

Proyecto Final de Integración Continua y Entrega Continua (CI/CD)

Duración estimada: 60 Minutos



Bienvenido al entorno de desarrollo del **Proyecto Final de Integración Continua y Entrega Continua (CI/CD)**. Ahora es el momento de aplicar todo lo que has aprendido en los módulos anteriores de este curso. Este entorno de laboratorio te proporcionará una aplicación de muestra y un clúster de OpenShift, que te permitirá llevar a cabo los siguientes objetivos:

Objetivos

- Crear un pipeline de CI en GitHub Actions con pasos para **linting** y **pruebas unitarias**.
- Usar Tekton para crear tareas para **linting**, **pruebas unitarias** y **construcción de una imagen**.
- Crear un Pipeline de CI en OpenShift que utilice los pasos de Tekton creados previamente.
- Añadir el paso de despliegue al pipeline de OpenShift que despliega el código en el clúster de OpenShift del laboratorio.

Debes completar todo el trabajo en el proyecto final en este entorno de laboratorio.

Requisitos previos

Información de seguridad importante

Bienvenido al Cloud IDE con OpenShift. Este entorno de laboratorio es donde se llevará a cabo todo su desarrollo. Tiene todas las herramientas que necesitará, incluido un clúster de OpenShift.

Es esencial entender que el entorno de laboratorio es **efímero**. Solo existe por un corto periodo y luego será destruido. Por lo tanto, debe enviar todos los cambios realizados a su propio repositorio de GitHub para recrearlo en un nuevo entorno de laboratorio, siempre que sea necesario.

Además, tenga en cuenta que este entorno es compartido y, por lo tanto, no es seguro. No debe almacenar información personal, nombres de usuario, contraseñas o tokens de acceso en este entorno para ningún propósito.

Su tarea

1. Si aún necesita generar un **Token de Acceso Personal de GitHub**, debe hacerlo ahora. Lo necesitará para enviar código de vuelta a su repositorio. Debe tener permisos repo y write y configurarse para expirar en 60 días. Cuando Git le pida una contraseña en el entorno de Cloud IDE, use su Token de Acceso Personal en su lugar.
2. Puede recrear este entorno realizando **Inicializar Entorno de Desarrollo** cada vez.
3. Cree un repositorio a partir de la plantilla de GitHub proporcionada para este laboratorio en el siguiente paso.

Crea tu propio repositorio en GitHub

Necesitarás tu repositorio para completar el proyecto final. Hemos proporcionado una Plantilla de GitHub para crear tu repositorio en tu propia cuenta de GitHub. **No hagas un Fork del repositorio ya que ya es una plantilla**. Esto evitará confusiones al hacer Pull Requests en el futuro.

Tu tarea

1. En un navegador, visita este repositorio de GitHub:
<https://github.com/ibm-developer-skills-network/vselh-ci-cd-final-project-template>
2. Desde la pestaña **Code** de GitHub, utiliza el botón verde **Use this template** para crear tu repositorio a partir de esta plantilla.
3. Selecciona **Create a new repository** del menú desplegable. En la siguiente pantalla, completa estos campos siguiendo la captura de pantalla a continuación:



vselh-ci-cd-final-project-template

Public template

generated from [ibm-developer-skills-network/coding-project-template](#)

main ▼1 branch0 tagsGo to file

captainfedoraskillup cleaning up setup file

1. Selecciona tu cuenta de GitHub de la lista desplegable.
2. Nombra el nuevo repositorio: `ci-cd-final-project`.
3. (Opcional) Agrega una descripción para que las personas sepan para qué es este repositorio.
4. Haz que el repositorio sea **Público** para que otros puedan verlo (y calificarlo).
5. Utiliza **Create repository from template** para crear el repositorio en tu cuenta de GitHub.

Create a new repository

A repository contains all project files, including the revision history. Already have elsewhere? [Import a repository](#).

Required fields are marked with an asterisk ().*

Repository template



ibm-developer-skills-network/vselh-ci-cd-final-project-template ▾

Start your repository with a template repository's contents.

☐

Include all branches

Copy all branches from ibm-developer-skills-network/vselh-ci-cd-final-project-template branch.

Owner *



captainfedoraskillup ▾

Repository name *

/

ci-cd-final-project



ci-cd-final-project is available.

Great repository names are short and memorable. Need inspiration? How about [1](#)

Description (optional)

Final project for CI/CD course



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



You are creating a public repository in your personal account.

Nota: Estos pasos solo necesitan hacerse una vez. Cada vez que vuelvas a entrar a este laboratorio, deberías comenzar desde la siguiente página, **Inicializar el Entorno de Desarrollo**.

Inicializar el Entorno de Desarrollo

Como se cubrió anteriormente, el entorno de Cloud IDE con OpenShift es efimero y puede eliminarse en cualquier momento. El entorno de Cloud IDE con OpenShift creará un nuevo entorno la próxima vez que ingreses al laboratorio. Desafortunadamente, necesitarás inicializar tu entorno de desarrollo cada vez. Esto no debería ocurrir con demasiada frecuencia, ya que el entorno puede durar varios días a la vez, pero cuando se haya ido, este es el procedimiento para recrearlo.

Resumen

Cada vez que necesites configurar tu entorno de desarrollo en el laboratorio, deberás ejecutar tres comandos.

Cada comando se explicará en mayor detalle, uno a la vez, en la siguiente sección.

{tu_cuenta_de_github} representa tu nombre de usuario de GitHub.

Los comandos incluyen:

```
git clone https://github.com/{your_github_account}/ci-cd-final-project.git
cd ci-cd-final-project
bash ./bin/setup.sh
exit
```

Ahora, discutamos estos comandos y expliquemos qué se necesita hacer.

Detalles de la tarea

Inicializa tu entorno utilizando los siguientes pasos:

1. Abre una terminal con Terminal -> New Terminal si no hay una abierta ya.
2. A continuación, utiliza el comando `export GITHUB_ACCOUNT=` para exportar una variable de entorno que contenga tu cuenta de GitHub.

Nota: Sustituye tu verdadera cuenta de GitHub que utilizaste para crear el repositorio por el marcador de posición {your_github_account} a continuación:

```
export GITHUB_ACCOUNT={your_github_account}
```

3. Luego utiliza los siguientes comandos para clonar tu repositorio, cambiar al directorio `devops-capstone-project` y ejecutar el comando `./bin/setup.sh`.

```
git clone https://github.com/{your_github_account}/ci-cd-final-project.git
cd ci-cd-final-project
bash ./bin/setup.sh
```

Deberías ver lo siguiente al final de la ejecución de la configuración:

```

! Problems theia:ci-cd-final-project x

Requirement already satisfied: urllib3<3,>=1.21.1 in /home/theia/ven
2.22.0->httpie==3.2.1->-r requirements.txt (line 23)) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /home/theia/ven
2.22.0->httpie==3.2.1->-r requirements.txt (line 23)) (2023.7.22)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /home/theia/
s]>=2.22.0->httpie==3.2.1->-r requirements.txt (line 23)) (1.7.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in /home/theia/
httpie==3.2.1->-r requirements.txt (line 23)) (3.0.0)
Requirement already satisfied: mdurl~=0.1 in /home/theia/venv/lib/pv
ch>=9.10.0->httpie==3.2.1->-r requirements.txt (line 23)) (0.1.2)
*****
CI/CD Final Project Environment Setup Complete
*****

Use 'exit' to close this terminal and open a new one to initialize t

```

4. Finalmente, cierra la terminal actual utilizando el comando `exit`. El entorno no estará completamente activo hasta que abras una nueva terminal en el siguiente paso.

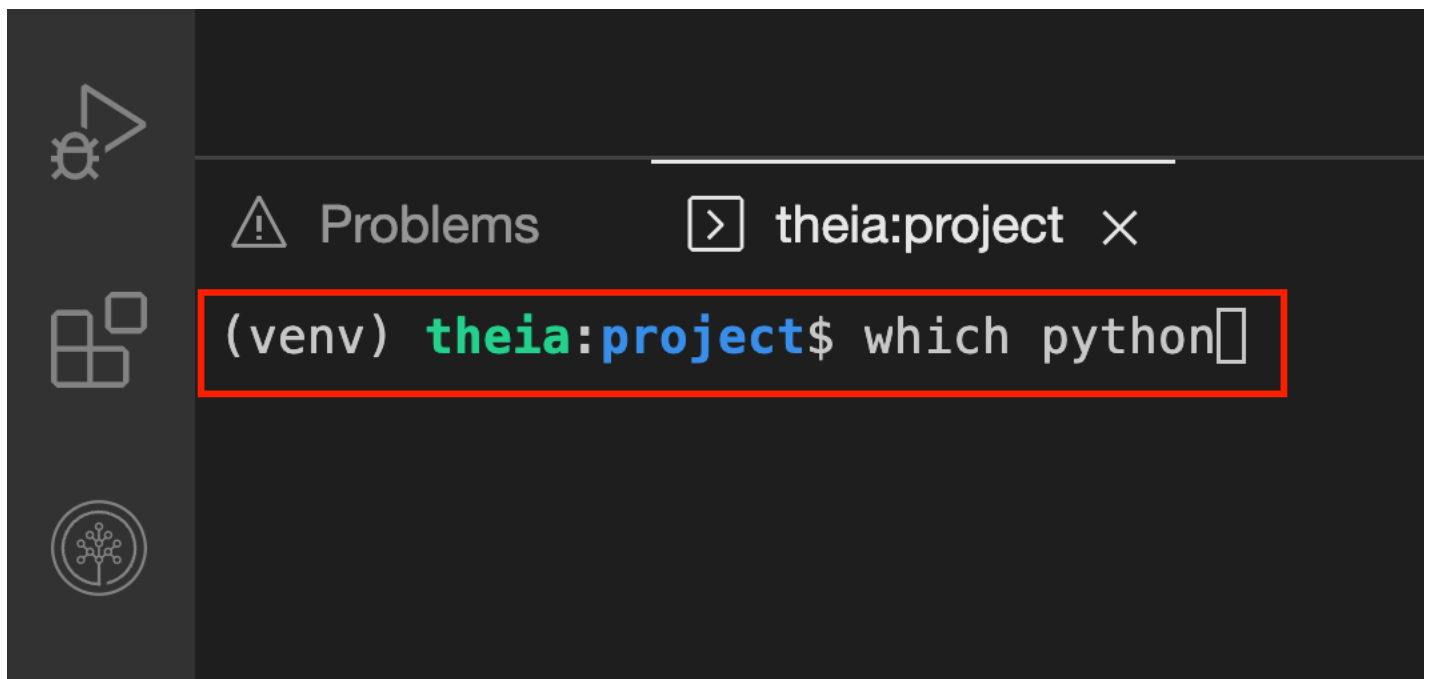
`exit`

Validar

Para validar que tu entorno está funcionando correctamente, debes abrir una nueva terminal porque el entorno virtual de Python solo se activará cuando haya una nueva terminal presente. Deberías haber terminado la tarea anterior utilizando el comando `exit` para salir de la terminal.

1. Abre una terminal con `Terminal -> New Terminal` y verifica que todo haya funcionado correctamente utilizando el comando `which python`:

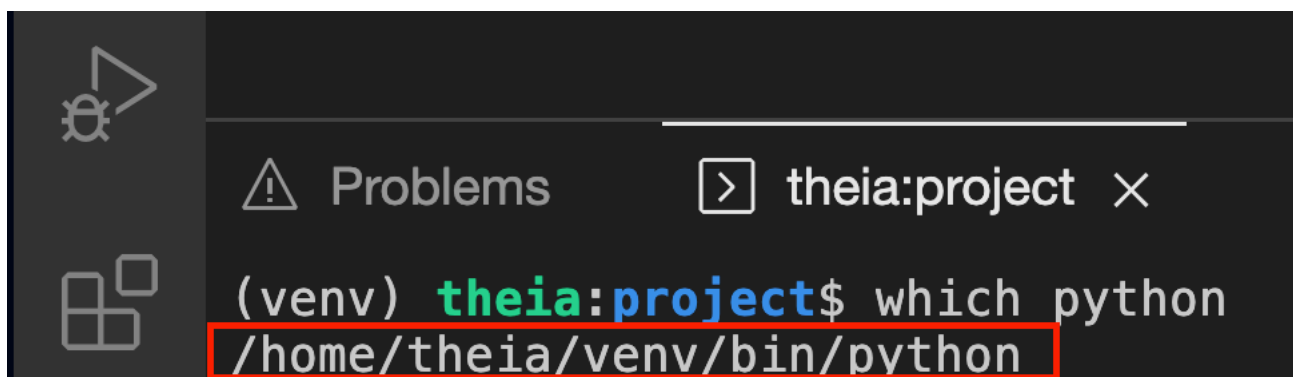
Tu indicador debería verse así:



Verifica qué Python estás utilizando:

```
which python
```

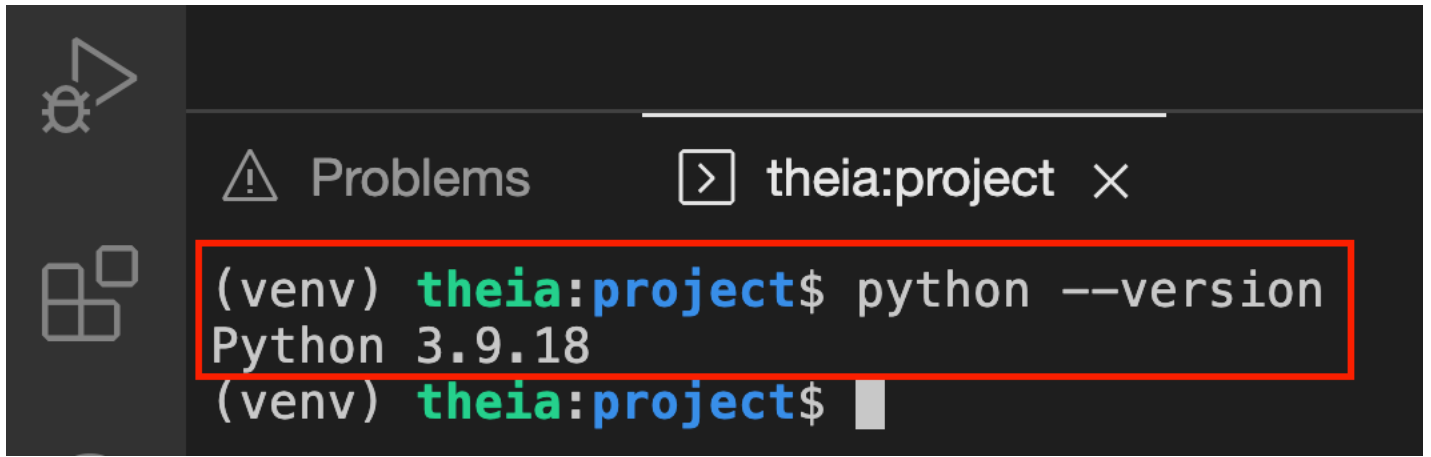
Deberías obtener:



Verifica la versión de Python:

```
python --version
```

Deberías obtener algún nivel de parche de Python 3.8:



```
(venv) theia:project$ python --version
Python 3.9.18
(venv) theia:project$
```

Esto completa la configuración del entorno de desarrollo.

Ahora estás listo para comenzar a trabajar.

Escenario del proyecto final

Eres parte de un equipo responsable de construir un microservicio innovador, una API RESTful que permite a los usuarios gestionar y rastrear contadores. Otro equipo ya ha desarrollado la interfaz de usuario (UI) para este microservicio, y ahora es tu turno de asegurar la fiabilidad y eficiencia de los servicios de backend.

Integración Continua (CI) con GitHub Actions

Tu primera tarea es configurar pipelines de CI utilizando GitHub Actions. La base de código incluye pruebas unitarias para los endpoints proporcionados. Tu objetivo es automatizar los procesos de linting y pruebas. Crearás un flujo de trabajo de GitHub Actions que se activará cada vez que se realicen cambios en el repositorio.

Despliegue Continuo (CD) con OpenShift Pipelines

En la segunda fase, establece pipelines de CD dentro de OpenShift Pipelines. Estos pipelines deben incluir linting, pruebas, construcción de una imagen y el despliegue sin problemas del microservicio en un clúster de OpenShift.

Necesitas proporcionar la URL de tu repositorio con el flujo de trabajo de GitHub y los archivos yaml de tekton, además de otras capturas de pantalla como evidencia de tu trabajo. Tu evidencia será esencial para la evaluación del proyecto por parte de tus compañeros. ¡Buena suerte con tu proyecto!

Ejercicio 1: Crear flujo de trabajo básico

Tu repositorio de GitHub tiene un archivo de flujo de trabajo vacío, `.github/workflows/workflow.yml`. Crearás el flujo de trabajo de CI escribiendo varios pasos en este archivo de flujo de trabajo.

[Open workflow.yml in IDE](#)

Tu tarea

Abre el archivo `.github/workflows/workflow.yml` y añade lo siguiente:

1. name: Flujo de trabajo de CI
2. workflow triggers: push en la rama principal y pull_request en la rama principal
3. Jobs
 - runs-on: ubuntu-latest
 - container: python:3.9-slim
4. Paso de Checkout:
 - name: Checkout
 - uses: actions/checkout@v3
5. Paso de Instalación de Dependencias:
 - name: Instalar dependencias
 - run python -m pip install --upgrade pip y pip install -r requirements.txt comandos

También puedes referirte a los videos y laboratorios en el módulo 2 del curso en caso de que quieras familiarizarte con los conceptos antes de continuar.

Sugerencia

► Haz clic aquí para una sugerencia.

Ejercicio 2: Agrega el paso de linting al flujo de trabajo de CI

A continuación, agregarás el paso de Lint al flujo de trabajo de GitHub. Utilizarás el módulo Flake8 para el linting. Abre el archivo `.github/workflows/workflow.yml` y completa las siguientes tareas.

[Open workflow.yml in IDE](#)

Tu tarea

Agrega una tarea de linting con los siguientes detalles:

1. name: Lint con flake8
2. commands:
 - o flake8 service --count --select=E9,F63,F7,F82 --show-source --statistics
 - o flake8 service --count --max-complexity=10 --max-line-length=127 --statistics

Puedes consultar los videos y laboratorios en el módulo 2 para obtener ayuda.

Sugerencia

► Haz clic aquí para una sugerencia.

Ejercicio 3: Agregar el paso de prueba al flujo de trabajo de CI

A continuación, agregarás el paso Test al flujo de trabajo de GitHub. Utilizarás el módulo Nose para ejecutar las pruebas. Abre el archivo `.github/workflows/workflow.yml` y completa las siguientes tareas.

Open **workflow.yml** in IDE

Tu tarea

Agrega un paso de prueba con los siguientes detalles:

1. name: Ejecutar pruebas unitarias con nose
2. command:
 - o nosetests -v --with-spec --spec-color --with-coverage --cover-package=app

Puedes consultar los videos y laboratorios en el módulo 2 para obtener ayuda.

Sugerencia

► Haz clic aquí para una sugerencia.

Ejercicio 4: Subir código CI a GitHub

Para probar el flujo de trabajo y la canalización de CI, necesitas confirmar los cambios y subir tu rama de vuelta al repositorio de GitHub. Como se describió anteriormente, cada nuevo empuje a la rama principal debería activar el flujo de trabajo.

Tu tarea

1. Configura la cuenta de Git con tu correo electrónico y nombre usando los comandos `git config --global user.email` y `git config --global user.name`.
 - Haz clic aquí para una pista.
2. El siguiente paso es preparar todos los cambios que hiciste en los ejercicios anteriores y subirlos a tu repositorio bifurcado en GitHub.
 - Haz clic aquí para una pista.

Tu salida debería verse similar a la imagen a continuación:

Solución



22 | - name: Lint with flake8

☐ theia:ci-cd-final-project ×

```
(venv) theia:ci-cd-final-project$ git push
Username for 'https://github.com': captainfedoraskillup
Password for 'https://captainfedoraskillup@github.com':
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 772 bytes | 772.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/captainfedoraskillup/ci-cd-final-project.git
    ba641b6..dc475f8  main -> main
(venv) theia:ci-cd-final-project$
```

Ejercicio 5: Validar el flujo de trabajo de GitHub Actions

Para validar que tu flujo de trabajo se ejecutó y fue exitoso, simplemente ve a tu versión del repositorio en GitHub y haz clic en Actions.



captainfedoraskillup / ci-cd-final-project


<> Code

Issues

Pull requests

Actions

Projects





 **ci-cd-final-project** Public

generated from [ibm-developer-skills-network/vselh-ci-cd-final-project-temp](#)

 main ▾

 1 branch

 0 tags

	captainfedoraskillup COMMIT MESSAGE	
	.github/workflows	COMMIT MESSAGE
	.tekton	Initial commit
	bin	Initial commit

Puedes hacer clic en el CI workflow para ver más detalles.

The screenshot shows the GitHub Actions page for the repository 'captainfedoraskillup / ci-cd-final-project'. The top navigation bar includes links for Code, Issues, Pull requests, Actions (which is highlighted), Projects, and a book icon. Below the navigation bar, the 'Actions' section is active, displaying a list of workflows: 'All workflows' (highlighted with a red box) and 'CI workflow'. To the right, the 'All workflows' section shows 'Showing runs from all workflow' and a box indicating '2 workflow runs'. One of the runs is highlighted with a red box, showing a green checkmark and the text 'COMMIT MESSAGE' and 'CI workflow #2: Commit dc4'. Below the workflows, there are links for 'Management' and 'Caches'.

Finalmente, puedes profundizar en la acción para confirmar que todos los pasos se completaron con éxito. Toma una captura de pantalla como se indica y nombra el archivo `cicd-github-validate(.png/jpg)`.

[← CI workflow](#)

✓ COMMIT MESSAGE #2

[Summary](#)

Jobs

✓ build

Run details

Usage

Workflow file

build

succeeded 5 minutes ago in 24

- > ✓ Set up job
- > ✓ Initialize containers
- > ✓ Checkout
- > ✓ Install dependencies
- > ✓ Lint with flake8
- > ✓ Run unit tests with
- > ✓ Post Checkout
- > ✓ Stop containers
- > ✓ Complete job

Ejercicio 6: Crear tarea de limpieza Tekton

Felicitaciones por haber creado con éxito el flujo de trabajo de CI de GitHub para extraer, analizar y probar tu código. El siguiente paso es crear el flujo de trabajo de CD en OpenShift. Antes de poder hacerlo, crea la tarea de limpieza que limpiará el espacio de trabajo de salida para que la canalización de CD pueda comenzar de nuevo.

Abre el archivo `.tekton/tasks.yml` y completa las siguientes tareas.

[Open tasks.yml in IDE](#)

Tu tarea

Agrega una tarea de limpieza con los siguientes detalles:

1. apiVersion: tekton.dev/v1beta1
2. kind: Task
3. name: cleanup

4. spec.workspaces.name: source

Esta tarea tendrá un solo paso llamado remove de la siguiente manera:

```
1. name: remove
2. image: alpine:3
3. env:
  o name: WORKSPACE_SOURCE_PATH
  o value: ${workspaces.source.path}
4. workingDir: ${workspaces.source.path}
5. securityContext
  o runAsNonRoot: false
  o runAsUser: 0
6. script:
  #!/usr/bin/env sh
  set -eu
  echo "Eliminando todos los archivos de ${WORKSPACE_SOURCE_PATH} ..."
  # Eliminar cualquier contenido existente del directorio si existe.
  #
  # No solo hacemos "rm -rf ${WORKSPACE_SOURCE_PATH}" porque ${WORKSPACE_SOURCE_PATH} podría ser "/"
  # o la raíz de un volumen montado.
  if [ -d "${WORKSPACE_SOURCE_PATH}" ] ; then
    # Eliminar archivos y directorios no ocultos
    rm -rf "${WORKSPACE_SOURCE_PATH:?}"/.*
    # Eliminar archivos y directorios que comienzan con . pero excluyendo ..
    rm -rf "${WORKSPACE_SOURCE_PATH}"/.[!..]*
    # Eliminar archivos y directorios que comienzan con .. más cualquier otro carácter
    rm -rf "${WORKSPACE_SOURCE_PATH}"/..?*
  fi
```

También puedes consultar los videos y laboratorios en el módulo 3 del curso en caso de que desees familiarizarte con los conceptos antes de continuar.

Sugerencia