Análisis de seguridad de arquitecturas basadas en Kubernetes

Manuel Simón Nóvoa

Máster Interuniversitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

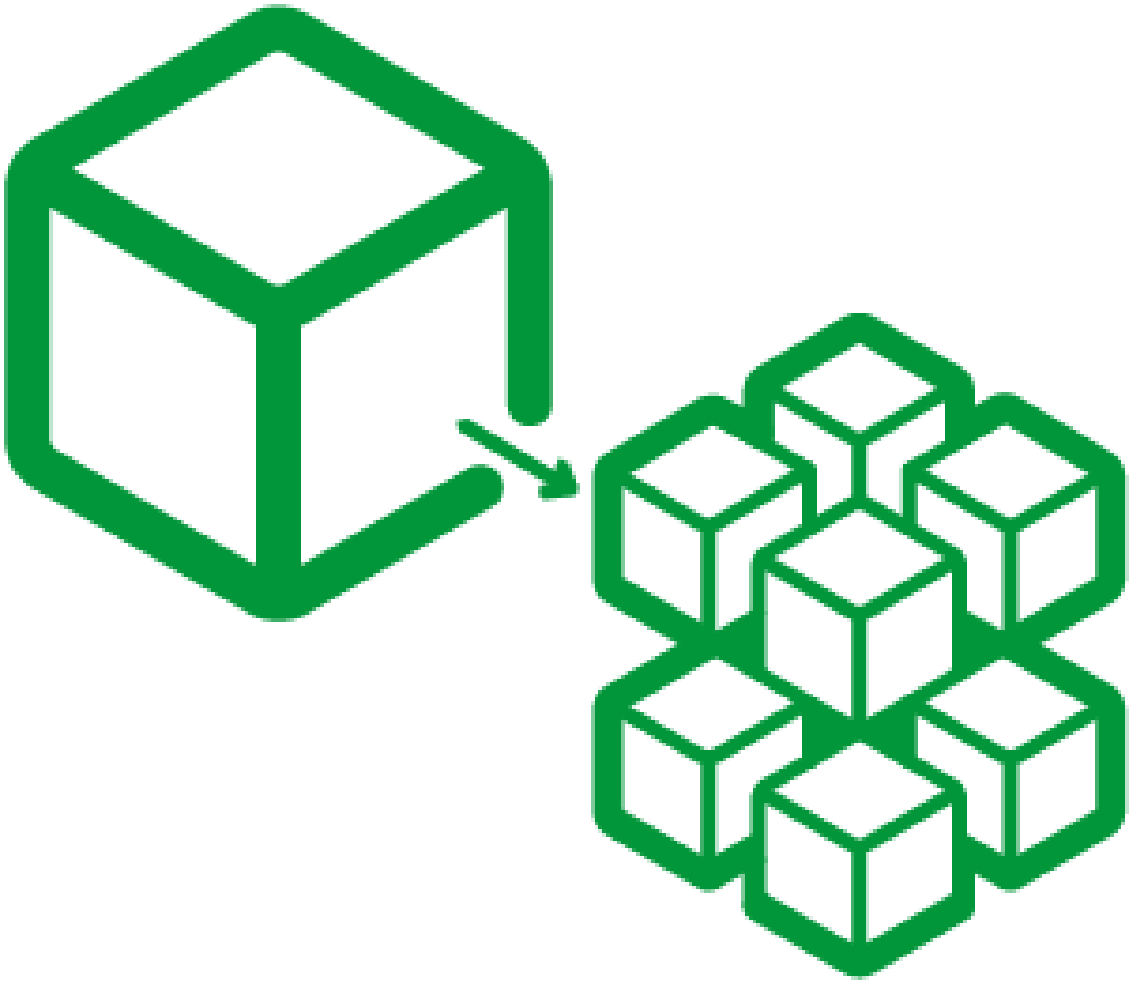
Universitat Oberta de Catalunya
Universitat Autònoma de Barcelona
Universitat Rovira i Virgili

Trabajo de Fin de Máster

Trabajo tutorizado por Juan Carlos Fernández Jara y Víctor García Font

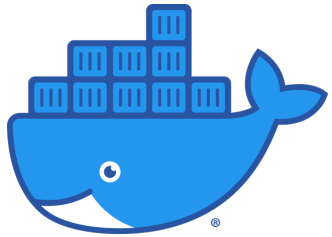
8 de junio de 2021

Contexto y justificación del trabajo: microservicios

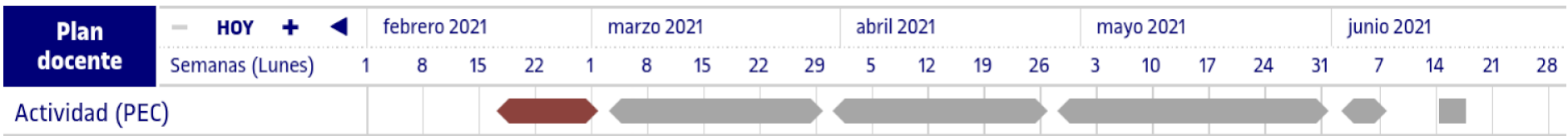


Objetivos del Trabajo

1. Planificación del trabajo
2. Estudio preliminar

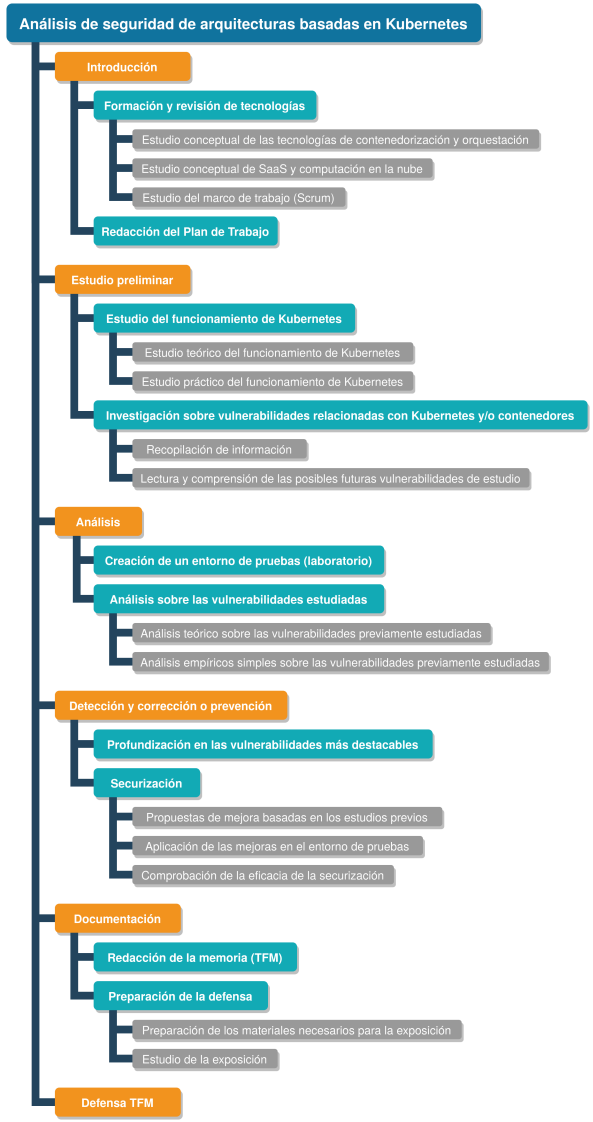


3. Creación de laboratorio de pruebas y fase de análisis de vulnerabilidades
4. Detección y corrección o prevención
5. Documentación
6. Defensa

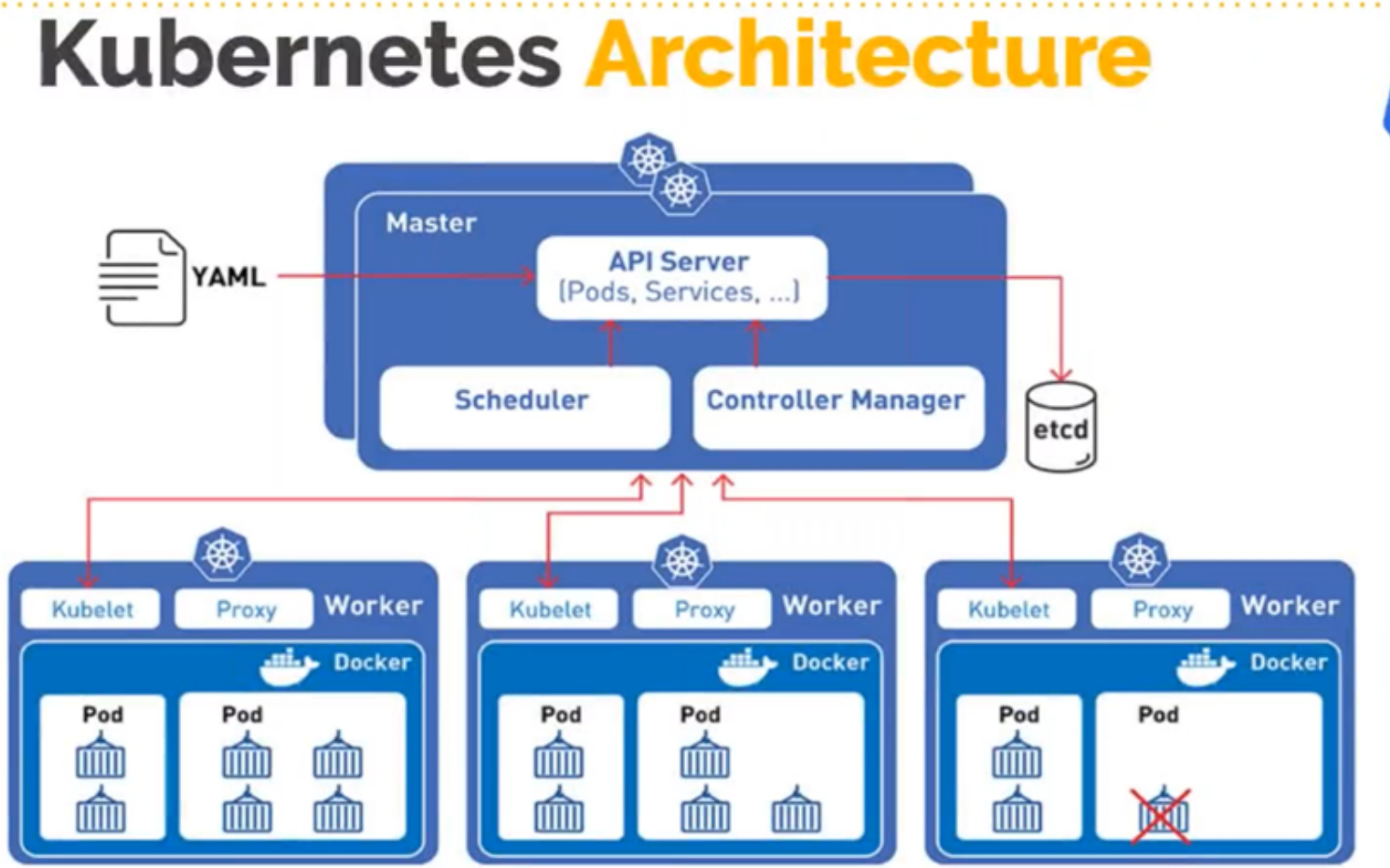


Fase de planificación

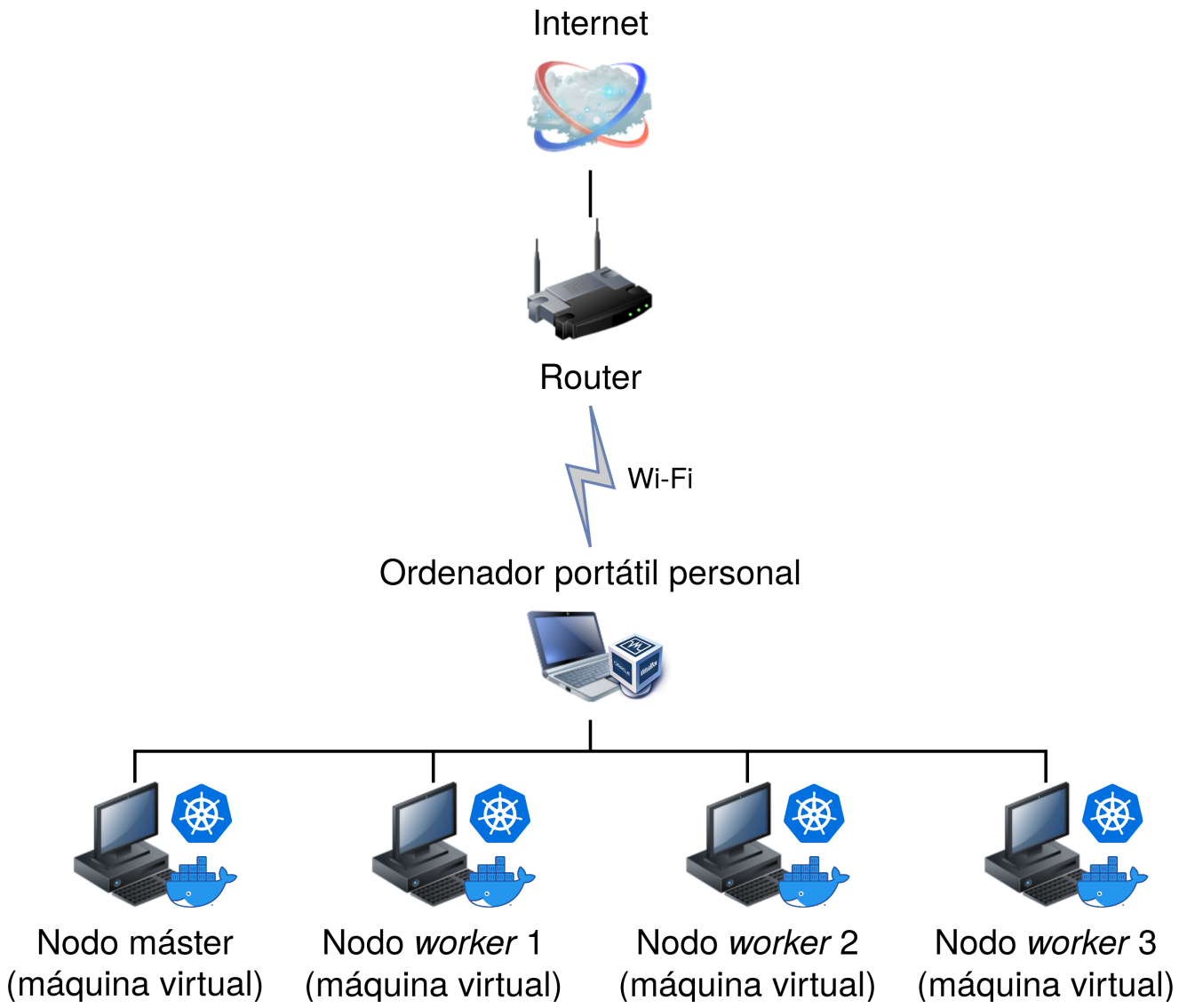
- Metodología de trabajo
- Planificación del trabajo: EDT
- Planificación temporal
- Gestión de la configuración
- Gestión de riesgos



Arquitectura Kubernetes



Laboratorio de pruebas



Plantilla estudio

Requisito detectado

Análisis de vulnerabilidades

Securización

RNF1 - Análisis estático sobre imágenes de contenedores

Requisito detectado

Comprobación de la existencia de vulnerabilidades en imágenes de contenedores descargadas desde **fuentes** bien conocidas y consideradas **"de confianza"**.

Análisis de vulnerabilidades

Utilización de la herramienta **"Docker Scan"** (Snyk) sobre imagen de confianza.

Securización

Establecido procedimiento estandarizado para detectar vulnerabilidades:

1. Selección de imagen.
2. Análisis estático con "Docker Scan".
3. Obtención de informe detallado de vulnerabilidades. El operador actúa atendiendo a los resultados obtenidos.

RNF1 - Análisis estático sobre imágenes de contenedores

Securización

Actuación en función del informe obtenido

- Continuar ejecutando imagen con vulnerabilidades, pero conociendo este hecho.
- **Ejecutar acciones de corrección** (p.e. actualizar software).
  - Finalizado el proceso de corrección, **se crea una nueva imagen** que sustituye a la original.

RNF1 - Análisis estático sobre imágenes de contenedores

```
  From: bash/bash@4.3-11+b1
  Fixed in: 4.3-11+deb8u1

✗ High severity vulnerability found in bash/bash
  Description: Improper Check for Dropped Privileges
  Info: https://snyk.io/vuln/SNYK-DEBIAN8-BASH-536279
  Introduced through: bash/bash@4.3-11+b1
  From: bash/bash@4.3-11+b1

✗ High severity vulnerability found in apt/libapt-pkg4.12
  Description: Arbitrary Code Injection
  Info: https://snyk.io/vuln/SNYK-DEBIAN8-APT-407401
  Introduced through: apt/libapt-pkg4.12@1.0.9.8.3, apt@1.0.9.8.3
  From: apt/libapt-pkg4.12@1.0.9.8.3
  From: apt@1.0.9.8.3 > apt/libapt-pkg4.12@1.0.9.8.3
  From: apt@1.0.9.8.3
  Fixed in: 1.0.9.8.5



Organization:      manuelsimon
Package manager:   deb
Project name:      docker-image|nginx
Docker image:      nginx:1.11.0
Platform:          linux/amd64
Licenses:          enabled

Tested 141 dependencies for known issues, found 436 issues.

Debian 8 is no longer supported by the Debian maintainers. Vulnerabili
```

```
✗ High severity vulnerability found in gcc-4.8/gcc-4.8-base
  Description: Information Exposure
  Info: https://snyk.io/vuln/SNYK-DEBIAN8-GCC48-347566
  Introduced through: gcc-4.8/gcc-4.8-base@4.8.4-1
  From: gcc-4.8/gcc-4.8-base@4.8.4-1

✗ High severity vulnerability found in dpkg
  Description: Directory Traversal
  Info: https://snyk.io/vuln/SNYK-DEBIAN8-DPKG-336522
  Introduced through: meta-common-packages@meta
  From: meta-common-packages@meta > dpkg@1.17.27

✗ High severity vulnerability found in bash
  Description: Improper Check for Dropped Privileges
  Info: https://snyk.io/vuln/SNYK-DEBIAN8-BASH-536279
  Introduced through: bash@4.3-11+deb8u2
  From: bash@4.3-11+deb8u2



Organization:      manuelsimon
Package manager:   deb
Project name:      docker-image|manuelsimon/nginxactualizado
Docker image:      manuelsimon/nginxactualizado:1.11.0
Platform:          linux/amd64
Licenses:          enabled

Tested 141 dependencies for known issues, found 136 issues.

Debian 8 is no longer supported by the Debian maintainers. Vulnerabi
```

RNF2 - Análisis estático sobre ficheros de conf. YAML

Requisito detectado

Comprobación de malas prácticas o configuraciones en ficheros YAMLs utilizados para los despliegues.

Análisis de vulnerabilidades

Utilización de la herramienta **Kubesec** para detectar errores de configuración.

Securización

Utilización de diversos parámetros en los ficheros YAML para obtener entornos más seguros. Realmente están **basados en mejoras dadas por otras pruebas** del proyecto: límites memoria, CPU; sin permisos privilegiados; modificación UID; etc. Pasamos de 0 puntos iniciales a un total de 7.

RNF3 - Limitación de recursos: memoria RAM

Requisito detectado

Comprobación de la posibilidad de hacer un **uso excesivo de memoria**, provocando un mal funcionamiento en otras aplicaciones, dentro o fuera del contenedor que ejecute la aplicación.

Análisis de vulnerabilidades

Uso de un programa que consume memoria en bucle infinito.

- Protección parcial dada por error OOM.
- Se consiguió que uno de los nodos ***worker*** **dejara de dar servicio** durante unos instantes.

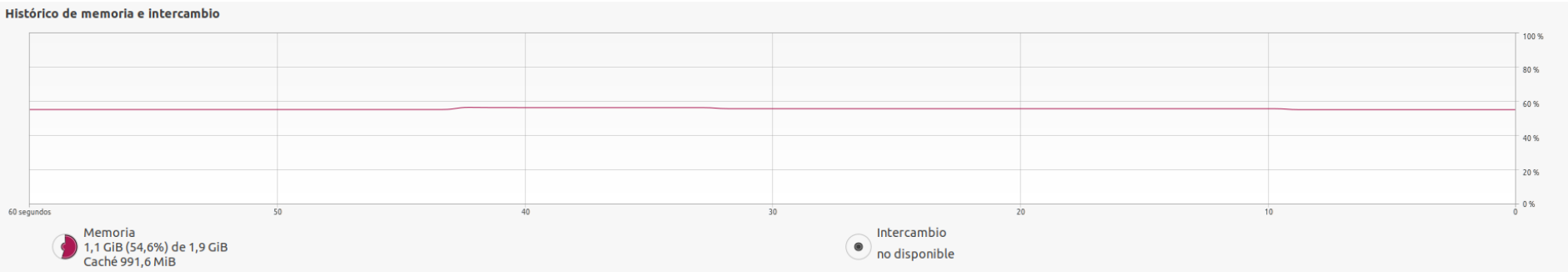
Securización

Uso de **configuración *resources*: *requests* y *limits*** en el YAML.

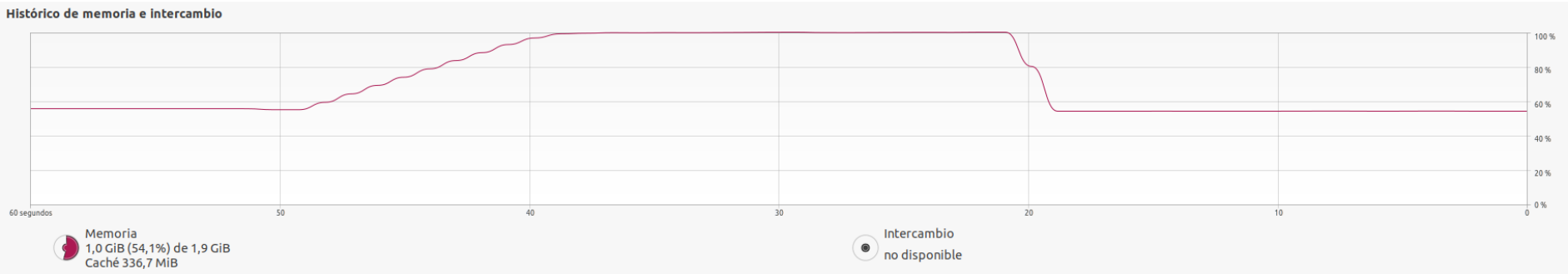
- El nodo ***worker*** **nunca deja de dar servicio**.

RNF3 - Limitación de recursos: memoria RAM

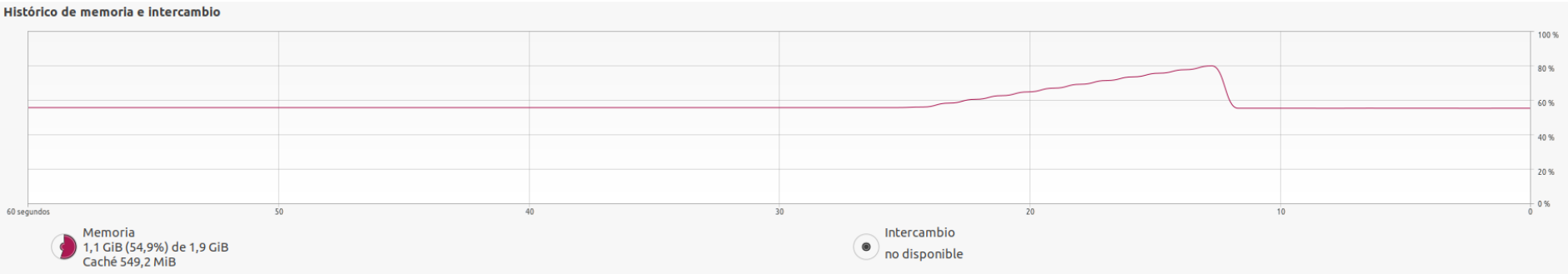
Estado de la memoria antes de la prueba:



Estado de la memoria en la fase de análisis:



Estado de la memoria en la fase de securización:



RNF4 - Limitación de recursos: CPU

Requisito detectado

Comprobación de la posibilidad de hacer un **uso excesivo de CPU**, provocando un mal funcionamiento en otras aplicaciones, dentro o fuera del contenedor que ejecute la aplicación.

Análisis de vulnerabilidades

Uso de un programa que consume CPU en bucle infinito.

- Se consiguió que uno de los nodos ***worker*** **dejara de dar servicio** durante unos instantes.

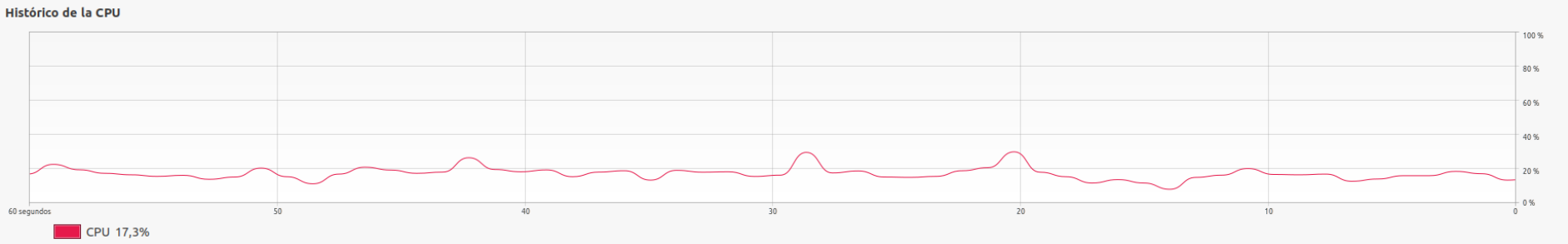
Securización

Uso de **configuración *resources*: *requests* y *limits*** en el YAML.

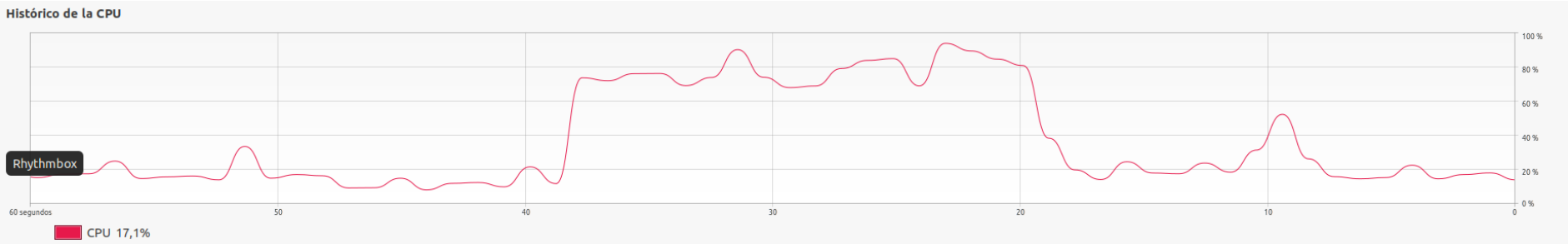
- El nodo ***worker*** **nunca deja de dar servicio**.

RNF4 - Limitación de recursos: CPU

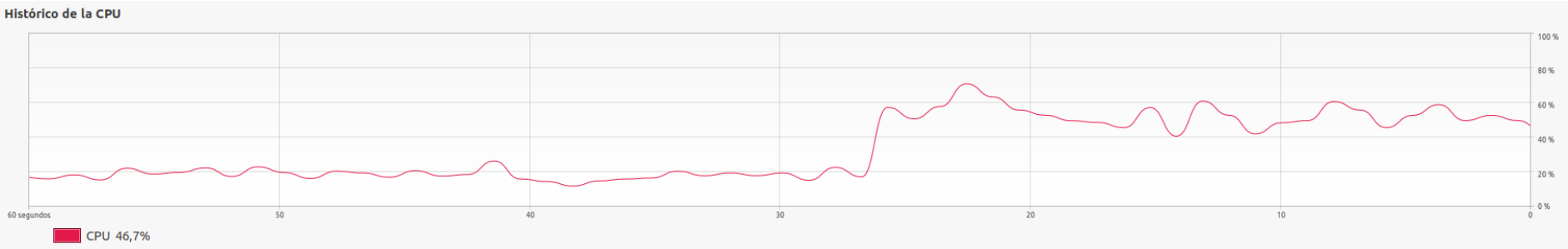
Estado de la CPU antes de la prueba:



Estado de la CPU en la fase de análisis:



Estado de la CPU en la fase de securización:



RNF5 - Limitación de recursos: red

Requisito detectado

Comprobación de la posibilidad de hacer un **uso excesivo de red**, provocando un mal funcionamiento en otras aplicaciones, dentro o fuera del contenedor que ejecute la aplicación.

Análisis de vulnerabilidades

- En este caso, no fue usado ningún software específico: *curl*
- **Uso del máximo ancho de banda disponible** desde un único contenedor.

Securización

⚠⚠Este ha sido el **único requisito que no fue finalmente abordado** en la fase de securización. Indicado para la fase de **trabajo futuro**:

- Aplicación de políticas de QoS.
- Delegación de esta limitación en el *Service Mesh* de Istio.

RNF6 - Limitación permisos ejecución dentro contenedores

Requisito detectado

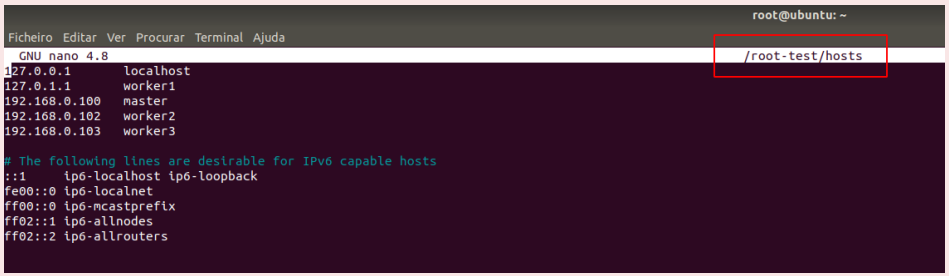
Comprobación de los **permisos de ejecución dentro de un contenedor** por defecto. En caso de ser superusuario, limitar su uso.

Análisis de vulnerabilidades

- **Implicaciones** a nivel de seguridad dentro del **contenedor**.

```
manuel@master:~$ kubectl apply -f ubuntu.yaml
pod/ubuntu created
manuel@master:~$ kubectl exec --stdin --tty ubuntu -- /bin/bash
root@ubuntu:/# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/#
```

- **Implicaciones** a nivel de seguridad en la **máquina anfitriona**.




```
Last login: Sun May  2 12:06:01 2021 from 192.168.1.43
manuel@worker1:~$ ping ataque
PING ataque (8.8.8.8) 56(84) bytes of data.
64 bytes from ataque (8.8.8.8): icmp_seq=1 ttl=116 time=14.1 ms
64 bytes from ataque (8.8.8.8): icmp_seq=2 ttl=116 time=14.3 ms
64 bytes from ataque (8.8.8.8): icmp_seq=3 ttl=116 time=13.8 ms
64 bytes from ataque (8.8.8.8): icmp_seq=4 ttl=116 time=14.2 ms
64 bytes from ataque (8.8.8.8): icmp_seq=5 ttl=116 time=14.5 ms
64 bytes from ataque (8.8.8.8): icmp_seq=6 ttl=116 time=14.3 ms
^C
--- ataque ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5004ms
rtt min/avg/max/mdev = 13.779/14.188/14.515/0.218 ms
manuel@worker1:~$
```

RNF6 - Limitación permisos ejecución dentro contenedores

Securización

La securización fue realizada empleando las "***pod security policies***".

- Mejoras a **nivel** de seguridad dentro del **contenedor**:
  - Obtención de nuevo UID != 0 → **sin permisos de *root***
  - **Imposibilidad de escalada** de privilegios

```
I have no name!@ubuntu-deployment-688c7667d9-ftxdb:/$ id
uid=1000 gid=3000 groups=3000
```

```
I have no name!@ubuntu-deployment-688c7667d9-ftxdb:/$ adduser
adduser: Only root may add a user or group to the system.
```

- Mejoras a **nivel** de seguridad en la **máquina anfitriona**:
  - Uso de ***namespaces*** **para mapear los UID y GID** fuera del contenedor.
    - Un usuario ***root*** **dentro** de un contenedor **no** es interpretado como tal **fuera** de él.

```
root@93f2f12210d1:~# echo "ataque    8.8.8.8" > /etchost/hosts
bash: /etchost/hosts: Permission denied
root@93f2f12210d1:~# id
uid=0(root) gid=0(root) groups=0(root)
root@93f2f12210d1:~#
```

RNF7 - Comparativa asialamiento MVs VS Contenedores

Requisito detectado

Si bien la **virtualización a nivel de SO** ofrece una metodología y despliegues más **ágiles**, presentan un **aislamiento** que podría ser considerado **"menor"**, en comparación con la virtualización de hardware. Comprobar esta diferencia.

Análisis de vulnerabilidades

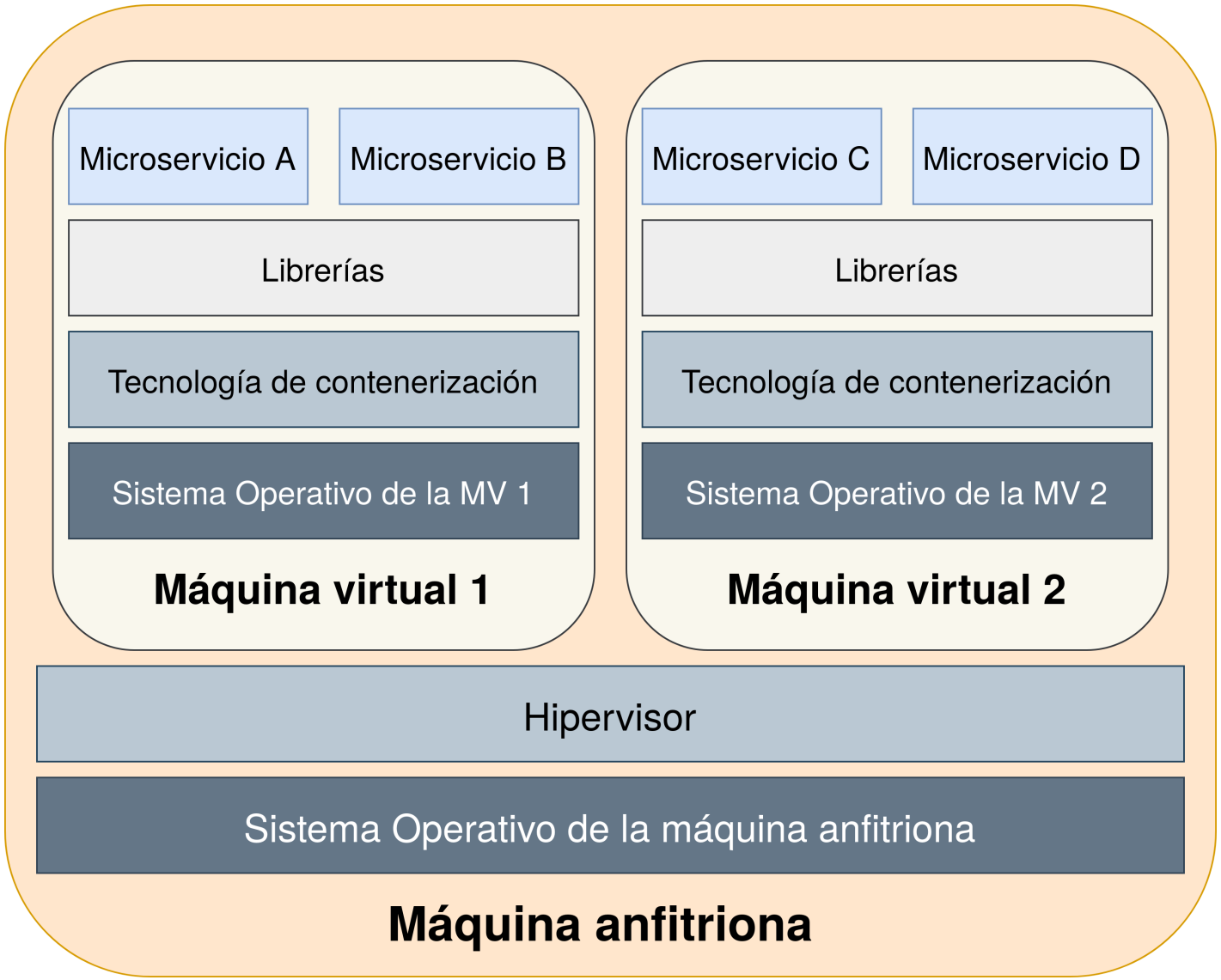
Se adaptó el laboratorio para incluir la **máquina anfitriona como nodo *worker***:

- ***Repetición de la prueba*** *de consumo excesivo de* ***memoria*** *RAM.*
  - **Fueron provocados problemas de DoS** que llevaron al **cierre inesperado** de una de las máquinas virtuales **(error grave)**.

Securización

Realmente, el laboratorio original ya está conformado por un modelo mixto, que consigue integrar los pros de ambas virtualizaciones. **Uso conjunto de MVs y contenedores**.

RNF7 - Comparativa asialamiento MVs VS Contenedores



RNF8 - Estudio de sistemas de solo lectura

Requisito detectado

En ciertas ocasiones, desplegaremos **aplicaciones** contenerizadas que **no precisen escribir en disco**. En este tipo de situación, permitir escrituras en disco puede constituir una vulnerabilidad innecesaria.

Análisis de vulnerabilidades

Escritura en disco habilitada y **permitida por defecto**.

```
root@nginx-deployment-84f4dd795b-62f9j:/# ls
bin  boot  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@nginx-deployment-84f4dd795b-62f9j:/# touch test
root@nginx-deployment-84f4dd795b-62f9j:/# ls
bin  boot  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  test  tmp  usr  var
root@nginx-deployment-84f4dd795b-62f9j:/#
```

Securización

Utilización de las "***pod security policies***" **para impedir la escritura en disco**. (*securityContext: readOnlyRootFileSystem*)

```
root@ubuntu-deployment-b67857cb9-wskws:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@ubuntu-deployment-b67857cb9-wskws:/# touch test
touch: cannot touch 'test': Read-only file system
root@ubuntu-deployment-b67857cb9-wskws:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

RNF9 - Validación de imágenes mediante firma digital

Requisito detectado

Poder asegurar las propiedades de autenticación, integridad y no repudio mediante el **uso de firmas digitales sobre imágenes**.

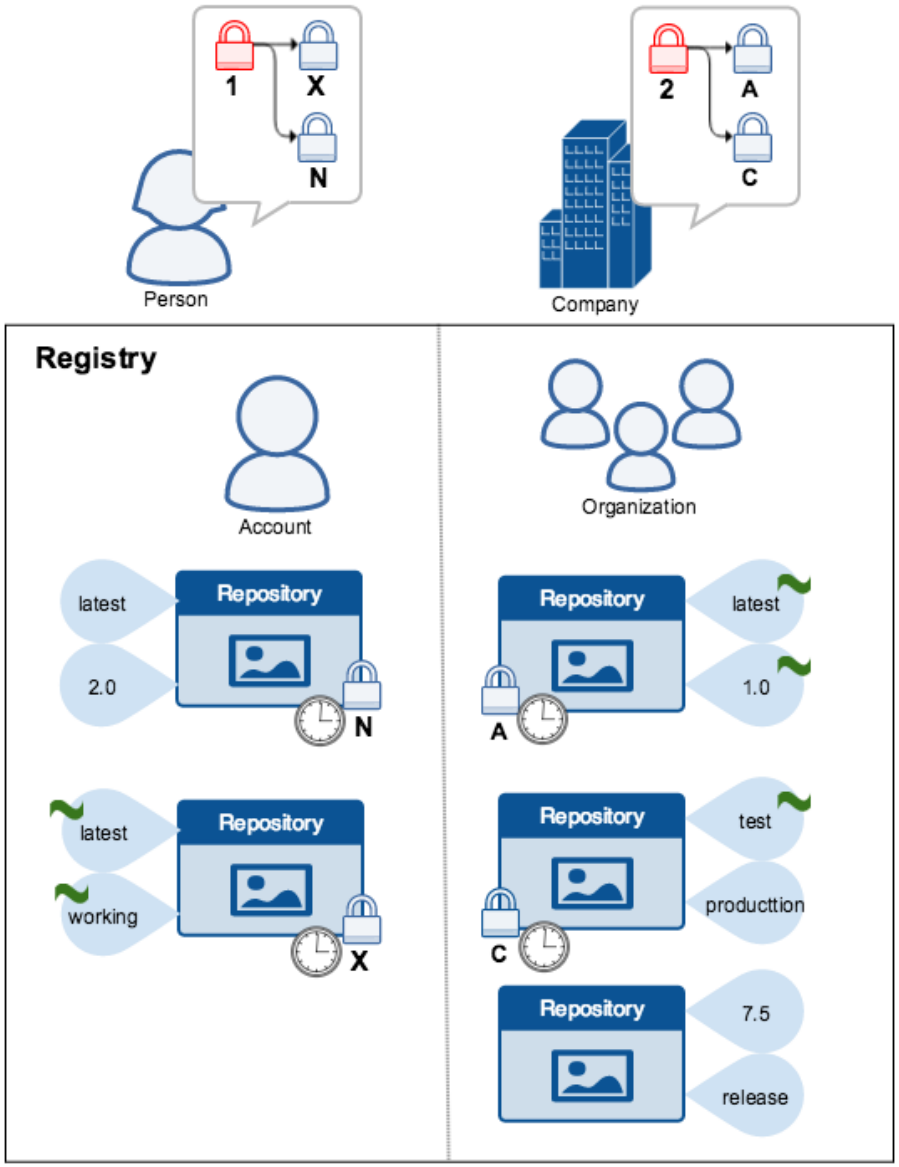
Análisis de vulnerabilidades

Posibilidad de descarga y despliegue de imágenes sin ningún tipo de mecanismo de verificación. Docker soporta valicación mediante **"Docker Content Trust", pero Kubernetes ¡no soporta esta herramienta por defecto!**

Securización

Integración de "Docker Content Trust" con Kubernetes a través de un software externo: **Connaiseur**.

RNF9 - Validación de imágenes mediante firma digital



Offline key

A offline key is used to create tagging keys. Offline keys belong to a person or an organization. Resides client-side. You should store these in a safe place and back them up.

Tagging key

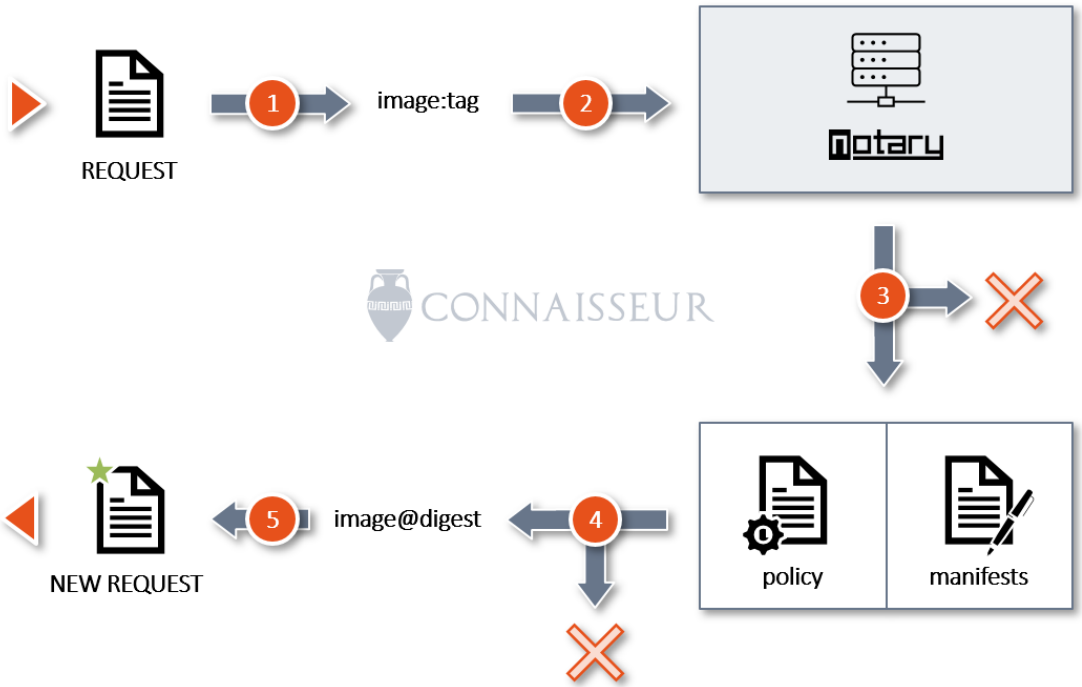
A tagging key is associated with an image repository. Creators with this key can push or pull any tag in this repository. This resides on client-side.

Timestamp Key

A timestamp key is associated with an image repository. This is created by Docker and resides on the server.

Signed tag.

RNF9 - Validación de imágenes mediante firma digital




```
root@master:~# kubectl run unsigned --image=manuelsimon/pruebafirmas:unsigned
Error from server: admission webhook "connaisseur-svc.connaisseur.svc" denied the request: could not find signed digest for image "docker.io/manuelsimon/pruebafirmas:unsigned" in trust data.
```

```
root@master:~# kubectl run signed --image=manuelsimon/pruebafirmas:signed
pod/signed created
```

RNF10 - Herencia de vulnerabilidades entre imágenes

Requisito detectado

Comprobación de la posible herencia de vulnerabilidades entre imágenes dependientes.

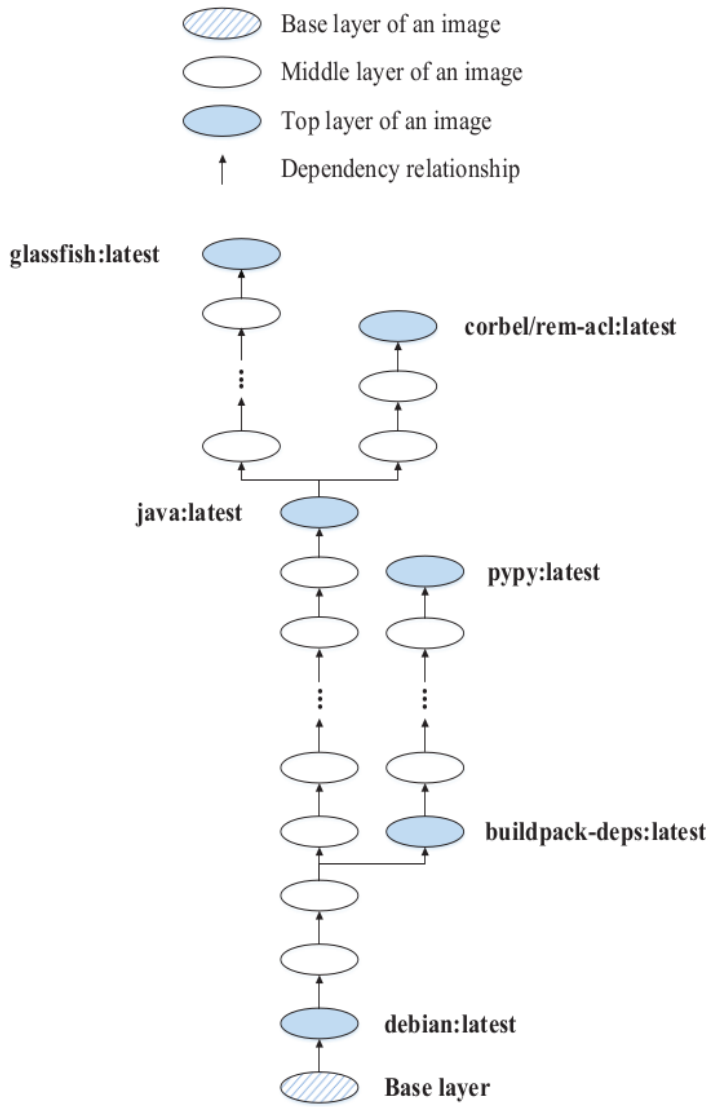
Análisis de vulnerabilidades

Las imágenes de contenedores no tienen por qué ser creadas desde cero, sino que pueden tomar otras como base, formando una **relación padre-hijo**. Debido a esta propiedad, una de las características que **pueden heredar** son las **vulnerabilidades** conocidas entre imágenes. → Estudio "*A Study of Security Vulnerabilities on Docker Hub*" (Rui Shu, Xiaohui Gu y William Enck).

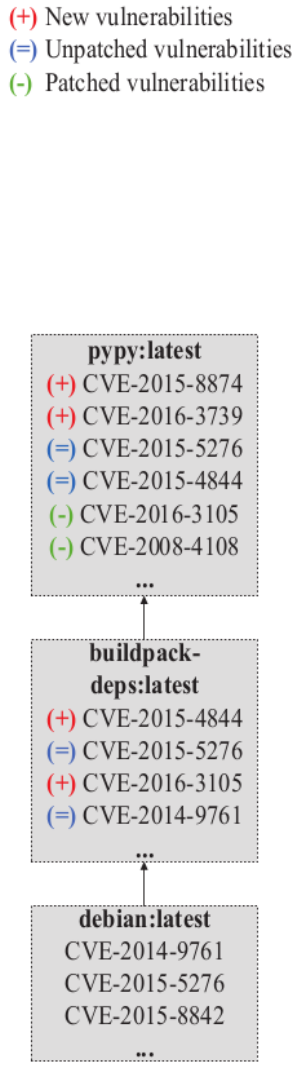
Securización

Ejecutar el **mismo procedimiento** que fue establecido para el **análisis estático sobre imágenes** de contenedores.

RNF10 - Herencia de vulnerabilidades entre imágenes



(a) Image dependency graph example



(b) Vulnerability propagation results

RNF11 - Diferenciación de espacios para diferentes usos

Requisito detectado

Separación en diferentes espacios para los múltiples equipos y/o aplicaciones, con el fin de **evitar conflictos** entre ellos.

Análisis de vulnerabilidades

En una configuración **"por defecto"**, todos los contenedores corren bajo el **mismo espacio de nombres *default***.

Securización

Creación de nuevos espacios de nombre, asociados a diferentes contextos, como desarrollo o producción. Despliegue de **nuevos contenedores** en estos nuevos espacios y comprobación de que estos **no son visibles desde un espacio ajeno**.

RNF11 - Diferenciación de espacios para diferentes usos

```
manuel@master:~$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                    READY
default       kubernetes-bootcamp-57978f5f5d-5d8z8    1/1
default       nginx-deployment-84f4dd795b-78l9p       1/1
default       nginx-deployment-84f4dd795b-7th5l       1/1
default       ubuntu-deployment-7ffd97df99-2z9ww      1/1
default       ubuntu-deployment-7ffd97df99-j88wp      1/1
default       ubuntu-deployment-7ffd97df99-rtpct      1/1
default       ubuntu-deployment-7ffd97df99-vtbxn      1/1
kube-system   coredns-558bd4d5db-lfmtc                1/1
kube-system   coredns-558bd4d5db-n742w                1/1
kube-system   etcd-master                             1/1
kube-system   kube-apiserver-master                   1/1
kube-system   kube-controller-manager-master          1/1
```

```
manuel@master:~$ kubectl create -f crear-namespace2.json
error: the path "crear-namespace2.json" does not exist
manuel@master:~$ nano crear-namespace2.json
manuel@master:~$ kubectl create -f crear-namespace2.json
namespace/espacio2 created
manuel@master:~$ kubectl get namespaces --show-labels
NAME              STATUS   AGE   LABELS
default           Active   57m   kubernetes.io/metadata.name=default
espacio1          Active   85s   kubernetes.io/metadata.name=espacio1,name=espacio1
espacio2          Active   12s   kubernetes.io/metadata.name=espacio2,name=espacio2
kube-node-lease   Active   57m   kubernetes.io/metadata.name=kube-node-lease
kube-public       Active   57m   kubernetes.io/metadata.name=kube-public
kube-system       Active   57m   kubernetes.io/metadata.name=kube-system
```

```
manuel@master:~$ kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
ubuntu-deployment   3/4     4            3           50s
manuel@master:~$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
ubuntu-deployment-688c7667d9-cflf8   1/1     Running   0          64s
ubuntu-deployment-688c7667d9-j4bs6   1/1     Running   0          64s
ubuntu-deployment-688c7667d9-rqtxh   0/1     Pending   0          64s
ubuntu-deployment-688c7667d9-wrfxz   1/1     Running   0          64s
manuel@master:~$ kubectl config use-context prod
Switched to context "prod".
manuel@master:~$ kubectl get pods
No resources found in espacio2 namespace.
```

Requisito detectado

Comprobar si existe la posibilidad de establecer **comunicación** entre diferentes **contenedores desplegados bajo diferentes nodos** del clúster. **Evitar una interconexión total**.

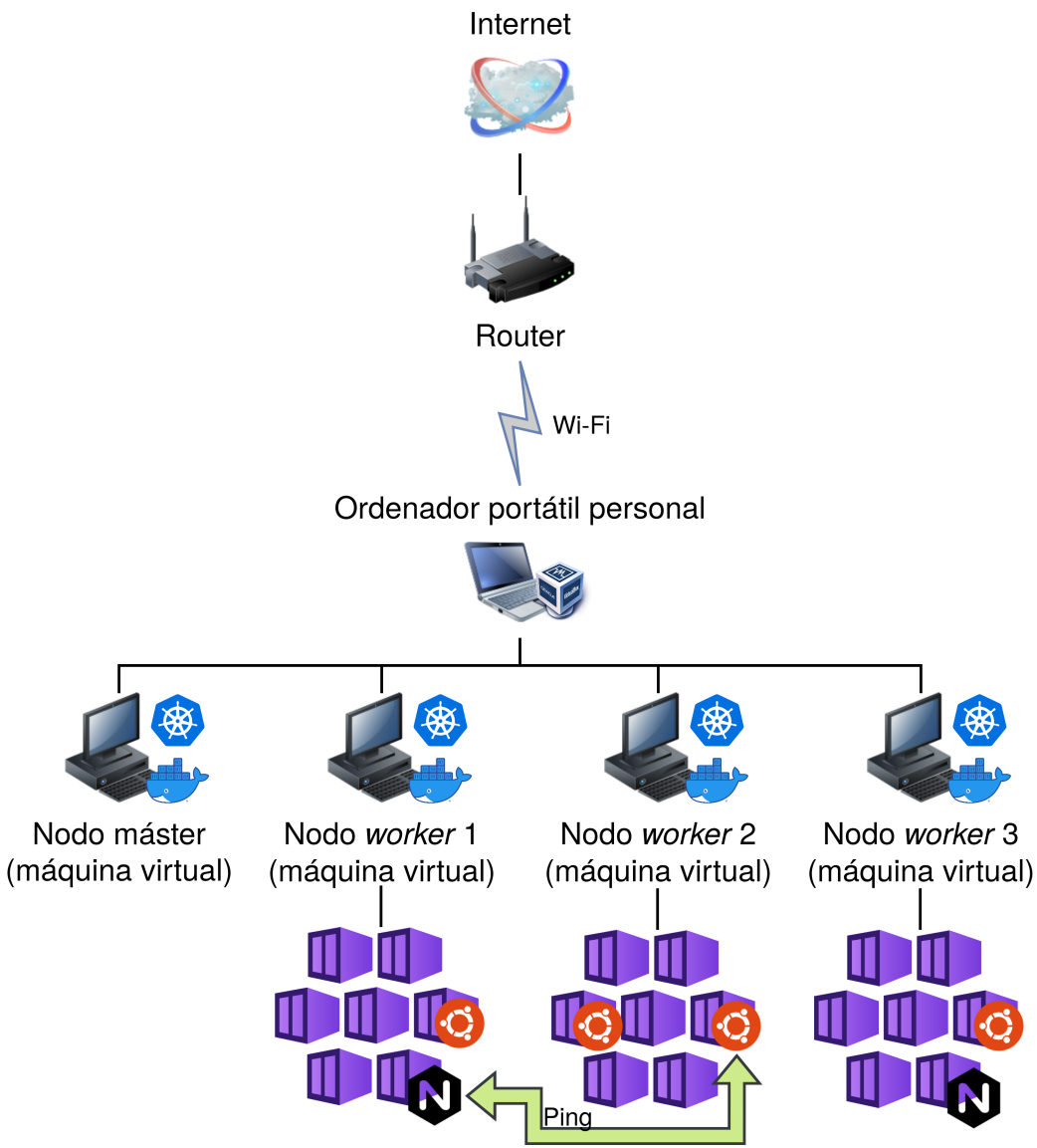
Análisis de vulnerabilidades

Despliegue de dos aplicaciones no dependientes en nodos diferentes y **comprobación de la existencia de intercomunicación** entre ellas.

Securización

Service Mesh

RNF12 - Limitación de la comunicación de contenedores en diferentes nodos



RNF13 - Configuración y uso del componente *Secret*

Requisito detectado

El componente ***Secret*** está pensado para **almacenar información** considerada privada de **manera cifrada**. Configurar el uso de dicho componente.

Análisis de vulnerabilidades

En caso de **no usar dicho componente, la información privada será indicada sin cifrar** en todas partes, incluyendo ficheros de configuración, que pueden ser públicos.

Securización

Una vez configurado el uso de este componente, es posible hacer **referencia a datos privados** desde los ficheros **YAML sin almacenarlos como texto plano**. En el **interior de los contenedores** desplegados, la **información es accesible** fácilmente y sin cifrar.

RNF13 - Configuración y uso del componente *Secret*

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-manuel
type: Opaque
data:
  USER_NAME: dXNlcg==
  PASSWORD: cGFzcw==
```

```
env:
- name: envuser
  valueFrom:
    secretKeyRef:
      name: secret-manuel
      key: USER_NAME
- name: envpass
  valueFrom:
    secretKeyRef:
      name: secret-manuel
      key: PASSWORD
```

```
manuel@master:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
ubuntu-deployment-799449f7c5-92swg   1/1     Running   0          74s
ubuntu-deployment-799449f7c5-b7rm7   1/1     Running   0          74s
manuel@master:~$ kubectl exec --stdin --tty ubuntu-deployment-799449f7c5-92swg -- /bin/bash
root@ubuntu-deployment-799449f7c5-92swg:/# echo $envuser
user
root@ubuntu-deployment-799449f7c5-92swg:/# echo $envpass
pass
```

RNF14 - Establecimiento de conexiones de red seguras

Requisito detectado

Uso de **interconexiones de red securizadas** entre los diferentes componentes: **habilitar comunicaciones con TLS** supondría la capacidad de prevenir el esnifado de tráfico indebido o la posibilidad de verificar la identidad de cada componente.

Análisis de vulnerabilidades

La **configuración por defecto** de Kubernetes **no incorpora una interconexión de red segura** entre los diferentes componentes de un clúster.

Securización

Service Mesh

RNF15 - Monitorización del clúster

Requisito detectado

El hecho de **monitorizar constantemente** los componentes que conforman un clúster puede ser de gran **ayuda para la detección de ataques** o comportamientos extraños provocados por software malintencionado.

Análisis de vulnerabilidades

Incapacidad de detección a más "alto nivel" del mal funcionamiento de un clúster o con una revisión automática.

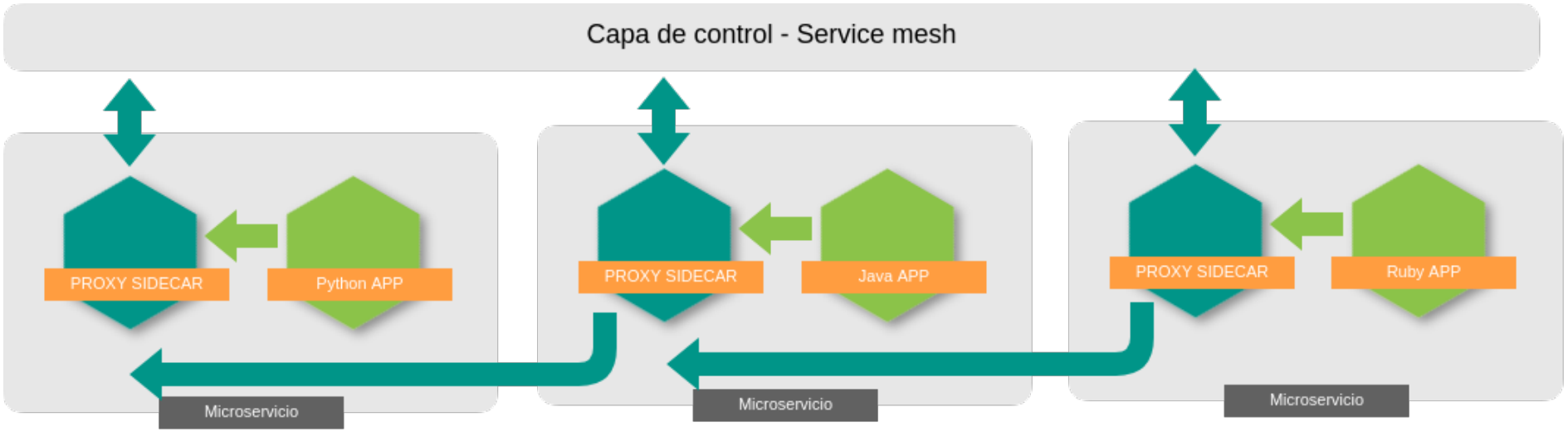
Securización

Service Mesh

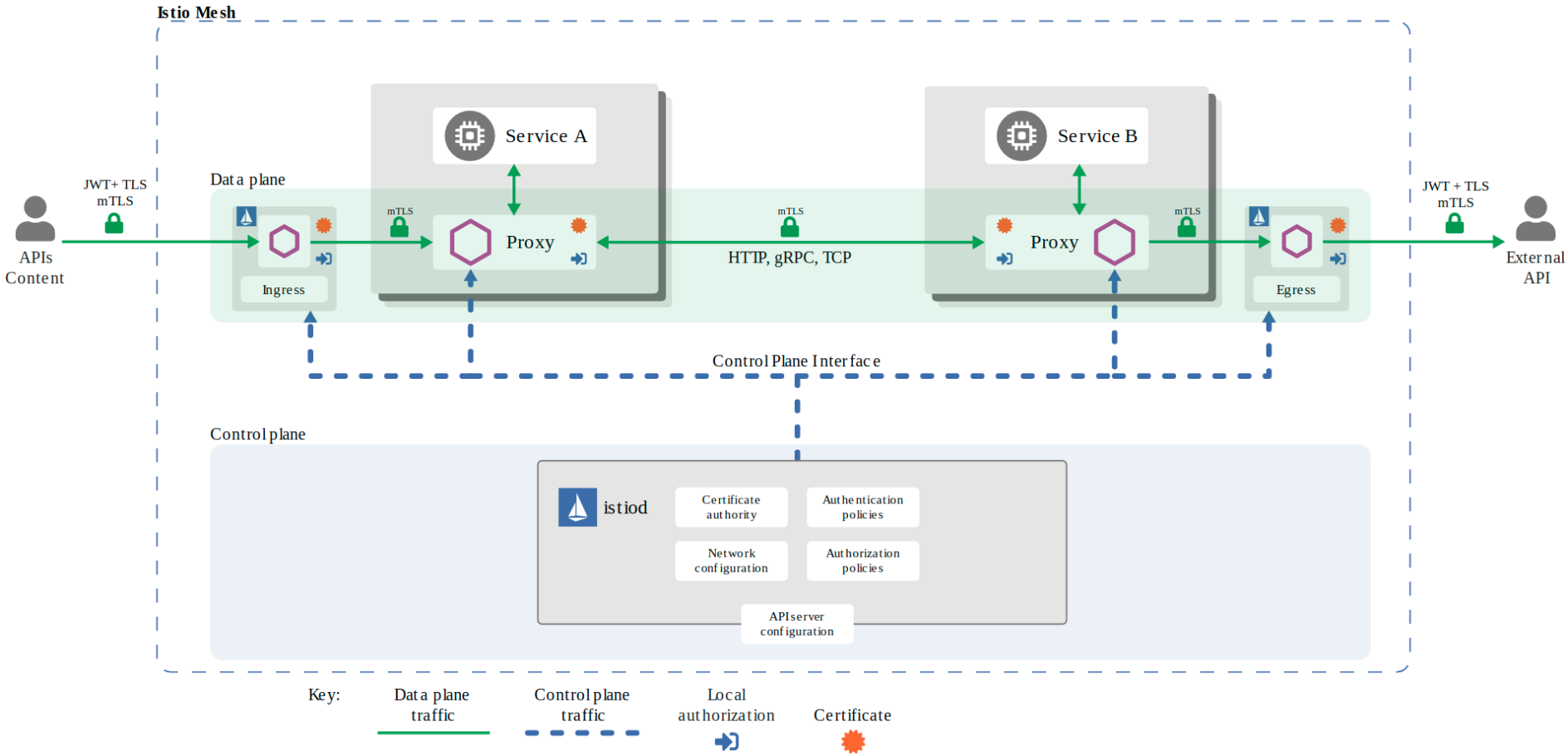
Service Mesh

- *Service Mesh* se trata de una solución especialmente diseñada para la **gestión de las comunicaciones** entre los diferentes microservicios desplegados en un clúster.
- **Separación** de toda **lógica comunicaciones** e incorporarla en las denominadas aplicaciones sidecar:
  - Lógica de negocio (*business logic*) en cada contenedor.
  - ***Endpoint*** estandarizado.
  - **Securización de la comunicación** entre los componentes (**TLS**), mediante el uso de **certificados**.
  - Lógica de **retry**.
  - Obtención de **métricas** y trazabilidad.

Service Mesh

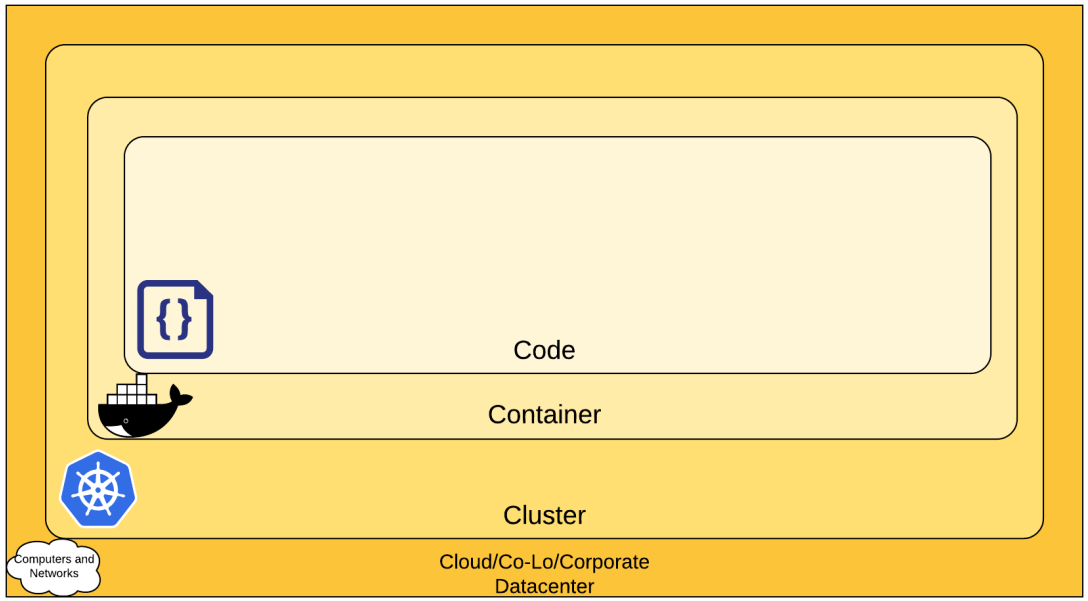


Implementación de *Service Mesh* de Istio



Conclusiones

- Estudio integral de securización de un clúster Kubernetes:
  - Proceso de planificación
  - Estudio teórico
  - Especificación de limitaciones y requisitos
  - Procedimiento de creación de un laboratorio de pruebas
  - Fase de análisis teórico-práctica
  - Fase de securización teórico-práctica
  - Trabajo futuro



Trabajo futuro

- Limitación de red
- Securización del componente *etcd*
- Securización de la red externa
- Control de Acceso Basado en Roles
- Mejora continua de los ficheros de configuración YAML
- Modelo centralizado: MAC
- Otras referencias de consulta

Análisis de seguridad de arquitecturas basadas en Kubernetes

Manuel Simón Nóvoa

Máster Interuniversitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

Universitat Oberta de Catalunya
Universitat Autònoma de Barcelona
Universitat Rovira i Virgili

Trabajo de Fin de Máster

Trabajo tutorizado por Juan Carlos Fernández Jara y Víctor García Font

8 de junio de 2021