



# iWHITESTACK CHALLENGE!

Whitestack Challenge #1

Abril 2024

# Contenidos

---

Contenidos

Introducción

[Bienvenidos a nuestro primer Whitestack Challenge!](#)

[La punta del iceberg](#)

Desafío

[0. Probar accesos](#)

[1. Compilar y probar el código de manera local](#)

[2. Crear una imagen de Docker de la webapp](#)

[3. Crear manifiestos k8s](#)

[4. Crear un Helm chart](#)

[5. Despliegue de la webapp en un cluster k8s](#)

[6. Actualización de una variable de entorno en la webapp](#)

[7. Actualización de la webapp desplegada](#)

Criterios de evaluación

Participación del Challenge

Entrega del Challenge



## Introducción

### ¡Bienvenidos a nuestro primer Whitestack Challenge!

Whitestack les da la bienvenida y les agradece por participar de la primera edición de los Whitestack Challenges.

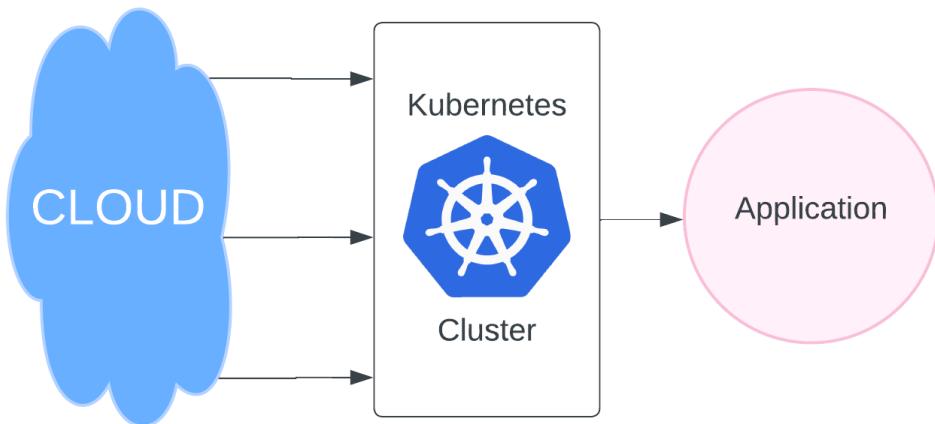
Estos serán una serie de desafíos tecnológicos centrados alrededor del emocionante mundo del *Cloud Computing*, que pondrán a prueba sus habilidades como desarrolladores y les darán la oportunidad de desarrollar sus conocimientos con experiencias de primera mano.

Adicionalmente, serán acompañados por un equipo de profesionales dedicados a estas tecnologías, quienes les brindarán apoyo durante el desafío.

¡Esperamos que esta pueda ser una experiencia enriquecedora y que juntos sigamos creciendo en la comunidad del Cloud en Latinoamérica!

### La punta del iceberg

En esta oportunidad, nuestro Whitestack Challenge será enfocado en la cima de un despliegue *Full Stack* de tecnologías de nube abiertas. Es decir, tendrán que usar una arquitectura de nube, provista por nosotros, como base para el despliegue de una aplicación. En específico, se provee un cluster de Kubernetes, desplegado sobre una nube, para que se realice el despliegue.



La aplicación seleccionada para este desafío es *Tengen-Tetris*, una web application creada usando *Python Flask* en el backend y *p5.js* en el frontend, la cual permite jugar *Tetris* en modo multijugador dentro de una misma red.

## Desafío

La misión de este desafío será lograr desplegar esta aplicación dentro del cluster de Kubernetes provisto, manteniendo intactas sus funcionalidades y permitiendo el modo de juego multijugador.

A continuación se presentarán los pasos a seguir para completar el primer Whitestack Challenge.

### 0. Probar accesos

Antes de comenzar el desafío, es importante probar el acceso al ambiente en el que vamos a desempeñar nuestras tareas.

Las interacciones con el cluster provisto se harán a través de la herramienta *kubectl*, por lo que esta debe estar instalada en su ambiente local. Los pasos de instalación de *kubectl* pueden ser vistos en la [documentación oficial de Kubernetes](#).

Para configurar *kubectl* para la comunicación con el cluster remoto usaremos un archivo configurador *kubeconfig*, el cual será entregado a cada participante con los datos específicos para su ambiente designado. Estas configuraciones le darán acceso al participante a un *namespace* del cluster. De esta manera se asegura que el trabajo de los participantes no afecte al resto.

El archivo tendrá la siguiente forma:

```
apiVersion: v1
kind: Config
clusters:
- name: "whitestackchallenge"
  cluster:
    server: "<dirección del cluster>"

users:
- name: "<usuario asignado al participante>"
  user:
    token: "<token del usuario>"

contexts:
- name: "whitestackchallenge"
  context:
    user: "<usuario asignado al participante>"
    cluster: "whitestackchallenge"

current-context: "whitestackchallenge"
```

Para configurar *kubectl* usando este archivo se debe exportar la variable de ambiente *KUBECONFIG* con su *path* como valor:

```
export KUBECONFIG=<path-al-archivo-de-config-provisto>"
```

Si las configuraciones fueron correctas, se debería ver el siguiente output al tratar de listar los pods del cluster:

```
$ kubectl get pods
No resources found in <namespace-asignado-al-participante> namespace.
```

## 1. Compilar y probar el código de manera local

Antes de poder empezar a trabajar en nuestro ambiente *Cloud*, debemos familiarizarnos con el código que se desea desplegar en este. En [este link](#) se encuentra el repositorio que contiene el código fuente de *Tengen-Tetris*.

El primer paso consiste en copiar el código fuente desde el repositorio y seguir las instrucciones para realizar un despliegue de manera local.

Adicionalmente, en este paso se debe realizar un rápido análisis del código y de las funcionalidades de la aplicación, con el fin de lograr un mejor entendimiento de estas y definir qué elementos de k8s serán necesarios para el posterior despliegue de la webapp. Prestar especial atención a puntos como: qué requisitos tiene la webapp, cómo se accede al frontend, etc.

Hint: seguir las instrucciones descritas en [el repositorio](#).

## 2. Crear una imagen de Docker de la webapp

Para poder desplegar nuestra aplicación en un cluster de Kubernetes, debemos contar con una imagen Docker que contenga el código fuente de nuestra webapp. En este paso, se deberá generar dicha imagen.

Se deben considerar los requisitos de la aplicación y como instalarlos en la imagen a través de un *Dockerfile*.

Finalmente, cuando se cuente con la imagen, se deberá subir a un repositorio de imágenes de *Docker Hub* y permitir que sea descargada públicamente.

Hint: más información en la [documentación oficial de Docker](#).

### 3. Crear manifiestos k8s

Para poder desplegar la imagen de nuestra webapp en un cluster de Kubernetes, tendrán que crear los manifiestos de los elementos k8s necesarios para levantar la aplicación y asegurar conectividad con esta.

Se debe considerar cuidadosamente la arquitectura de elementos a usar en el despliegue (*deployment*, *replicaset*, *service*, etc.) y generar un manifiesto en formato *YAML* para cada uno.

Una vez que se cuente con los manifiestos necesarios para desplegar la webapp en kubernetes, se debe probar que estos sean correctos. Esto se puede hacer realizando un despliegue en el cluster usando comandos *kubectl*. Una vez desplegados los manifiestos, debemos ser capaces de ingresar al frontend de la webapp por browser para probar las funcionalidades de la aplicación (modo un jugador y modo dos jugadores).

Luego de que se confirme el funcionamiento de la imagen de Docker de nuestra webapp y de los manifiestos k8s creados para desplegarla en el cluster, se debe desinstalar antes de continuar con los siguientes pasos.

Hint: en caso de tener dudas, ver la [documentación oficial de Kubernetes](#) (desde "Deploy an App")



### 4. Crear un Helm chart

Al querer manejar nuestro despliegue con Helm, se hace necesario crear un Helm chart. Este debe permitirnos desplegar todos los elementos k8s previamente creados, llegando al mismo estado que el despliegue manual usando comandos *kubectl*.

Adicionalmente, este chart debe permitirnos configurar las características del despliegue a través de un archivo de values. Se debe poder afinar detalles como: recursos requeridos, número de réplicas, imagen a usar, etc.

Hint: para más información sobre la creación de Helm charts, ver la [documentación oficial](#).

### 5. Despliegue de la webapp en un cluster k8s

Una vez que se cuente con el Helm chart para nuestro despliegue, crear un archivo de values con las configuraciones necesarias para este y realizar el despliegue en el cluster provisto.

Al terminar el despliegue, verificar su funcionamiento del mismo modo en que se hizo en el paso tres.

## 6. Actualización de una variable de entorno en la webapp

Una vez desplegada la webapp en el cluster haciendo uso de su imagen Docker, manifiestos k8s y Helm chart, se deberá realizar una actualización al código.

Especificamente, queremos ser capaces de configurar el puerto y la IP donde se expone el frontend de nuestra aplicación. Para esto, se deberá editar el código fuente de nuestra webapp para permitir definir el IP y puerto a usar. Una vez hechos los cambios, se hará necesario crear una nueva imagen de docker que contenga la versión actualizada del código. Se tiene completa libertad para elegir cómo implementar este cambio, pero se deben tener en cuenta las buenas prácticas de desarrollo al momento de hacerlo.

Estos campos deben poder ser configurados a través de el archivo de values y propagados correctamente a los manifiestos k8s por Helm, es entonces necesario generar nuevas versiones de los manifiestos y el chart, teniendo en consideración los cambios hechos al código.

Hint: el IP y puerto usados por la aplicación se encuentran hardcodeados en [esta línea](#).

## 7. Actualización de la webapp desplegada

Al contar con la imagen de Docker actualizada, y con los manifiestos y el chart nuevo, se deberá actualizar la aplicación previamente levantada en el cluster, asegurando una afectación mínima de su funcionamiento.

Sin desinstalar el despliegue que debe estar corriendo en el cluster, se deberá actualizar a la última versión del código, con nuevos valores para el puerto y el IP donde se expone el frontend.

## 8. Documentación de proceso

Como parte de este challenge, se espera que los participantes documenten el trabajo realizado.

En este documento, y sin entrar en mucho detalle, se deben justificar las decisiones de diseño tomadas, como se resolvieron cada uno de los pasos y cualquier problema que se pueda presentar durante el challenge, junto con la forma en la que se soluciona.

## Criterios de evaluación

Los puntos que serán evaluados de este challenge serán los siguientes:

- La imagen Docker creada: esta debe ser desplegable en un contenedor y contener el código de la webapp provista.
- Los manifiestos k8s: Se debe haber diseñado una estructura de elementos k8s que se aadecue a las necesidades del despliegue, generando los manifiestos para cada uno.
- El Helm chart producido: Este debe seguir las mejores prácticas de diseño y contar con la características de personalización del despliegue requeridas.
- La aplicación desplegada en el cluster: Esta se debe encontrar funcional y desplegada con un diseño de elementos k8s adecuado
- Acceso a la app desde el cluster: Debe ser posible interactuar con la aplicación a través de sus servicios para probar su funcionamiento.
- Habilidad de actualización: se debe demostrar la capacidad de actualizar el software desplegado en el cluster a una nueva versión, adecuando el Helm chart para reflejar estos cambios.
- Documentación: Se debe haber documentado el trabajo realizado durante el challenge de manera clara y concisa
- Creatividad: Se considerará el nivel de innovación y originalidad al momento de desarrollar el challenge.

## Participación del Challenge

Recuerda que para participar en el Challenge debes ratificar tu inscripción, siguiendo las instrucciones indicadas en:

<https://whitestack.com/es/whitestack-challenge/>

## Entrega del Challenge

El desafío puede ser desarrollado dentro de la Infraestructura que Whitestack ha dispuesto para este propósito. O bien, en caso que el participante lo desee, puede ser desarrollado dentro de un ambiente que la misma persona levante para el desarrollo del reto.

En caso que realices el desafío dentro de la infraestructura dispuesta por Whitestack, una vez concluido el plazo, podremos ver directamente el resultado de tu trabajo.

En caso que realices el desafío en tu propio ambiente, el proyecto debe ser subido a un repositorio GIT, y una vez finalizado, debes escribir a [whitestackchallenge@whitestack.com](mailto:whitestackchallenge@whitestack.com), indicando que has finalizado el Challenge #| e incluir el link al repositorio GIT para revisar tu trabajo.

**SI TIENES CUALQUIER DUDA NOS PUEDES ESCRIBIR  
A [WHITESTACKCHALLENGE@WHITESTACK.COM](mailto:WHITESTACKCHALLENGE@WHITESTACK.COM)**

**iWHITESTACK  
CHALLENGE!**

# **iWHITESTACK CHALLENGE!**