# Autoescalado horizontal en Kubernetes con Aprendizaje Reforzado

Contenido

[Autoescalado horizontal en Kubernetes con Aprendizaje Reforzado 1](#_Toc77013405)

[Introducción 2](#_Toc77013406)

[Entorno de trabajo 3](#_Toc77013407)

[¿Qué es Kubernetes? 3](#_Toc77013408)

[Pod 4](#_Toc77013409)

[Deployment 4](#_Toc77013410)

[Replica Set 4](#_Toc77013411)

[Service 5](#_Toc77013412)

[Namespace 5](#_Toc77013413)

[Horizontal Pod Autoscaling 5](#_Toc77013414)

[Monitorización 6](#_Toc77013415)

[Estado 7](#_Toc77013416)

[Acción 7](#_Toc77013417)

[¿Qué es el Aprendizaje Reforzado? 9](#_Toc77013418)

[Markov decision Processes 10](#_Toc77013419)

[Montecarlo 10](#_Toc77013420)

[Temporal Difference 10](#_Toc77013421)

[SARSA 11](#_Toc77013422)

[Q-Learning 11](#_Toc77013423)

[Deep Q-Learning 11](#_Toc77013424)

[Implementación 12](#_Toc77013425)

[Resultados 13](#_Toc77013426)

[Conclusiones 14](#_Toc77013427)

[Bibliografía 15](#_Toc77013428)

[Repositorio de Código 15](#_Toc77013429)

[Otros Recursos 16](#_Toc77013430)

# Introducción

Con el advenimiento de los servicios en Cloud proporcionados por las grandes compañías tecnológicas (AWS, Azure, GCP) cada vez son más las empresas o particulares que deciden utilizarlos para desplegar sus aplicaciones. A medida que se comienzan a utilizar de forma más generalizada e intensiva estas infraestructuras surge inevitablemente la preocupación por hacer el uso más racional de las mismas dado los costes que suponen. Estos se basan normalmente en la cantidad de recursos (número de máquinas y tipo de las mismas) instanciadas. La gestión manual del escalado horizontal en estas plataformas es siempre susceptible de ser mejorada por sistemas automáticos que optimicen el rendimiento de las aplicaciones y el coste. De hecho las plataformas disponen normalmente de servicios que permiten a los clientes configurar el autoescalado de su infraestructura mediante sistemas de reglas basadas en umbrales para distintos parámetros. Sin embargo estos sistemas pueden ser complicados de configurar correctamente y presentar limitaciones a la hora de poder optimizar los mencionados umbrales. En el presente trabajo exploramos la posibilidad de aplicar algunos de los algoritmos existentes de Aprendizaje Reforzado al problema del auto-escalado horizontal para comprobar cómo se comportan.

En nuestro caso no lo haremos directamente sobre las plataformas Cloud mencionadas anteriormente, debido a los costes en los que podríamos incurrir, sino sobre un pequeño laboratorio desplegado en un equipo portátil consistente en un cluster de Kubernetes en el que estableceremos una comparación de rendimiento entre el sistema nativo de autescalado, denominado HPA (Horizontal Pod Autoscaler) y una solución que aplique diferentes algoritmos de aprendizaje reforzado.

El objetivo es, por tanto, evaluar si alguno de estos algoritmos puede aportar alguna ventaja significativa, y en qué condiciones sobre el mencionado sistema propio de autoescalado horizontal de Kubernetes.

## Entorno de trabajo

La pieza fundamental de nuestro laboratorio es Kubernetes. Afortunadamente podemos instalarlo de forma sencilla en nuestro ordenador personal gracias a herramientas cómo Minikube que, si bien no permite explotar todas las capacidades disponibles en un cluster de Kubernetes, sí permite trabajar con la mayoría de ellas y, desde luego, nos proporciona un entorno adecuado a nuestras necesidades [[1]](#footnote-1).

### ¿Qué es Kubernetes?

Tal y como se define en su propio sitio web “Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios” [[2]](#footnote-2). Para nuestros propósitos podemos pensar en ella como una plataforma de contenedores. En cada uno de estos contenedores se desplegará una aplicación y podremos tener varios contenedores con la misma aplicación desplegada, tal y cómo se muestra en la imagen siguiente:

Interfaz de usuario gráfica

Descripción generada automáticamente

Fig 1: Despliegue en Contenedores [[3]](#footnote-3)

Un esquema que muestra con algo más de detalle la arquitectura de kubernetes sería el siguiente:

Diagrama

Descripción generada automáticamente

Fig 2: Arquitectura de Kubernetes [[4]](#footnote-4)

### Pod

Un Pod representa en Kubernetes una unidad de ejecución de un proceso. Puede contener uno o más contenedores en los que, a su vez, se ejecuta, por regla general, una única aplicación.

Los pods tienen una vida efímera y, en el caso de que presenten fallos, el sistema puede eliminarlos y crear otros nuevos que los sustituyan. De esa forma se mantiene el servicio en alta disponibulidad.

### Deployment

El Deployment es un objeto de Kubernetes que permite declarar el estado y las características deseadas para el despliegue de una determinada aplicación. En él se especifíca información cómo pueda ser, por ejemplo, la imagen del contenedor con la aplicación a desplegar, el número de Pods que queremos se levanten, los selectores o etiquetas con los que lo identificaremos, puertos etc.

### Replica Set

Dentro del Deployment el “Replica Set” en concreto es el elemento encargado de velar por la alta disponibilidad del sistema. Su labor es, por tanto, mantener constante el número de pods activos que se haya indicado en el Deployment.

### Service

Debido la mencionada naturaleza efímera de los Pods, Kubernetes proporciona una fachada que no adolece de dicho problema y a la que se pueden dirigir las peticiones a los diferentes recursos de la aplicación. El servicio redirigirá a su vez dichas peticiones a cualquiera de los Pods que estén disponibles en ese momento.

### Namespace

Los namespaces permiten establecer clusters virtuales dentro de un cluster físico para aislar unos recursos de otros, por ejemplo para separar entornos de Desarrollo, Test y Producción, o aplicaciones de distintos equipos de trabajo. En nuestro caso crearemos un Namespace para desplegar en él los contenedores con los que vamos a establecer nuestra comparativa.

### Horizontal Pod Autoscaling

Kubernetes tiene su propio sistema para aumentar o disminuir el número de pods de un deployment de acuerdo al número de peticiones que recibe o a la carga de trabajo a la que está sometido. Lo hace mediante un tipo de recurso denominado HPA (Horizontal Pod Autoscaler) que consulta el API (Application Program Interface) del servidor de Métricas (que es necesario activar en el cluster) para decidir, en función de las reglas que se le proporcionen, si es necesario incrementar o disminuir el número de pods.

Para nuestra comparación utilizaremos cómo referencia el ejemplo descrito en la web de documentación de Kubernetes [[5]](#footnote-5).

Se trata de una página web desplegada en un servidor Apache que realiza un elevado número de operaciones con el objetivo de aumentar significativamente el consumo de CPU.

Crearemos el deployment de Kubernetes mediante el fichero yaml en el que declaramos todos los recursos que vamos a utilizar. Previamente tendremos que haber habilitado el plugin con el servidor de métricas, para que el componente HPA pueda funcionar. El deployment se denominará “php-apache” y los nombres de los pods comenzarán también con la misma cadena de caracteres. También será el nombre de nuestro Namespace.

El HPA podemos crearlo también ejecutando el siguiente comando en un terminal:



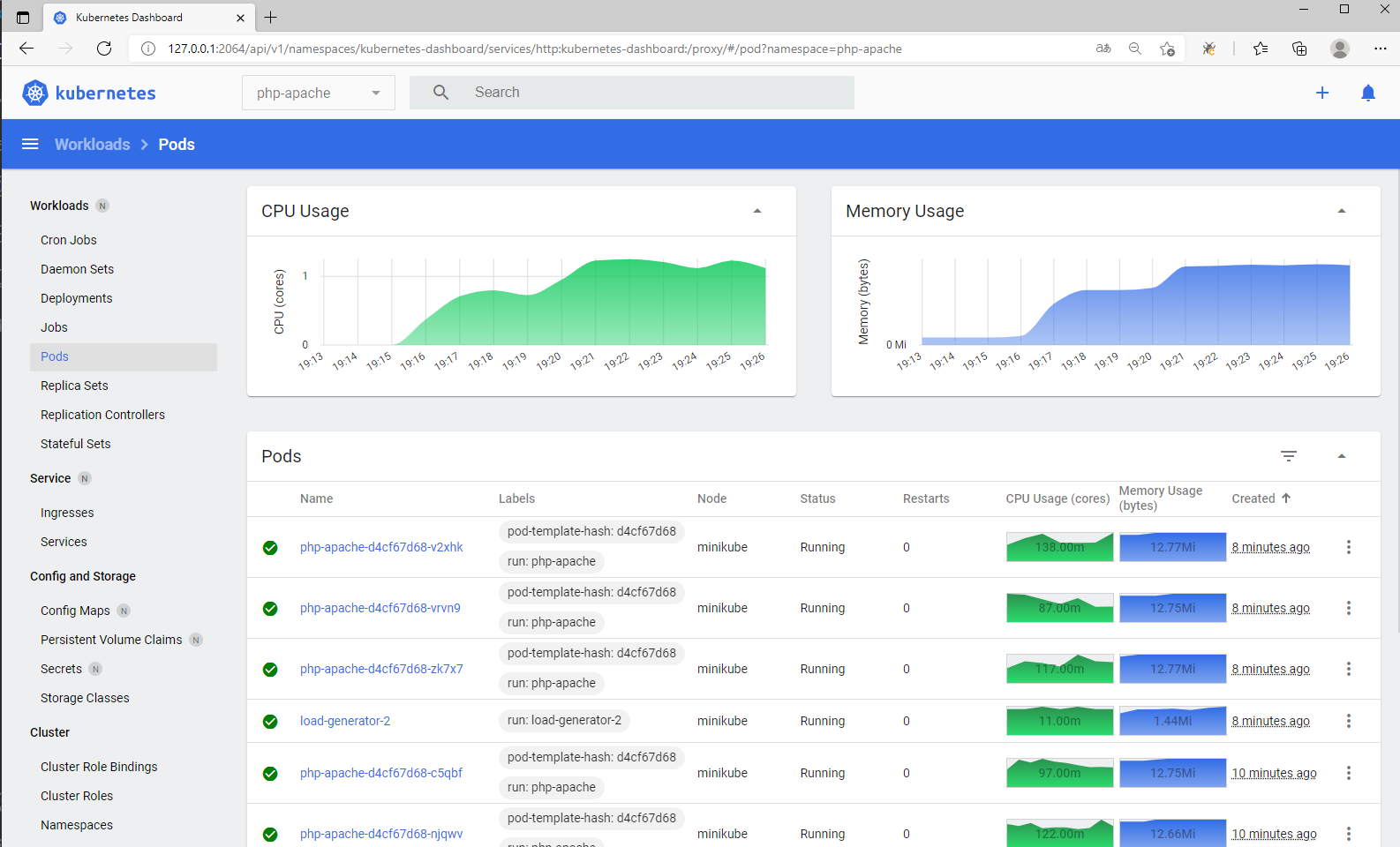
Cuando queramos aumentar el porcentaje de CPU utilizado, lo haremos ejecutando el siguiente comando:



Con el mismo crearemos un nuevo Pod en el que se ejecutará continuamente un bucle de llamadas al servidor Apache.

### Monitorización

Para poder consultar el estado del cluster de forma visual activaremos el Dashboard que nos proporciona Minikube. En la pantalla podremos ver el nivel de consumo de CPU y Memoria además de los distintos Pods que están funcionando en cada momento.



### Estado

Además de poder consultar el estado de forma visual con el Dashboard necesitaremos para poder alimentar nuestro algoritmo de aprendizaje reforzado una forma programática de obtenerlo. Utilizaremos para ello el Cliente Python de Kubernetes [[6]](#footnote-6).

Con apenas cuatro líneas de código podemos obtener gracias a esta herramienta el consumo de CPU y Memoria de cada uno de los Pods desplegados, en este caso, en el namespace “php-apache”



### Acción

Para poder aplicar también de forma programática sobre el cluster de Kubernetes el comando para aumentar o disminuir el número de Pods haremos uso de un script Python que utilizará el paquete “os” y su método “system”.



El script recibirá por parámetro el número de Pods que decida el algoritmo de aprendizaje reforzado y se lo comunicará a kubernetes para que éste actúe en consecuencia.

## ¿Qué es el Aprendizaje Reforzado?

Uno de lo rasgos diferenciales de los algoritmos de aprendizaje reforzado es que son capaces, hasta cierto, punto de aprender por sí mismos de la información que reciben sin que sea necesario “enseñarles” directamente cuales son las respuestas correctas a determinado problema y cuales no. Esta característica los diferencia de los algoritmos de Aprendizaje Supervisado si bien no los significa demasiado frente a otros algoritmos del campo denominado Aprendizaje No Supervisado, cuyo ejemplo más representativo es el clustering. La diferencia con respecto a este último, cuyo propósito principal es encontrar patrones en un conjunto de datos no etiquetados, es que el aprendizaje reforzado tiene una cierta componente de orientación hacia la consecución de un objetivo concreto, formulado mediante una función que trata de representar lo que se considera el éxito a la hora de resolver la tarea o problema.

El Aprendizaje Supervisado implica una suerte de de relación Profesor-Alumno en el que éste último, el algoritmo, puede llegar a imitar perfectamente a su profesor, el conjunto de datos etiquetados [[7]](#footnote-7). Éstos, sin embargo, pueden estar limitados en cuanto a la cantidad de la que podemos disponer además de que, en ocasiones, son muy complicados de obtener e incluso el esfuerzo para ello puede no compensar el resultado posterior.

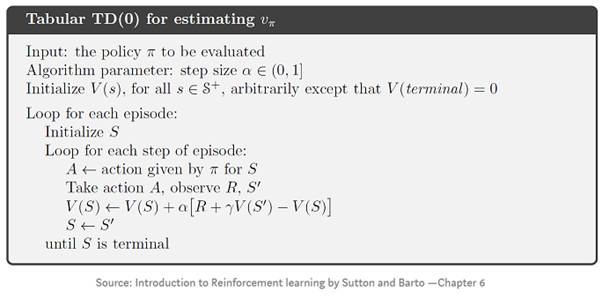
Tareas de Control o decisión frente a tareas de predicción o de clasificación (supervised). En muchas tareas de control el entorno no está controlado y produce sus propios datos además de poder ser probabilística.



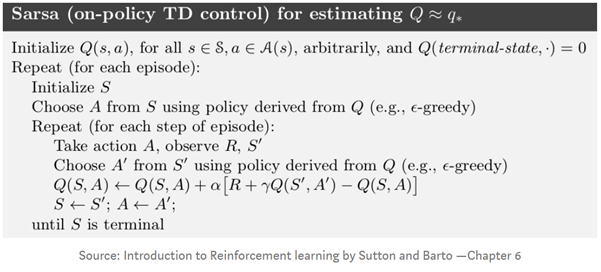
### Markov decision Processes

### Montecarlo

### Temporal Difference

**[[8]](#footnote-8)**

### SARSA

****

### Q-Learning

****

### Deep Q-Learning

## Implementación

**Environment**

**Estados**

**Acciones**

**Recompensas**

**Agente**

**Simulación**

## Resultados

## Conclusiones

## Bibliografía

Manning Publications. «Deep Reinforcement Learning in Action». Accedido 12 de julio de 2021. https://www.manning.com/books/deep-reinforcement-learning-in-action.

Kubernetes. «Horizontal Pod Autoscaler Walkthrough». Accedido 12 de julio de 2021. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/.

*kubernetes-client/python*. Python. 2016. Reprint, Kubernetes Clients, 2021. https://github.com/kubernetes-client/python.

minikube. «Minikube». Accedido 12 de julio de 2021. https://minikube.sigs.k8s.io/docs/.

Kubernetes. «¿Qué es Kubernetes?» Accedido 12 de julio de 2021. https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/.

Sutton, Richard S., y Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second edition. Cambridge, Mass: A Bradford Book, 1998.

Ushio, Tsuyoshi. «Kubernetes in Three Diagrams». Medium, 5 de febrero de 2018. https://tsuyoshiushio.medium.com/kubernetes-in-three-diagrams-6aba8432541c.

### Repositorio de Código

El código del proyecto puede encontrarse en el siguiente repositorio de GitHub:

<https://github.com/mlalandag/HPAwDRL>

## Otros Recursos

<https://github.com/kubernetes-client/python>

1. «Minikube». [↑](#footnote-ref-1)
2. «¿Qué es Kubernetes?» [↑](#footnote-ref-2)
3. «¿Qué es Kubernetes?» [↑](#footnote-ref-3)
4. Ushio, «Kubernetes in Three Diagrams». [↑](#footnote-ref-4)
5. «Horizontal Pod Autoscaler Walkthrough». [↑](#footnote-ref-5)
6. *kubernetes-client/python*. [↑](#footnote-ref-6)
7. «Deep Reinforcement Learning in Action». [↑](#footnote-ref-7)
8. Sutton y Barto, *Reinforcement Learning*. [↑](#footnote-ref-8)