# IT Industry Challenges

José Ángel Morena

# Índice

# 1. Administración de sistemas en el mundo profesional

- Que es la administración
- Perfiles y tareas
- Tecnologías importantes

# 1. Administración de sistemas en el mundo profesional

Professional profiles:

- DBA
- Network sysadmin
- Security admin
- Technical support staff
- System admin
- Hardware (datacenter people).

Responsabilidades (tradicionalmente):

- Comprobaciones diarias de sistemas/software.
- Realización de copias de seguridad de datos.
- Aplicación de actualizaciones del sistema operativo y cambios de configuración.
- Instalación y configuración de nuevo hardware/software.
- Agregar/eliminar/crear/modificar información de cuenta de usuario, restablecer contraseñas, etc.
- Responder consultas técnicas.
- Responsabilidad por la seguridad.
- Responsabilidad de documentar la configuración del sistema y arquitectura de las diferentes soluciones.
- Solución de problemas de cualquier problema informado o problemas informados.
- Ajuste del rendimiento del sistema.
- Mantener la red en funcionamiento. (red corporativa, VPN, DNS, DHCP, etc.)

# 1. Administración de sistemas en el mundo profesional

- Sin infraestructura las aplicaciones no pueden correr.
- Es vital entender cómo funcionan los entornos.
- Troubleshooting.
- Automatizar operaciones.
- Ejemplos de posts de empleo en Linkedin.

Tecnologías importantes (tradicionalmente):

- Linux.
- Network skills. (HTTP, TCP, UDP, DNS, DCHP, firewalls, IPAM, etc.)
- Cloud and hypervisors.
- Bases de datos.
- Containers.
- Observabilidad.
- Integración contínua y Despliegue contínuo.
- Windows (Windows AD, domain controller).

# 1. Administración de sistemas en el mundo profesional

**Operacional** (Mantenimiento):

- Creación de records DNS
- Actualización de servidores y reinicios programados.
- Creación de recursos en hipervisores/cloud-providers.
- Mantenimiento.
- Network ACL
- Incidencias.
- etc.

**Engineering** (Implementación de soluciones, mejoras, documentación):

- Despliegues de entornos
- Automatización
- etc.

**Training.** Es importante estar actualizado:

- https://www.redhat.com/es/services/training-and-certification
- https://www.cncf.io/certification/training/
- https://www.elastic.co/es/training/
- https://www.cisco.com/c/en/us/training-events/training-certifications/certifications.html
- etc.

# 2. Retos que enfrentan los departamentos de sistemas

Problema 1: Departamento de sistemas, cuello de botella.

Problema 2: Brecha entre el departamento de sistemas y desarrollo.

# 2. Retos que enfrentan los departamentos de sistemas

## Problema 1: Cuello de botella

Cuello de botella asegurado:

- # > 100 equipos de desarrollo.
- 1 equipo de administración.

Algunas posibles soluciones:

- Escalar verticalmente (añadir más recursos).
- **Escalar horizontalmente (crear más equipos con diferentes responsabilidades).**
- **Productización.**
- **Self-Service.**

# 2. Retos que enfrentan los departamentos de sistemas

**Problema 2: Brecha entre el departamento de sistemas y desarrollo.**

- Poco entendimiento y dependencias implica menor productividad.

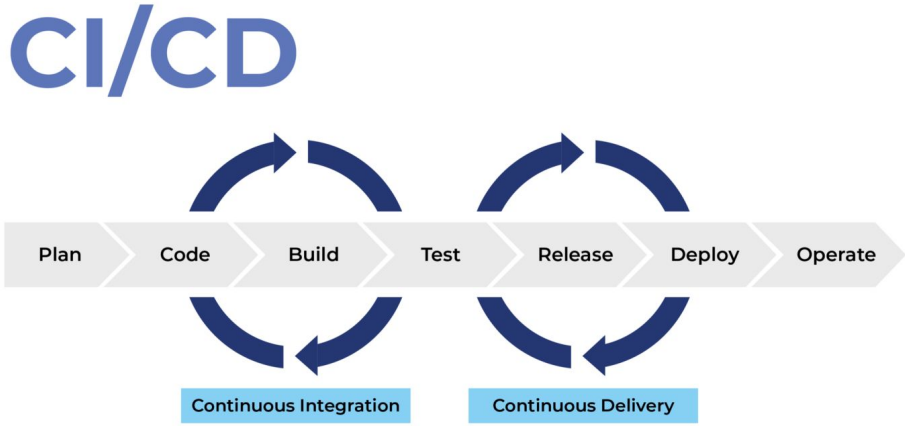
Soluciones:

- DevOps
- Site Reliability Engineering

# 3. Soluciones actuales a los problemas

DevOps:

- Movimiento cultural.
- Perfil híbrido.
- Agilidad.



# 3. Soluciones actuales a los problemas

SRE (Site Reliability Engineering):

- Gestionar operaciones como un problema de software
- Automatización como un producto
- Gestionar miles de recursos de forma transparente.

# 3. Soluciones actuales a los problemas

DevOps vs SRE:

- DevOps es un movimiento cultural
- SRE puede considerarse una implementación del DevOps
- DevOps se centra en mejorar el proceso de desarrollo
- SRE se centra más en gestionar eficientemente recursos.

# 3. Soluciones actuales a los problemas

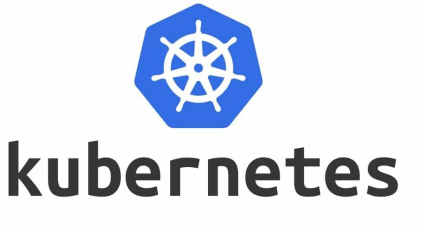
DevOps y SRE:

- Ambos perfiles requieren background en operaciones y desarrollo.
- Actualmente cada empresa lo diferencía desde su punto de vista.
- Ambos son fundamentales en las organizaciones.

https://www.linkedin.com/jobs/search/?keywords=sre
https://www.linkedin.com/jobs/search/?keywords=devops

# 4. Containers, Kubernetes y Cloud

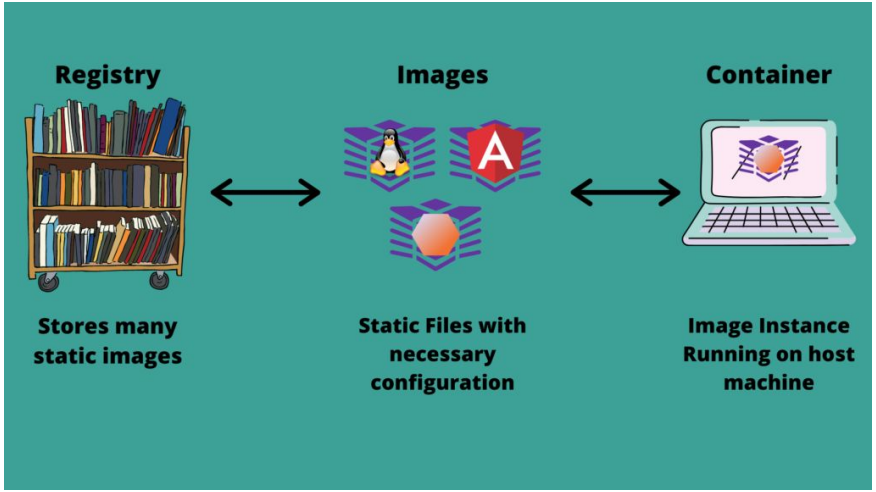
- Nuevos conceptos implican nuevas tecnologías
- Kubernetes, el presente y el futuro






# 4. Containers

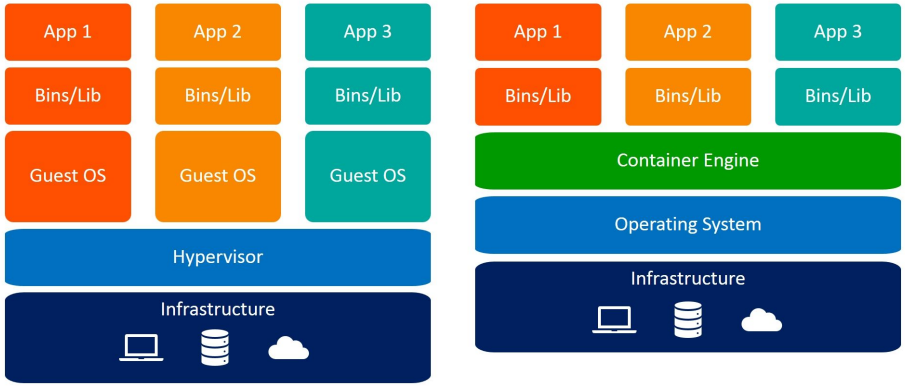
- Un contenedor de Linux es un proceso
- Aplicación empaquetada junto a sus dependencias
- Aislamiento de procesos (Linux namespaces)
- Control de recursos (CGroups)



# 4. Containers

Ventajas:

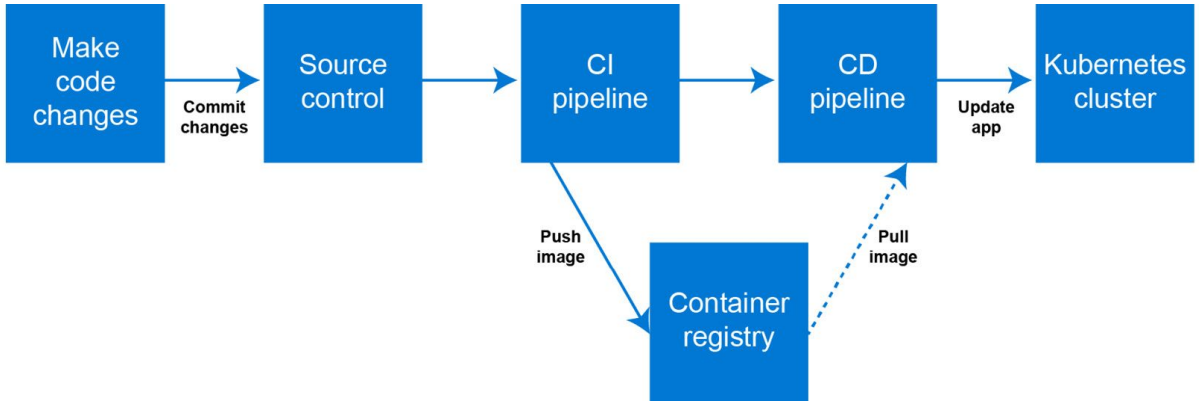
- Acelerar el despliegue de aplicaciones
- Menor consumo de recursos
- Mayor portabilidad
- Mayor velocidad



# 4. Containers

Casos de uso:

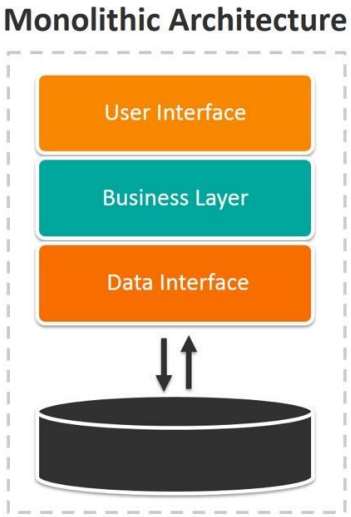
- Lift and Shift
- Solución adecuada para microservicios
- Solución adecuada para pipelines CI/CD
- Solución para ejecución de trabajos repetitivos.

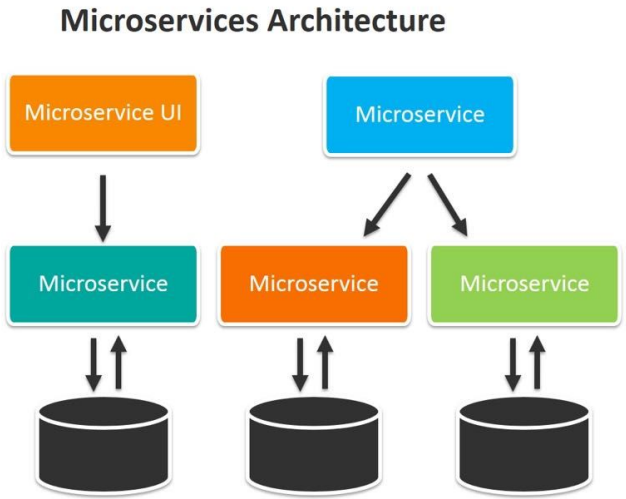


# 4. Containers

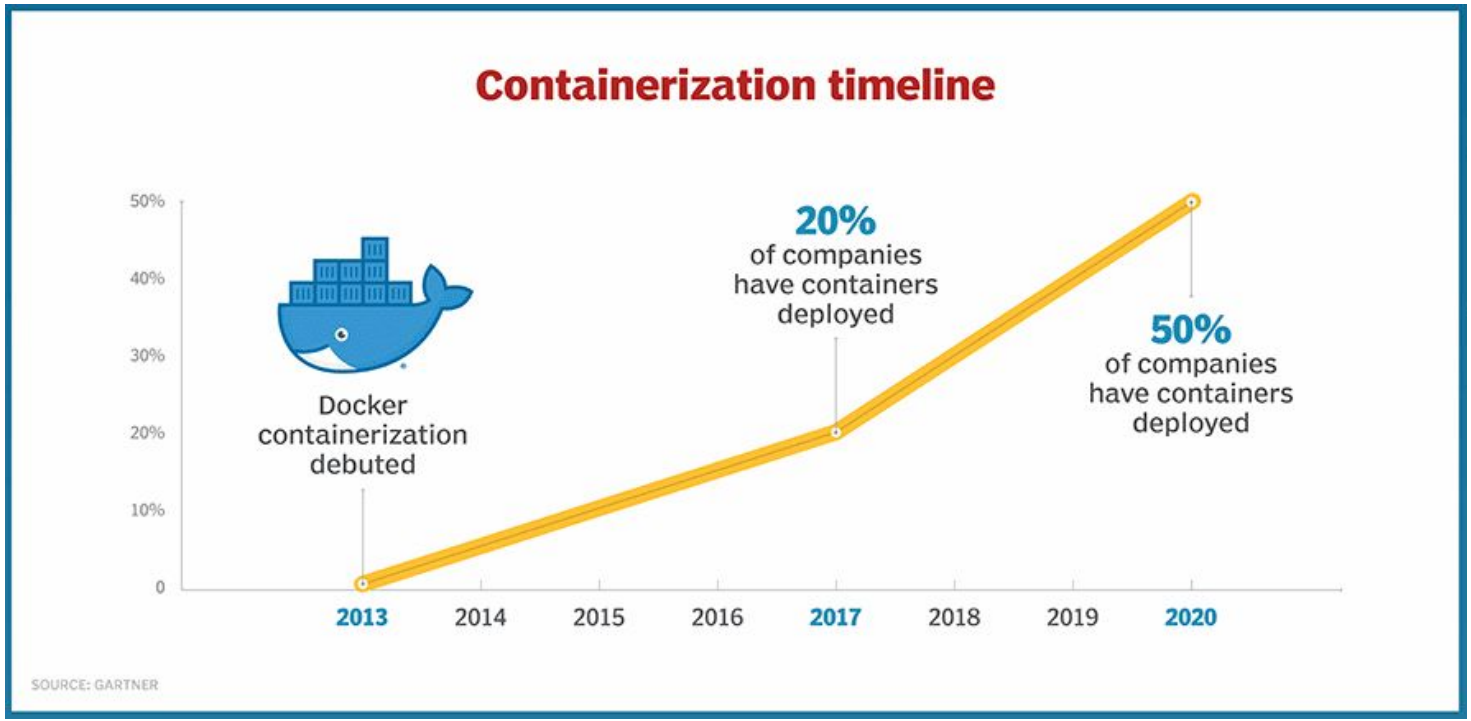
Microservicios:

- Arquitectura de software.
- Divide y vencerás.



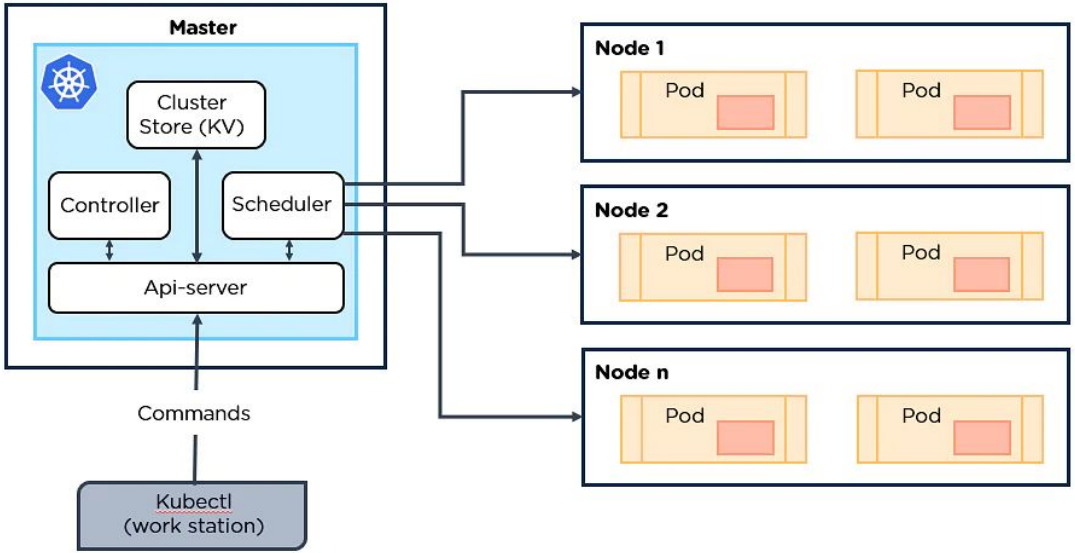



# 4. Adopción Containers



# 4. Kubernetes

- Etimología: Proviene del Griego, significa piloto
- Google liberó a la comunidad **el proyecto** Kubernetes en 2014
- Orquestador de contenedores (**estilo declarativo**)



# 4. Kubernetes

Ventajas:

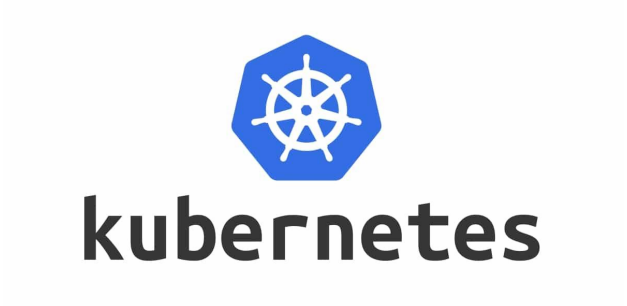
- Scheduling automático
- Capacidades Self-Healing
- Estilo declarativo
- Escalado horizontal y balanceo de carga
- Entornos consistentes

Desventajas:

- Altamente complejo
- Difícil de administrar
- Es un proyecto no un producto

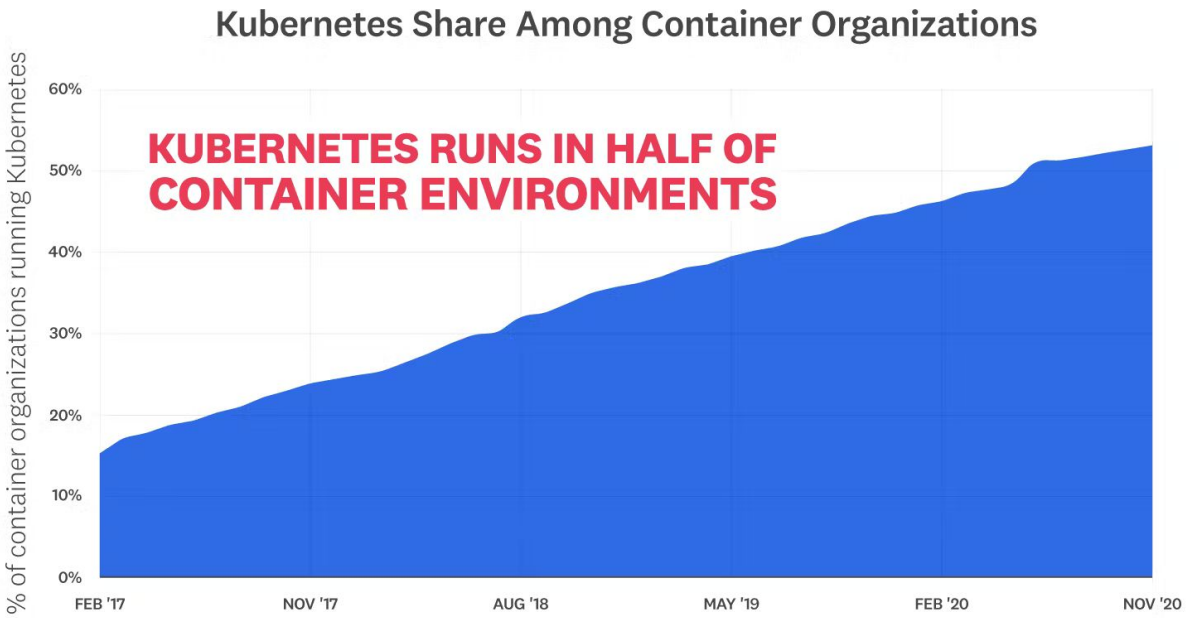
Casos de uso:

- Gestionar miles de contenedores





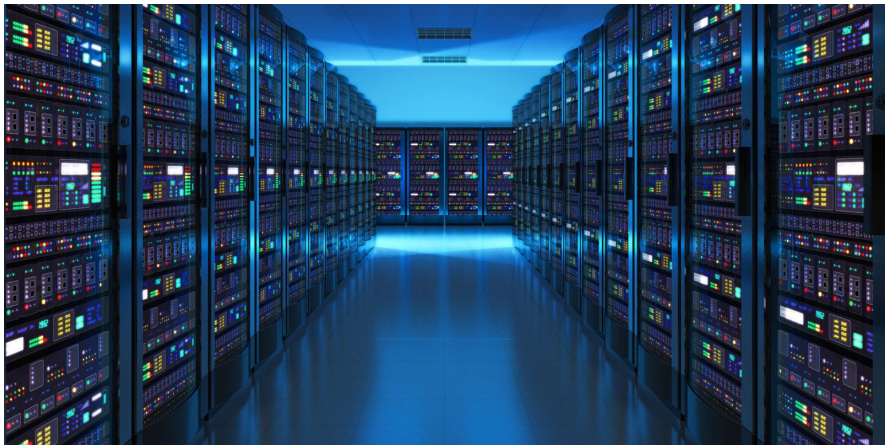
# 4. Adopción Kubernetes




*Source: Datadog*

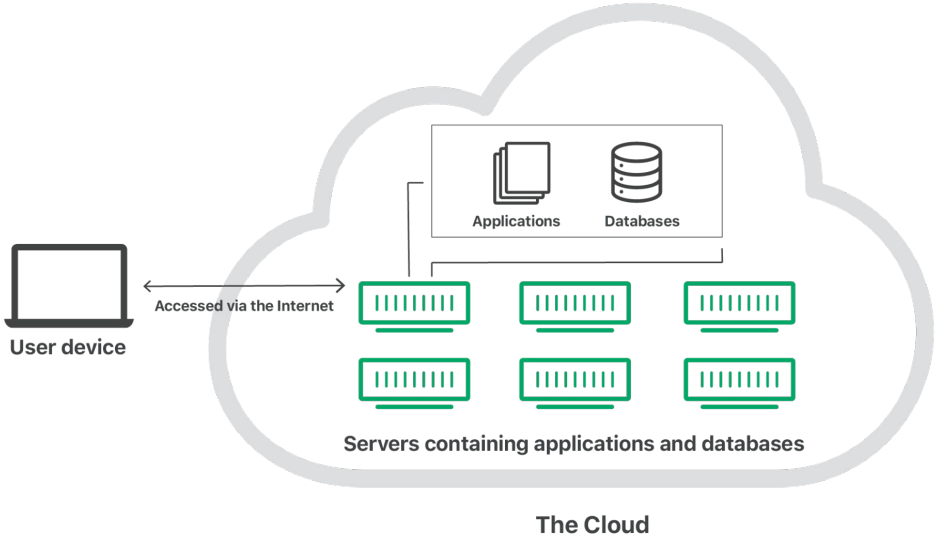
# 4. Cloud

- Proveedores de cloud (AWS, Azure, GCP, etc.) venden sus recursos por Internet.



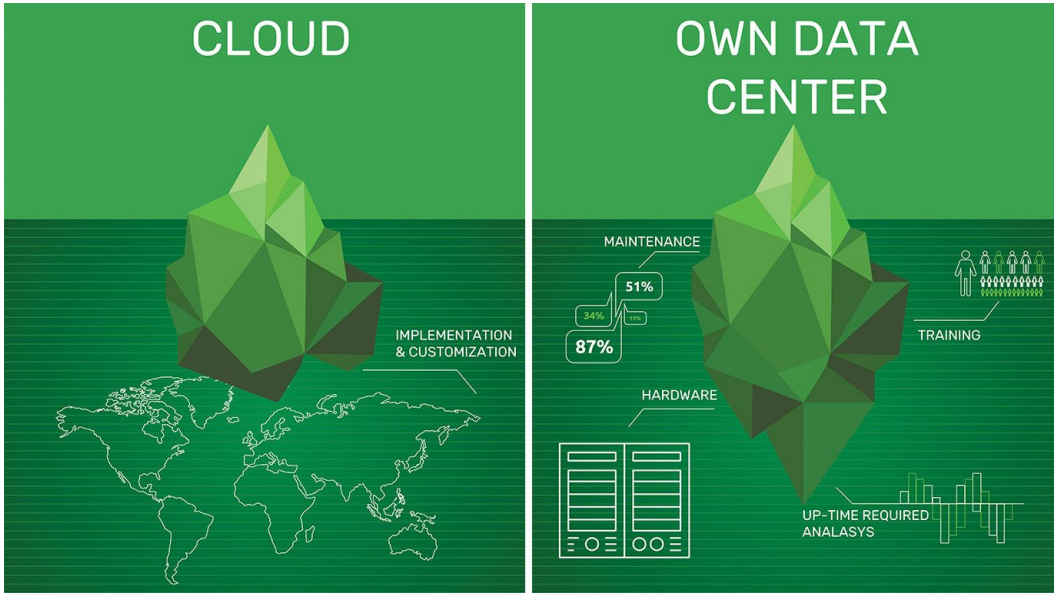
# 4. Cloud

- No tenemos que mantener nuestra propia infraestructura física.
- Ofrecen distintos productos e integraciones entre estos:
  - VMs
  - Balanceadores de carga
  - Bases de datos
  - Kubernetes
  - CICD
  - Hadoop
  - IA
  - etc
- **Vendor lock-in**



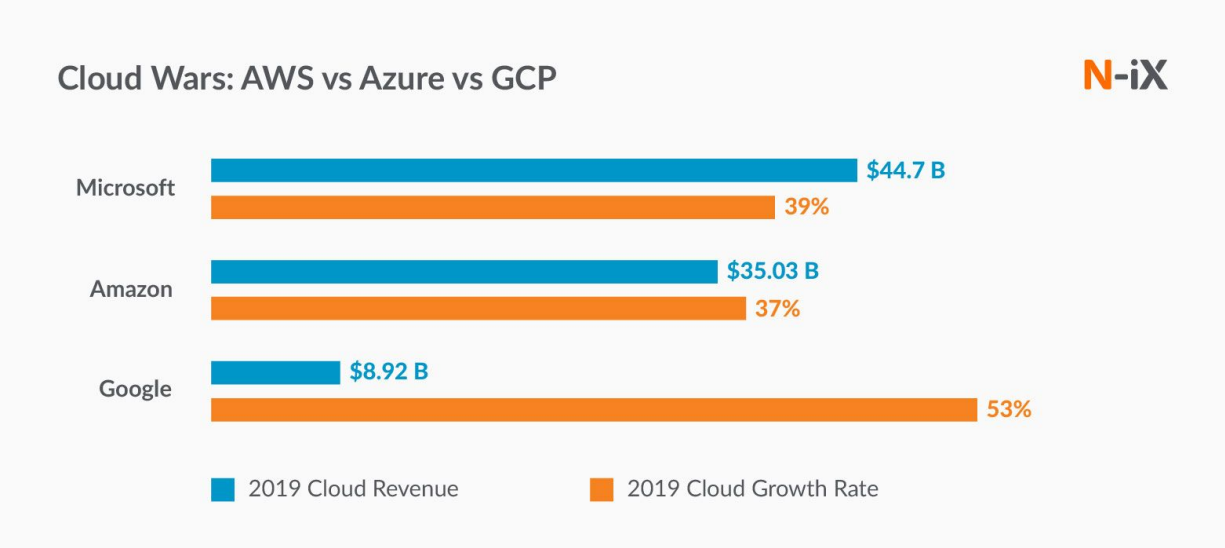
# 4. Cloud

Datacenter vs Cloud



# 4. Cloud

Ganancia:



# 4. Contenedores, Kubernetes y Cloud

**Conclusiones**:

- **Contenedores**: Son el nuevo formato para empaquetar aplicaciones (artefactos).
- **Kubernetes**: Es el estándar de despliegue automático y declarativo de los artefactos (aplicaciones)
- **Cloud**: Reducir costes y facilita el mantenimiento de los entornos. Es amigable con los desarrolladores.
- **OpenSource**: Es el catalizador de estas tecnologías y del cambio en las organizaciones.

**Problema 1: Sistemas cuello de botella**

Sistemas ya no es un cuello de botella. SREs crean las herramientas para facilitar la gestión de entornos a los desarrolladores.

**Problema 2: Brecha entre desarrolladores y sistemas**

Estas tecnologías acercan a los desarrolladores y administradores, creando perfiles híbridos que tienen que conocer ambos mundos.

# 5. IaC, CaC, GitOps

**IaaC** (Infraestructura como código): Motor de automatización renderiza código en forma de infraestructura y recursos. Aprovisiona nueva infraestructura o configura la ya implementada. (Ejemplos de motores: Terraform, Ansible).

- Versionar la infraestructura facilita los cambios y permite gestionar los entornos de forma más transparente.

**CaaC** (Configuración como código): Motor de automatización que mapea configuraciones.

- Versionar las configuraciones facilita los cambios y permite gestionar las configuraciones de forma más transparente.
- Los cambios se pueden propagar masivamente.

**GitOps**: Código de infraestructura o configuraciones versionado en Git (por ejemplo en GitLab, Bitbucket, etc.)

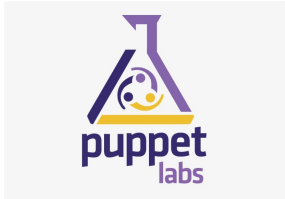
- Este concepto no solo aplica a infraestructura o configuraciones
- Versionar el código te permite mantener un histórico y retroceder a una versión anterior si es necesario.
- Ejemplo workflow:
    - 1. El equipo SRE y sistemas gestionan una implementación multitenant geodistribuida de múltiples clusters de k8s.
    - 2. El acceso a los proyectos del cluster está puesto en forma de código.
    - 3. Cuando un nuevo equipo necesita acceso a la plataforma, crea una Merge Request contra el repositorio.
    - 4. El equipo SRE revisa el código y ejecuta la Merge Request.
    - 5. Un pipeline CICD propaga los cambios contra los clusters automáticamente tras el Merge.
    - 6. El nuevo equipo de Developers quiere hacer cambios porque se ha equivocado.
    - 7. Se siguen los pasos anteriores y el pipeline propaga los cambios.

# 5. IaC, CaC, GitOps

Tecnologías destacables:



- Ansible:
  - Agentless (vía SSH).
  - Modular.
  - Basado en python (interpretado).
  - Imperativo. (Muy recomendable para CaaC, también puede usarse para IaaC).
- Terraform:
  - Agentless (provisioners)
  - Modular
  - Basado en Goland
  - Declarativo. (Muy recomendable para IaaC, muy poco recomendable para CaaC).
- Puppet:
  - Agentful (puppet agent).
  - Modular
  - Basado en Ruby
  - Declarativo
  - Muy error-prone



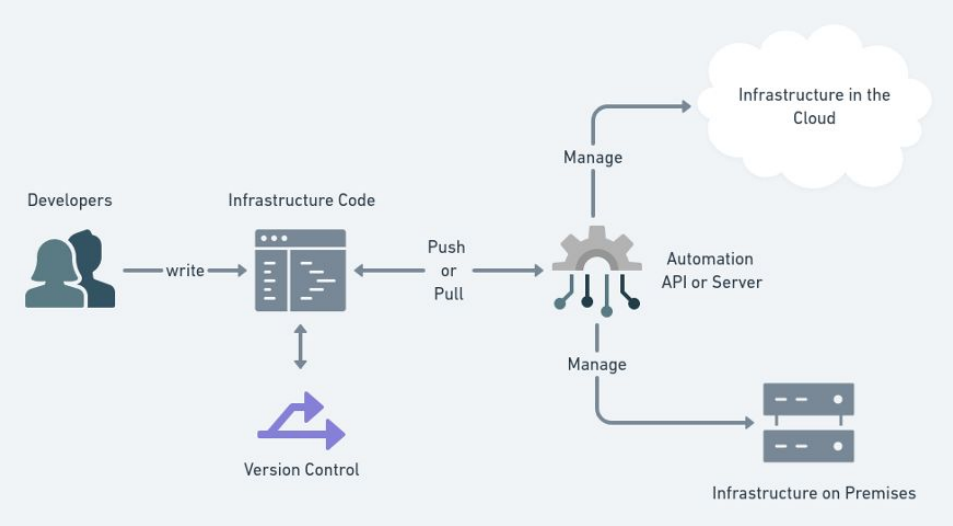
# 5. IaC, CaC, GitOps

IaaC y CaaC:

- Herramientas fundamentales para los SRE.
- Imagina gestionar más de 5k VMs, necesitas tener todo como código versionado.
- YAML *(YAML Ain't Markup Language)* es la clave.
    - Lenguaje de serialización de objetos legible
- No intervención manual en las máquinas virtuales, hipervisores, cloud-providers, etc. Todo está como código.

# 5. IaC, CaC, GitOps

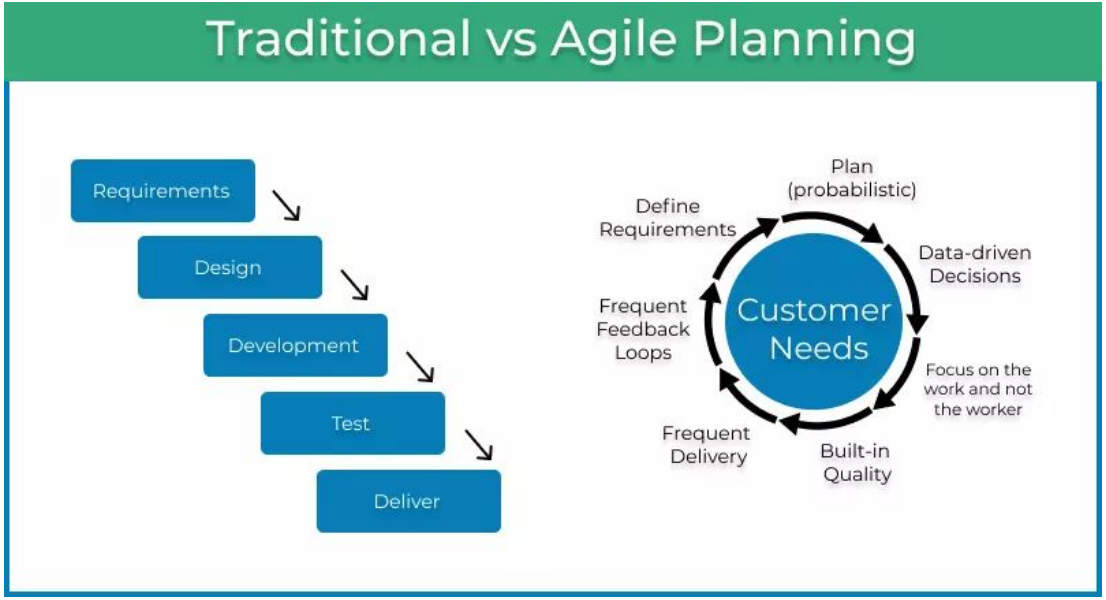
Ejemplo Workflow:



# 6. Metodologías de trabajo modernas

- Trabajo con tickets.
  - Trabajo operacional.
  - Bugs y peticiones.
- Agile (Scrum).
  - Productización (product owner, stakeholders).
  - Mayor colaboración.
  - Iteraciones cortas (sprints)

# 6. Metodologías de trabajo modernas



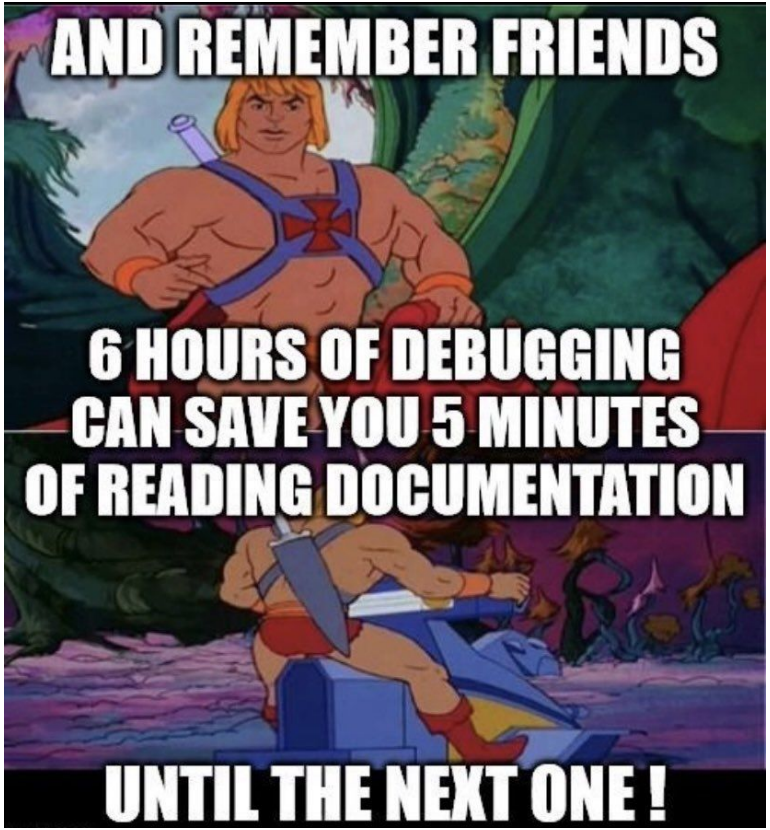
# 7. Conclusiones

- El mercado se adapta a las nuevas necesidades. DevOps y Site Reliability Engineering han revolucionado el suministro de soluciones a los dos problemas mencionados anteriormente.
- Nuevas soluciones tecnológicas nacen para resolver las necesidades.
- Las nuevas necesidades y tecnologías también plantean nuevos retos. El sector de IT se transforma a un paradigma radicalmente nuevo.
- El open-source es vital en esta transformación.
- A pesar de que todo cambia la base SIEMPRE es la misma.
- Ninguna solución es perfecta. Todo depende del caso de uso o del problema que intentamos aproximar. Hay que valorar siempre pros y contras.

https://www.linkedin.com/jobs/search/?keywords=sre

https://www.linkedin.com/jobs/search/?keywords=devops

https://www.linkedin.com/jobs/search/?keywords=sysadmin


AND REMEMBER FRIENDS
6 HOURS OF DEBUGGING
CAN SAVE YOU 5 MINUTES
OF READING DOCUMENTATION
UNTIL THE NEXT ONE !

The end...