# Autoescalado horizontal en Kubernetes con Aprendizaje Reforzado

Contenido

[Autoescalado horizontal en Kubernetes con Aprendizaje Reforzado 1](#_Toc76573829)

[Introducción 1](#_Toc76573830)

[Entorno de trabajo 2](#_Toc76573831)

[¿Qué es Kubernetes? 2](#_Toc76573832)

[Pod 4](#_Toc76573833)

[Deployment 4](#_Toc76573834)

[Horizontal Pod Autoscaling 4](#_Toc76573835)

[Estado 5](#_Toc76573836)

[Acción 7](#_Toc76573837)

[¿Qué es el Aprendizaje Reforzado? 7](#_Toc76573838)

[Markov decision Processes 8](#_Toc76573839)

[Montecarlo 8](#_Toc76573840)

[Temporal Difference 8](#_Toc76573841)

[SARSA 9](#_Toc76573842)

[Q-Learning 9](#_Toc76573843)

[Deep Q-Learning 9](#_Toc76573844)

[Implementación 10](#_Toc76573845)

[Resultados 10](#_Toc76573846)

[Conclusiones 10](#_Toc76573847)

## 

## 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.

<https://minikube.sigs.k8s.io/docs/>

### ¿Qué es Kubernetes?

<https://kubernetes.io/es/docs/concepts/overview/what-is-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”. 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.

Interfaz de usuario gráfica

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

### Pod

### Deployment

Activar Kubernetes Dashboard

<https://minikube.sigs.k8s.io/docs/handbook/dashboard/>

### 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 nececsario incrementar o disminuir el número de pods.

Para nuestra comparación utilizaremos cómo referencia el ejemplo descrito en la web de documentación de Kubernetes.

<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>

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

kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10 -n php-apache

Cuando queramos aumentar el porcentaje de CPU utilizado, lo haremos ejecutando el siguiente comando en un terminal.

kubectl run -i --tty load-generator --rm --image=busybox -n php-apache --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"

### Estado

#### Kubernetes Python Client

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

#### Prometheus

Prometheus es una herramienta de código abierto que permite obtener y almacenar métricas y series temporales de datos. Dispone de integraciones para numerosos sistemas, incluido kubernetes, y será el método que nos permitirá obtener la información de entrada que necesitamos para nuestro agente de aprendizaje reforzado.

Interfaz de usuario gráfica

Descripción generada automáticamente

https://prometheus.io/

Instalar Prometheus en Minikube

https://blog.marcnuri.com/prometheus-grafana-setup-minikube

kubectl get svc

minikube service prometheus-server-np --url

Metrics

**sum (rate (container\_cpu\_usage\_seconds\_total{namespace="php-apache", name!~".\*prometheus.\*", image!="", container!="POD", id!~".\*/docker/.\*"}[3m])) by (pod)**

Number of pods:

count(kube\_pod\_info{namespace="php-apache"}) by (namespace)

count(kube\_pod\_info{namespace="php-apache", pod!~".\*load.\*"}) by (namespace)

### Acción

Para poder aplicar 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”. Por ejemplo, para establecer 3 Pods en el cluster se ejecutará:

kubectl scale deployment php-apache --replicas=3 -n php-apache

## ¿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.

Aprendizaje Supervisado >> teacher-student relationship, limitados en cuanto a la cantidad de datos que podemos darles (a veces es imposible obtenerlos). Podrán hacer lo mismo que nosostros muy bien, pero no pasarán de ahí.

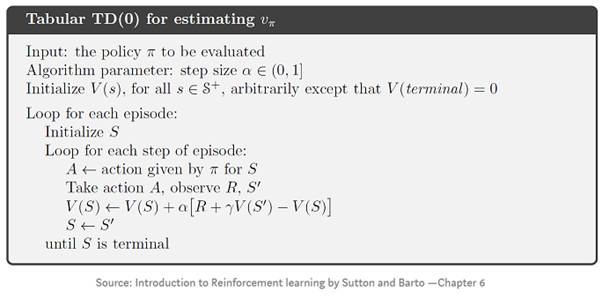
Tareas de Control o decisión frente a tareas de predicción o de clasificación (supervised). En muchas tareas de control el entorno no está controlado y produce sus propios datos además de poder ser probabilística.



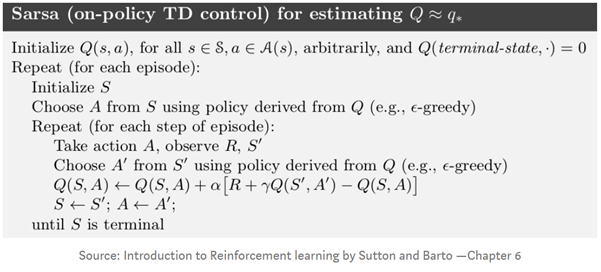
### Markov decision Processes

### Montecarlo

### Temporal Difference

****

### SARSA

****

### Q-Learning

****

### Deep Q-Learning

## Implementación

**Environment**

**Estados**

**Acciones**

**Recompensas**

**Agente**

**Simulación**

## Resultados

## Conclusiones