asegurarse que todas se encuentren funcionando
- Proveer un balanceador de carga (interno o externo) automáticamente para nuestros servicios
- Definir diferentes mecanismos para hacer roll-outs de código
- Políticas de scaling automáticas
- Jobs batch
- Correr servicio con datos stateful
- Y muchas otras cosas (CRDs, Service catalog, RBAC)

Recursos de Kubernetes

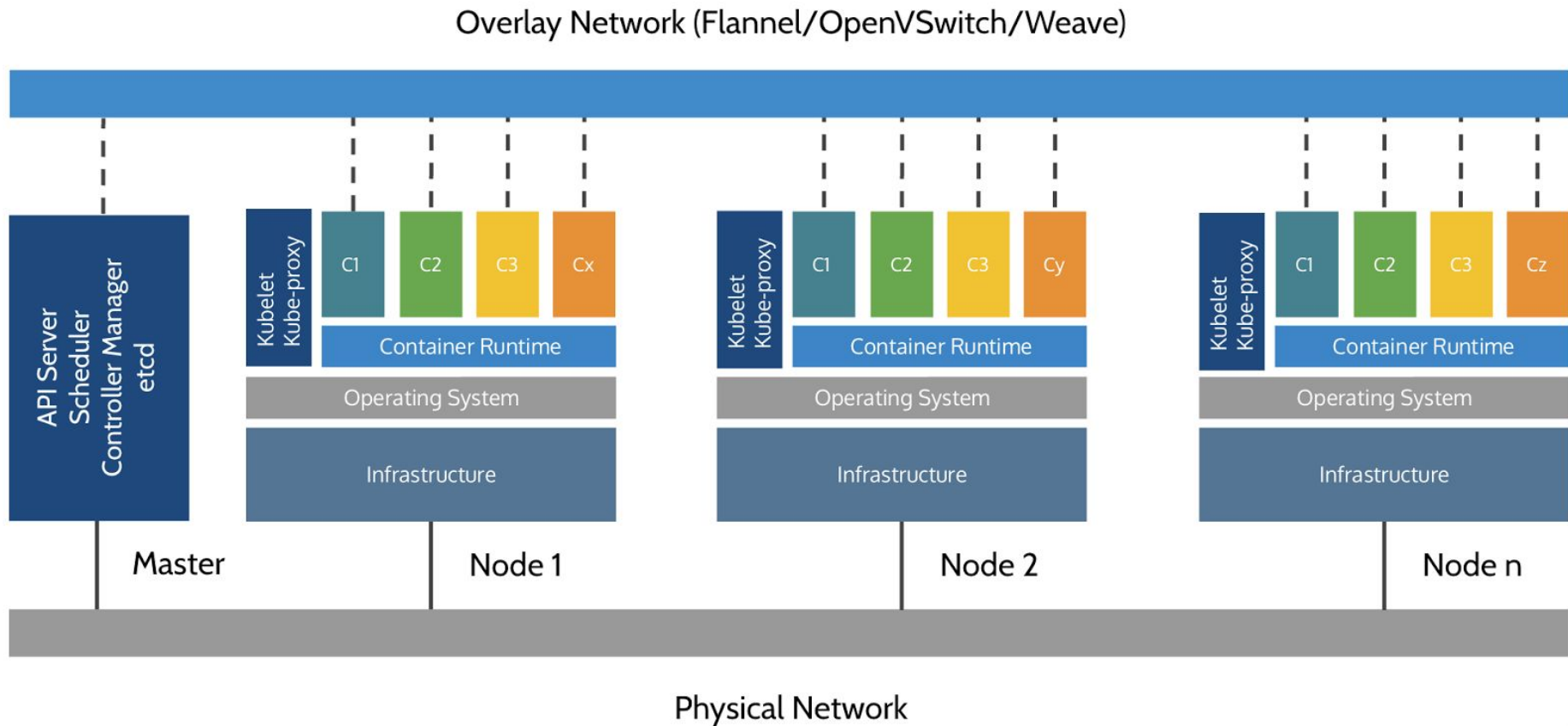
Conceptos



Arquitectura de Kubernetes



Arquitectura de Kubernetes



Kubernetes-Declarativo vs Imperativo

- Kubernetes hace énfasis en ser un sistema declarativo
- Declarativo: “*Quiero una taza de té*”
- Imperativo: “*Hervir agua, agregar hojas de té y servir en una taza*”
- Declarativo parece sencillo (siempre y cuando uno sepa cómo hacerlo)
- Todo en K8s se crea desde un *spec* que describe cuál es el estado deseado del sistema
- Kubernetes constantemente converge con esa especificación

Kubernetes-Networking

- Todo el cluster es una gran red del mismo segmento
- Todos los nodos deben conectarse entre sí, sin NAT
- Todos los pods deben conectarse entre sí, sin NAT
- kube-proxy es el componente para conectarnos a pods y contenedores (userland proxy / iptables)
- Los pods trabajan a capa 3 y los servicios a capa 4
- Concepto de CNI (container networking interface)

On prem

- Deploy yours

Cloud

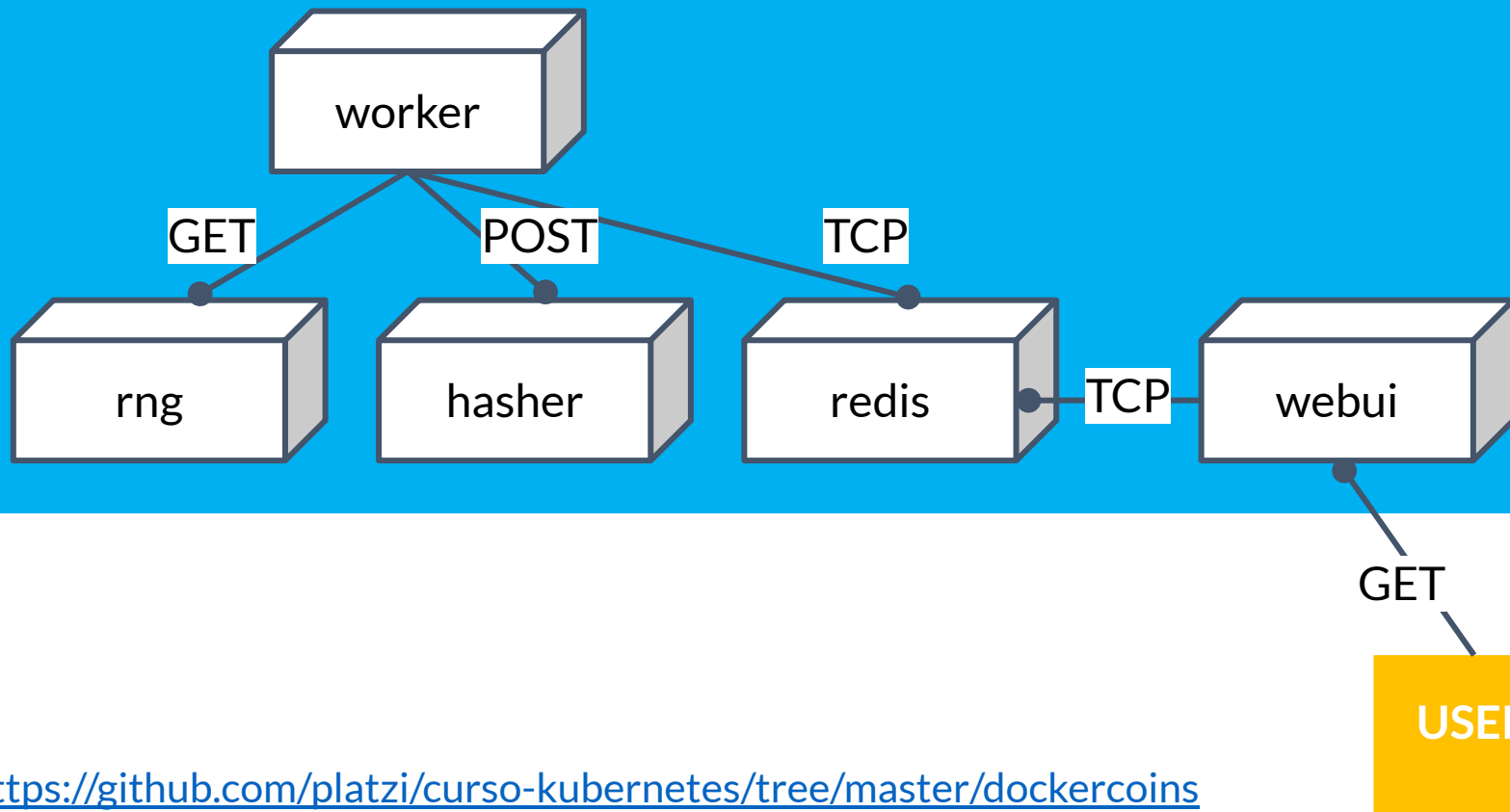
- Hosted options:
 - AWS EKS
 - Azure AKS
 - Google GKE

Aplicación de prueba

- Clonar el repo aplicativo
- Iniciar la aplicación con “docker-compose up”
- Docker compose va a buildear las imágenes e iniciar los servicios
- Conectarse al puerto :8080 local y navegar la UI
- * Notar la velocidad del hashing (~ 4/sec)

Aplicación de prueba

DockerCoins application (five containers)



<https://github.com/platzi/curso-kubernetes/tree/master/dockercoins>

Desplegar cluster de Kubernetes

<https://github.com/kelseyhightower/kubernetes-the-hard-way>

Kubectl

Creando nuestro primer POD

Pods / Deployment / Replica Set

Accediendo a los contenedores

- Cluster IP:
Una IP virtual es asignada para el servicio
- NodePort
Un puerto es asignado para el servicio en todos los nodos
- LoadBalancer
Un balanceador externo es provisionado para el servicio.
Sólo disponible cuando se usa un servicio con un balanceador
- ExternalName
Una entrada de DNS manejado por CoreDNS

Servicios y endpoints

Nuestra App en K8s

Kubectrl proxy & port-forward

Kubernetes dashboard

<https://blog.heptio.com/on-securing-the-kubernetes-dashboard-16b09b1b7aca>

Escalar un deployment & Daemon sets

Rolling updates

Healthchecks

Helm

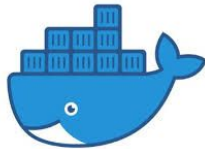
Gestionando configuraciones

- Nuestras aplicaciones suelen requerir de configuraciones
- Diferentes formas de configuraciones
 - Argumentos por línea de comandos ([fuente](#))
 - Variables de entorno (env map en el spec)
 - Archivos de configuración (config maps)
 - Guardan tanto archivos como valores clave/valor.
- Exponer las configuraciones tanto como archivos o variables de entorno

Config Maps

Volúmenes

Volúmenes: Docker y K8s



Permiten compartir información entre contenedores del mismo host

Permiten compartir información entre contenedores del mismo POD

Permiten acceder a mecanismos de storage

Permiten acceder a mecanismos de storage

Docker config y docker secrets

Se utilizan para el manejo de secretos y configuraciones

Volúmenes: Ciclo de vida

- Está relacionado al ciclo de vida de los PODS
- El volumen se crea cuando el POD se crea
 - Esto aplica principalmente para los volúmenes **emptyDir**.
 - Para otro tipo se conectan en vez de crearse
- Un volumen se mantiene aún cuando se reinicie el contenedor
- Un volumen se destruye cuando el POD se elimina

Namespaces

Namespaces

- No provee aislamiento de recursos
- Un pod en un namespace A puede comunicarse con otro en un namespace B
- Un pod en el namespace **default** puede comunicarse con otro en el **kube-system**
- Desde cualquier pod en el cluster podemos comunicarnos al k8s API
(<https://kubernetes.default.svc.cluster.local:443/>)

Autenticación y autorización

- Cuando el API server recibe un request intenta autorizarlo con uno o más de uno de los siguientes métodos: Certificados TLS, Bearer tokens, Basic Auth o Proxy de autenticación
- Si cualquier método rechaza la solicitud, se devuelve 401 (unauthorized).
- Si el request no es aceptado o rechazado, el usuario es anónimo
- Por defecto el usuario anónimo no puede hacer ninguna operación en el cluster.

Autenticación vía Tokens

- Los tokens son enviados en cabeceras http (Authorization: Bearer token)
- Los tokens pueden ser validados con diferentes métodos:
 - estáticos quemados en el API server
 - bootstrap tokens (especiales para la creación del cluster)
 - OpenID tokens (delegar autenticación a proveedores OAuth)
 - Service accounts

Service account tokens

- Existen en la API de kubernetes. (kubectl get serviceaccounts)
- Pueden crearse / eliminarse y actualizarse
- Un service account está asociado a secretos (kubectl get secrets)
- Son utilizados para otorgar permisos a aplicaciones y servicios

RBAC (Role based access control)

- Nos permite especificar permisos de manera granular
- Los permisos son expresados como reglas
- Las reglas son una combinación de:
 - Verbos como: create, get list, update, delete...
 - Recursos como: pods, nodes, services....
 - Nombres de recursos específicos
 - Subrecursos (ej: logs a pods)

RBAC (Role based access control)

- Un *role* es un objeto que contiene una lista de *rules*
 - Ej. el rol “external-loadbalancer-config” puede:
 - List, get en recursos endpoints, services y pods
 - update en recursos servicios
- Un *rolebinding* asocia un rol a un usuario
- Pueden existir usuarios ,roles y rolebindings con el mismo nombre
- Una buena práctica es tener 1-1-1 bindings
- Los Cluster-scope permissions permiten definir permisos a nivel cluster y no sólo namespace
- Un pod puede estar asociado a un service-account
 - El token se encuentra en (/var/run/secrets)

Próximos pasos

- Establecer una cultura de containers en la organización
 - Escribir Dockerfiles para una aplicación (o servicio)
 - Escribir compose files para describir servicios
 - Mostrar las ventajas de correr aplicaciones en contenedores
 - Configurar builds automáticos de imágenes
 - Automatizar el CI / CD (staging) pipeline
- Developer experience
 - On board / tools / deploy / components
- Elección de un cluster de producción
 - Cloud / Managed / Self-Managed
 - Un cluster grande o múltiples pequeños
- Recordar el uso de namespaces

Próximos pasos (cont...)

- Servicios con estado (stateful)
 - Intentar evitarlos al principio
 - Técnicas para exponerlos a los PODS (ExternalName / ClusterIP / Ambassador)
 - Storage provider , Persistent Volumes y Stateful set
- Gestión del tráfico HTTP
 - Ingress controllers (virtual host routing)
- Configuración de la aplicación
 - Secretos y config maps
- Stack deployments
 - GitOps (infrastructure as code)
 - Helm / Spinnaker / Brigade

Muchas gracias

[@marcosnils](#)
