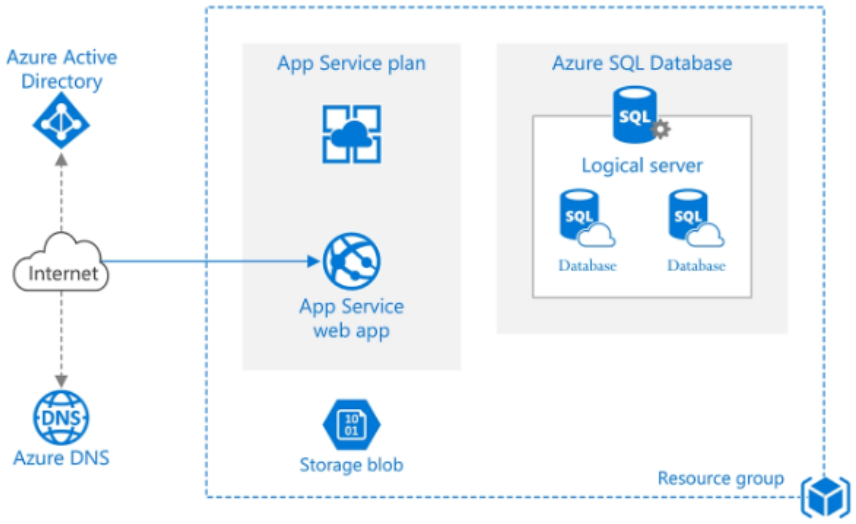
# Azure App Service

## Introducció

Azure App Service és  un servei al núvol que permet desplegar aplicacions web.




En aquesta activitat formativa farem servir d'exemple l'aplicació python que es troba a https://gitlab.com/xtec/flask.

**La manera de distribuir aplicacions web avui en dia es mitjançant imatges que serveixen per crear contenidors.**

Com veurem a continuació és un procediment molt senzill per crear aplicacions que no depenen del sistema operatiu concret ni de les llibreries que té instal.lat.

Una imatge guardada en un registre es pot descarregar, crear un contenidor a partir de la imatge i executar l'aplicació en qualsevol sistema operatiu Linux de manera immediata i predictible.

Aquest són exemples d'aplicacions desplegats amb contenidors:

- https://xtec-flask.azurewebsites.net
- https://dorayenny3434.azurewebsites.net/

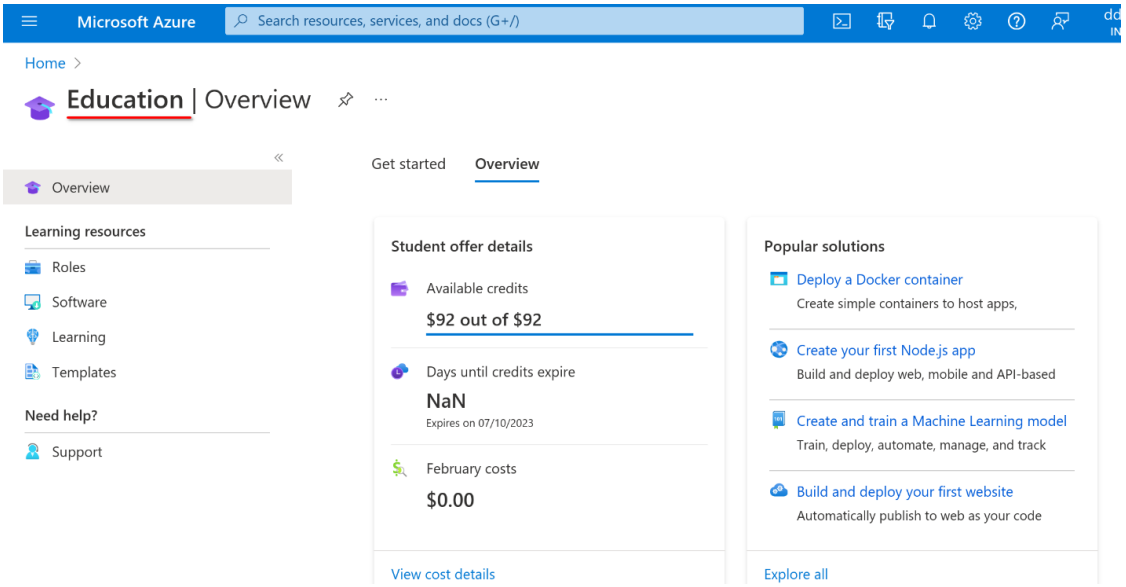
## 01- Azure

https://azure.microsoft.com/es-es/free/students/

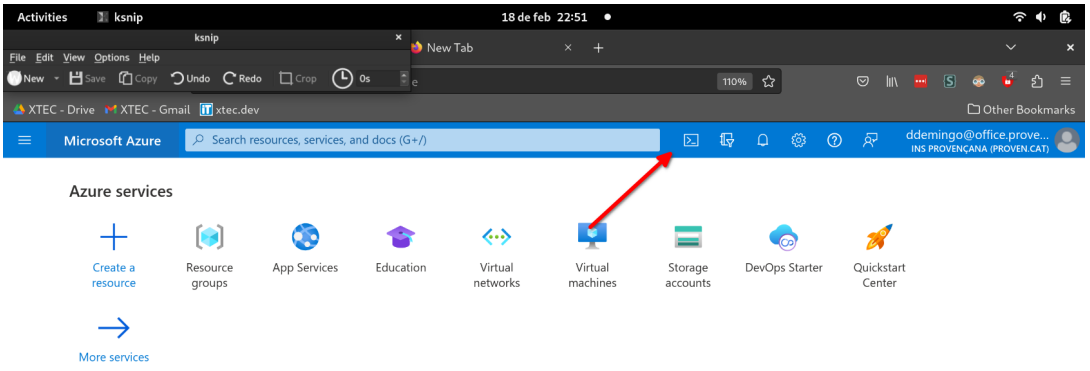
1. **Ves al portal d'Azure i autentica't amb el teu compte d'estudiant.**

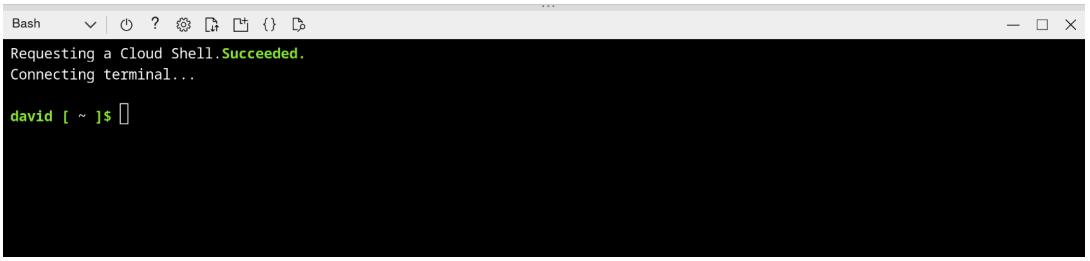
[https://portal.azure.com](https://portal.azure.com)

Un cop dins ves a Education i solicita el teu credit gratuït.




2. **Per gestionar els recursos d'Azure pots fer servir el cloud shell.**



**Ara treballarem a la cloud shell d'Azure**

## 02 -Resource Group

El primer que heu de fer és crear un grup de recursos.

Un grup de recursos permet gestionar de manera conjunta un conjunt de recursos i tot recurs ha de pertànyer a un grup de recursos.

Per example, si elimino el grup de recursos, tots els recursos que pertanyen al grup seran eliminats.

Un grup de recursos és un recurs per si mateix, i s'ha de crear en una **regió concreta,** encara que els recursos que pertanyen al grup poden estar en qualsevol regió.

1. Per veure les regions disponibles pots executar:

```
$ az account list-locations -o table
DisplayName               Name                 RegionalDisplayName
East US                   eastus               (US) East US
East US 2                 eastus2              (US) East US 2
South Central US          southcentralus       (US) South Central US
...
```

També pots veure el map

https://infrastructuremap.microsoft.com

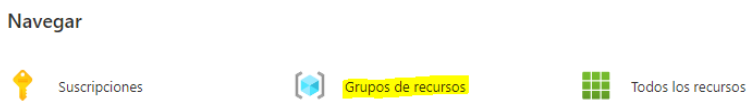
https://map-cdn.buildazure.com/#

https://azure.microsoft.com/en-us/explore/global-infrastructure/geographies/#overview

2. Crea un **grup de recursos** a `westeurope`

```
az group create --name dawbio --location westeurope
```

Reviseu el **grupo de  recursos** en el Portal Azure




## 03 -AppService

A continuació **crearem un recurs de servei d'aplicacions web.**

Azure permet crear un únic recurs gratuït per regió. Una de les limitacions d'aquest sku és que només permet 60 minuts d'execució al dia (el servei només funciona sota demanda).

1. **Crea un appservice plan a `francecentral`:**

```
az appservice plan create --name dawbio-app-service --resource-group dawbio
--location francecentral --is-linux --sku F1
```

2. Verifiqueu al portal Azure

**Apliació web**

3. El proper pas és crea una aplicació web.

El espai de noms de les aplicacions web està compartit per totes les aplicacions d'Azure. Per tant has de triar un nom que sigui únic:

```
az webapp create --name dawbio-thebest --resource-group dawbio --plan
dawbio-app-service --runtime 'PYTHON|3.9'
```

Un dels paràmetres que hem de passar és un runtime.

Per desplegar aplicacions web, azure proporciona runtimes concrets per Java, Python i altres llenguatges. En un model tradicional el servidor ha de preparar un entorn per poder executar una aplicació web.

En aquest cas es demana un entorn amb python, i concretament, amb la versió 3.9.

4. Pots veure les aplicacions web del grup de recursos amb aquesta comanda:

```
az webapp list --resource-group dawbio -o table
```

## 04 -Contenidor

A  l'igual que els altres clouds públics, Azure també proporciona desplegament directe de contenidors.

Al cap i a la fi, els runtimes són contenidors que crea Azure per executar la teva aplicació.

Per tant, és més senzill proporcionar directament el contenidor.

1. Configura l'aplicació web perquè faci servir la imatge **`registry.gitlab.com/xtec/flask`**

```
az webapp config container set --docker-custom-image-name
registry.gitlab.com/xtec/flask --name dawbio-thebest --resource-group dawbio
```

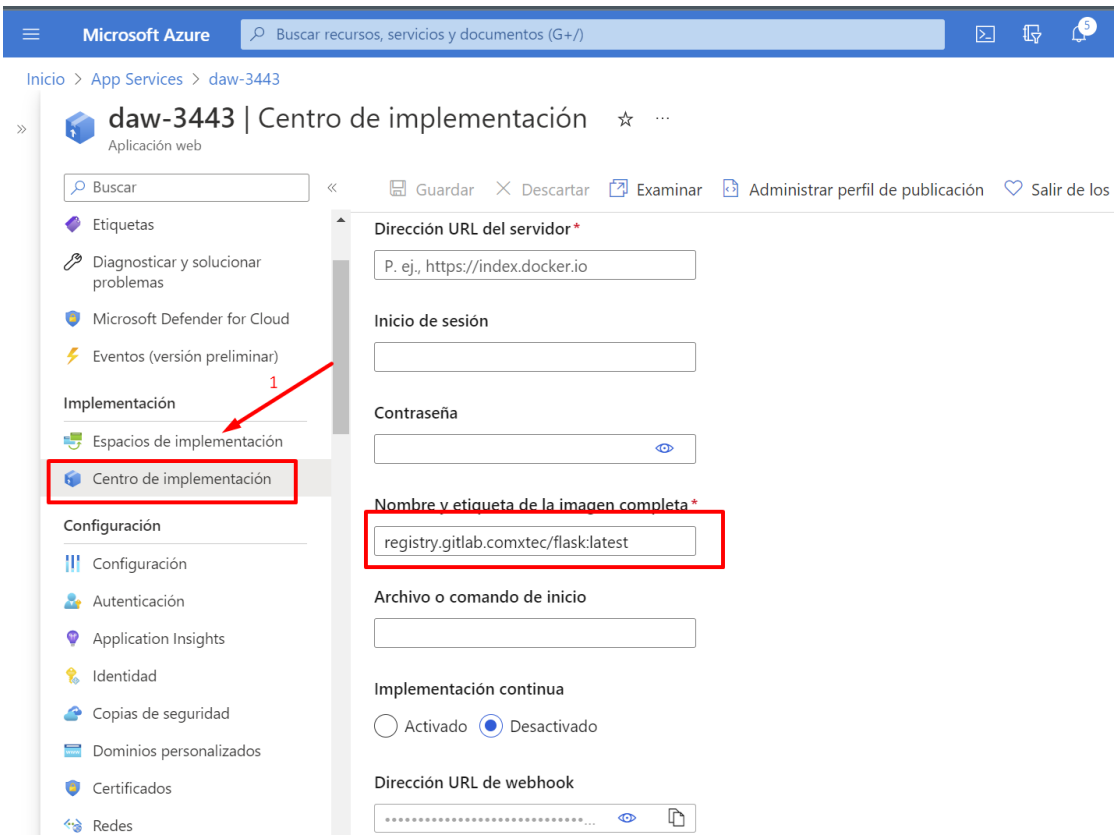
2. Pots veure com s'han aplicat els canvis amb la comanda que mostra la configuració del contenidor:

```
az webapp config container show --name dawbio-thebest --resource-group dawbio
```

El resultat serà semblant a aquest:

```
[
  {
      "name": "WEBSITES_ENABLE_APP_SERVICE_STORAGE",
      "slotSetting": false,
      "value": "false"
  },
  {
      "name": "DOCKER_CUSTOM_IMAGE_NAME",
      "value": "DOCKER|registry.gitlab.com/xtec/flask"
  }
]
```

3. També pots anar al portal d'Azure i a Centro de implementación i comprovar-ho:




**Hostname**

4. Si vols saber el hostname de les aplicacions web pots utilitzar aquesta comands:

```
az webapp list --resource-group  dawbio --query '[].hostNames' --output tsv
```

El resultat mostra les adresses web:

```
dawbio-thebest.azurewebsites.net
```

Navega a un d'aquest llocs web, per exemple: http://dawbio-thebest.azurewebsites.net/

## 05 - Administració de l'aplicació

https://learn.microsoft.com/es-es/azure/app-service/resources-kudu#kudu-features

En tot moment pots anar directament a la pàgina de control de l'aplicació web (Kudu)

Obre la URL en el navegador:

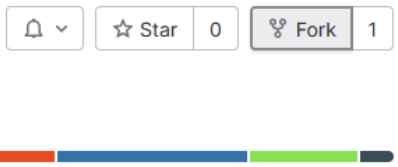
https://dawbio-thebest.scm.azurewebsites.net/

Pots observar que només has d'afegir `scm` a la URL

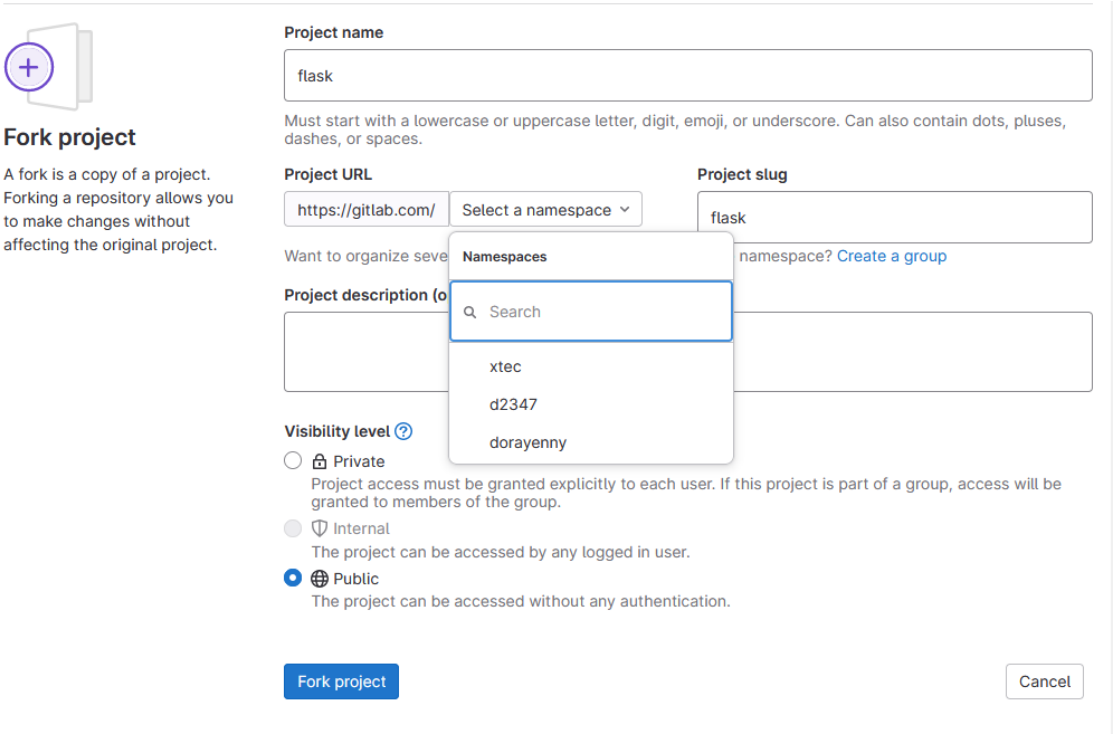
## 06 -Crear una imatge

**1.-** Entra al teu compte de gitlab (si no en en tens, creat un nou compte)

Ves a la pàgina https://gitlab.com/xtec/flask i fes un fork del projecte




**2.-** S'obrirà una nova finestra per indicar com vols clonar el projecte:




**3.-** Arrenca una màquina virtual ubuntu

```
Set-ExecutionPolicy -Scope CurrentUser Unrestricted
curl.exe https://gitlab.com/xtec/box/-/raw/main/box.ps1 -o box.ps1
.\box.ps1 start 1
.\box.ps1 ssh 1
```

**Consell**. Per defecte la màquina només utilitza 1 CPU i 1024 MB de memòria. Pots parar la máquina per incrementar els recursos, i torna a arrencar la màquina.

**4.-** Clona el **teu** projecte:

```
git clone ...
ls
cd ...
```

**5.-** Obre el fitxer `Dockerfile`.

```
FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt --no-cache-dir
COPY . .
```

```
EXPOSE 80
CMD ["gunicorn", "--bind", "0.0.0.0:80", "app:app"]
```

Aquest fitxer té les instruccions que farà servir docker per crear una imatge quan executes la comanda `docker build`.

**Com pots veure en el fitxer:**

- Utilitza com a base una imatge `python:3.8-slim-buster`
- Instal.la les llibreries necessàries amb el programa `pip`.
- Copia el contingut de projecte, excepte el que exclou el fitxer `.dockerignore`
- Configura la comanda d'inici perqué executi gunicorn

**6.-** Obre el fitxer `.gitlab-ci.yml`

```
build:
  image: docker:20.10.10
  services:
    - docker:20.10.10-dind
  #rules:
  #  - if: $CI_PIPELINE_SOURCE == "schedule"
  script:
    - echo $CI_REGISTRY_PASSWORD | docker login -u $CI_REGISTRY_USER
$CI_REGISTRY --password-stdin
    - docker build -t $CI_REGISTRY_IMAGE .
    - docker push $CI_REGISTRY_IMAGE
```

Cada cop que fas un push del teu codi, gitlab segueix les instruccions que es troben en un arxiu que s'ha de nomenar .gitlab.yml.

En aquest cas el fitxer té les instruccions per fer servir docker dins de docker, crear una imatge i pujar la imatge a un registre de gitlab lligat al projecte.

En aquesta adreça pots veure com s'ha executat un d'aquests jobs:
https://gitlab.com/xtec/flask/-/jobs/3724357425

En aquesta adreça estan les imatges del projecte que es generen mitjançant aquesta integració continua: https://gitlab.com/xtec/flask/container_registry

**7.-** Modifica el teu projecte i fes un **push.**

```
git clone https://gitlab.com/.../flask
cd flask
nano templates/index.html
git commit -a -m "He canviat ..."

git push
```

Si el teu compte de gitlab està verificat, es crearà una imatge en el registre del teu projecte.

**8.-** Modifica l'aplicació web perquè faci servir la teva imatge:

```
az webapp config container set --docker-custom-image-name
registry.gitlab.com/<el-teu-usuari>/flask --name dawbio-thebest
--resource-group dawbio
```

**9.** Verifica al portal Azure

Hauries de verificar el teu compte, però també tenim: ...

## 07- Docker Hub

Si no vols verificar el teu compte de gitlab, haurás de crear la imatge i pujar-la a docker hub.

1. Instal.la docker:

```
sudo apt -y update
sudo apt -y install docker.io
```

2. Construeix una imatge amb el tag `xtecd/flask`

```
cd flask
docker build -t dokdoy/flask .
docker images
```

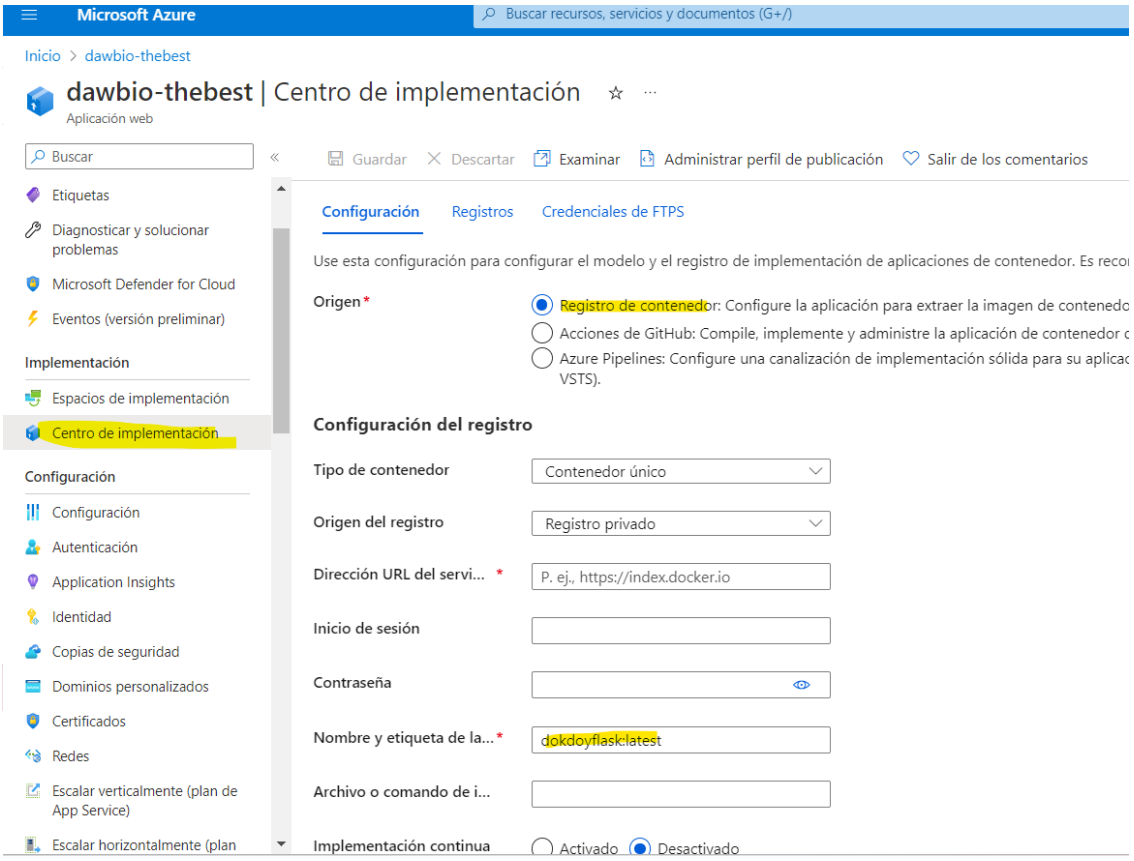
3. Autenticat amb el teu compte de docker hub i puja la imatge a docker hub:

```
docker login
docker push dokdoy/flask
```

4. Modifica l'aplicació web perquè faci servir la imatge de docker hub:

```
az webapp config container set --docker-custom-image-name dokdoy/flask --name
dawbio-thebest --resource-group dawbio
```

5. Verifiqueu la web al portal Azure
6. Verifiqueu el centro d'implementació d'Azure




---

## Azure Container Registry

Enlloc de fer servir el registre de gitlab o el de docker hub pots fer servir directament un registre a Azure.

**1.-** Instal.la el client d'Azure

```
curl -L https://aka.ms/InstallAzureCli | bash
source .bashrc
```

**2.-** Autenticat amb la comanda `az login`:

```
az login
To sign in, use a web browser to open the page
https://microsoft.com/devicelogin and enter the code CW5S2QCAM to
authenticate.
```

**3.-** Crea un registro de contenedor con el comando [az acr create](#) .

```
az acr create -g daw -n registry3443 --sku Basic
```

El nom del registre ha de ser únic a Azure, i contenir caràcteres alfanumèricos de 5 a 50.

En la salida JSON del comando, busque el `loginServer` valor , que es el nombre completo del Registro (todo en minúsculas), que debe incluir el nombre del Registro especificado.

```
 "loginServer": "registry3443.azurecr.io",
```

**4.-** Inicie sesión en el registro mediante el comando az acr login .

```
az acr login -n resistry3443
```

El comando anterior agrega "azurecr.io" al nombre para crear el nombre completo del Registro. Si se ejecuta correctamente, verá el mensaje "Inicio de sesión correcto".

**5.-** Compile la imagen con el comando az acr build .

```
az acr build -r registry3443 -g daw -t flask:latest .
```

Nota:

- El punto (".") al final del comando indica la ubicación del código fuente que se va a compilar. Si no ejecuta este comando en el directorio raíz de la aplicación de ejemplo, especifique la ruta de acceso al código.
- Si deja la `-t` opción (igual `--imageque` ) , el comando pone en cola una compilación de contexto local sin insertarla en el registro. Compilar sin insertar puede ser útil para comprobar que se compila la imagen.

**6.-** Confirme que la imagen de contenedor se creó con el comando az acr repository list .

```
az acr repository list -n registry3443
```

## Configuració de la identitat administrada

Pendent

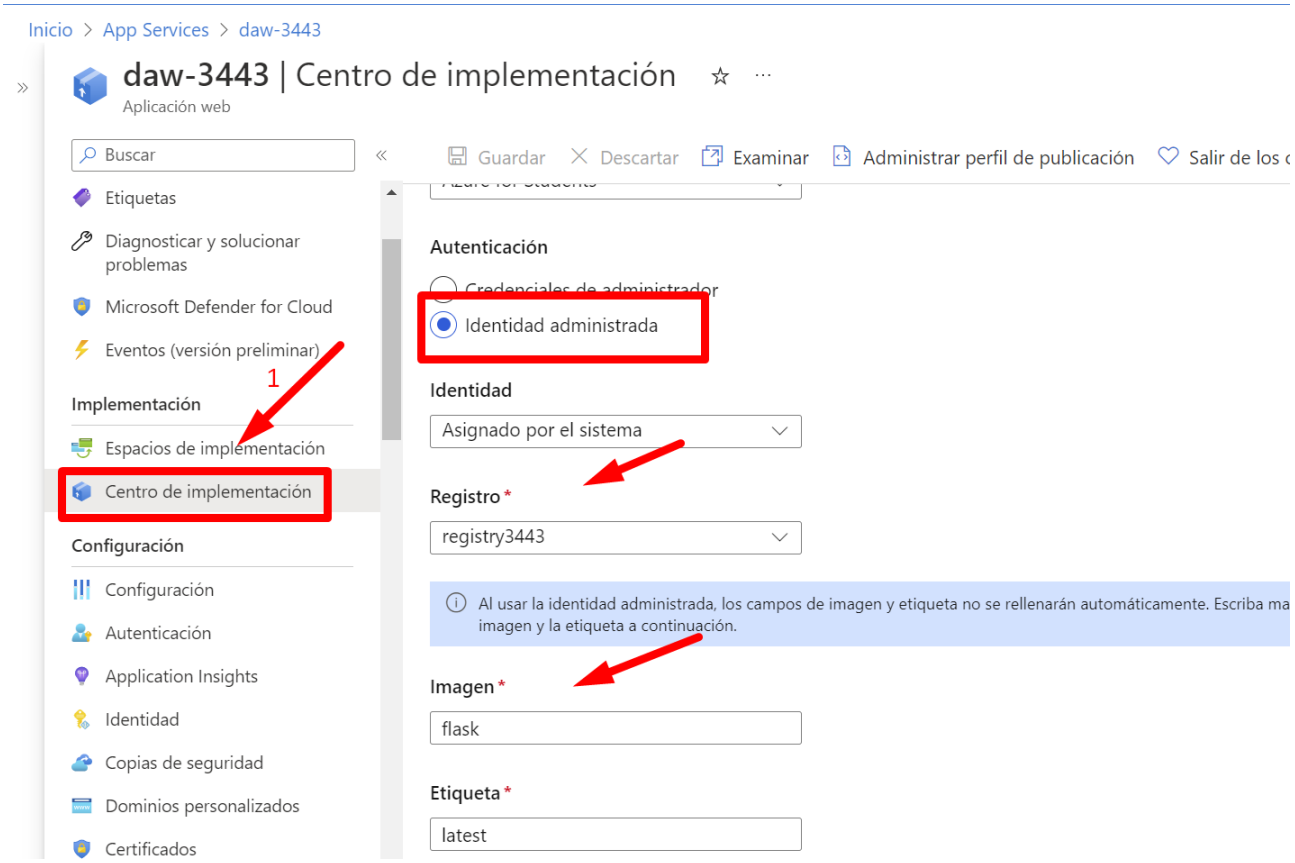
**1.-** Configura l'aplicació web perquè faci servir la imatge del registre d'Azure:

```
az webapp config container set --docker-custom-image-name
registry3443.azurecr.io/flask --name daw-3443 --resource-group daw
```

**2.-** Com que el respositori d'Azure és privat, has de configurar la aplicación web para usar la identidad administrada con el comando [az webapp config set](#) .

```
az webapp config set --resource-group daw --name daw-3443
--generic-configurations '{"acrUseManagedIdentityCreds": true}'
```

Pots veure el resultat al portal d'Azure:

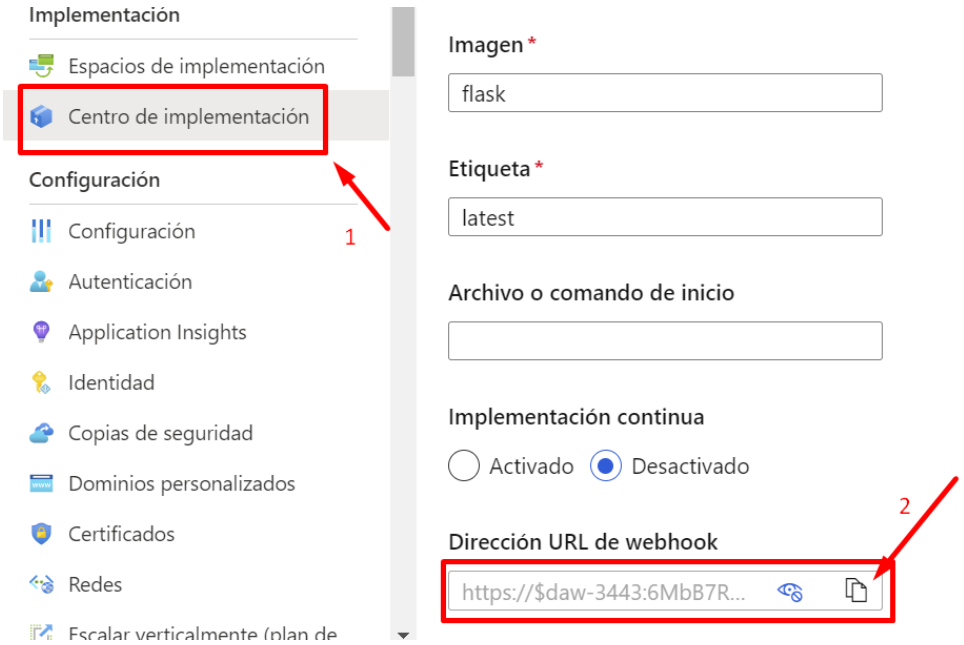



## Webhook

App Service fa servir el registre d'imatges per executar l'aplicació web.

Al arrencar un contenidor descarrega la imatge, però mentres està funcionant no verifica si aquesta imatge s'ha modificat.

Per assabentar a l'aplicació és fa servir un webhook: una url  per notificar que s'ha d'actualitzar el contenidor.




Aquesta url porta incorporat una credencial per autoritzar el pull de la nova imatge i actualitzar l'aplicació web.

Pots obtenir aquesta credencial amb aquesta comanda:

```
az webapp deployment list-publishing-credentials   --resource-group daw
--name daw-3443  --query publishingPassword --output tsv
```

Pendent

## Azure Container Registry

Si fem servir el registre d'Azure, el podem configurar perquè comuniqui a AppService que s'ha fet un push d'una nova imatge a través del webhook.

**1.-** Obtenga la credencial de ámbito de aplicación con el comando az webapp deployment list-publishing-credentials

```
CREDENTIAL=$(az webapp deployment list-publishing-credentials
--resource-group daw  --name daw-3443  --query publishingPassword --output
tsv)
echo $CREDENTIAL
```

**4.-** Use la credencial de ámbito de aplicación para crear un webhook con el comando az acr webhook create .

SERVICE_URI='https://$daw-3443:'$CREDENTIAL'@daw-3443.scm.azurewebsites.net/api/registry/webhook'

az acr webhook create  --actions push --name webhookforflask  --resource-group daw --registry `registry3443`  --uri $SERVICE_URI

### Gitlab

Si volem que un cop gitlab ha generat la imatge, comuniqui a AppService que s'ha fet creat una nova imatge, podem configurar el fitxer `.gitlab-ci.yml` perqué avisi mitjançant el webhook.

## Webgrafia


- Documentación de Azure App Service.
- Hospedaje de una aplicación web con Azure App Service.
- https://learn.microsoft.com/es-es/azure/developer/python/tutorial-containerize-deploy-python-web-app-azure-04?tabs=azure-cli%2Cterminal-bash
- Configuración de Azure App Service
- Configuración de planes de Azure App Service
- https://learn.microsoft.com/es-es/training/modules/host-a-web-app-with-azure-app-service/7-exercise-deploy-your-code-to-app-service?pivots=python
- Implementación y ejecución de una aplicación web en contenedores con Azure App Service - Training | Microsoft Learn
- https://docs.gitlab.com/ee/ci/pipelines/