# Autoescalado horizontal en Kubernetes con Aprendizaje Reforzado

## 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

Activar Kubernetes Dashboard

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

### Horizontal Pod Autoscaling

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

Activar Kubernetes Metrics Server

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

## 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)**

sum (rate (container\_cpu\_usage\_seconds\_total{namespace="php-apache"}[2m])) by (pod\_name)

container\_cpu\_usage\_seconds\_total{namespace="php-apache", name!~".\*prometheus.\*", image!="", container!="POD", id!~".\*/docker/.\*"}[2m]

sum (rate (container\_cpu\_usage\_seconds\_total{namespace="php-apache"}[2m])) / sum (machine\_cpu\_cores) \* 100

¿¿???

sum(rate(container\_cpu\_usage\_seconds\_total{name!~".\*prometheus.\*", image!="", container\_name!="POD"}[5m])) by (pod\_name, container\_name) /

sum(container\_spec\_cpu\_quota{name!~".\*prometheus.\*", image!="", container\_name!="POD"}/container\_spec\_cpu\_period{name!~".\*prometheus.\*", image!="", container\_name!="POD"}) by (pod\_name, container\_name)

Number of pods:

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

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

Increase Load

kubectl run -i --tty load-generator2 --rm --image=busybox -n php-apache --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"

## Python Client

https://github.com/kubernetes-client/python/#kubernetes-python-client

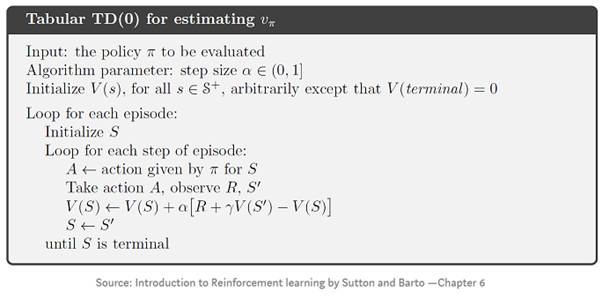
## ¿Qué es el Aprendizaje Reforzado?



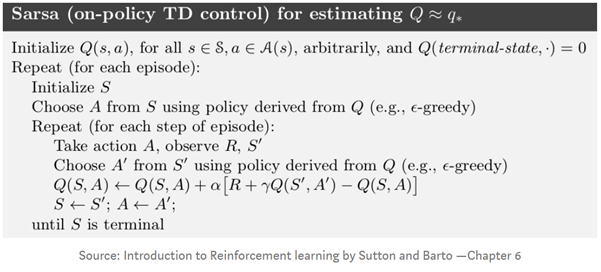
**Markov decision Processes**

**Montecarlo**

**Temporal Difference**

****

**SARSA**

****

**Q-Learning**

****

**Deep Q-Learning**

A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments

**Environment**

**Estados**

**Acciones**

**Recompensas**

**Agente**

**Simulación**

**Implementación**

**Resultados**

**Conclusiones**