

Memoria Docker | Informática Como Servicio

Autor - Lucas José Lara García

Opción 2 - Desplegar, mediante docker-compose, un contenedor que contenga dos servicios con las dos imágenes anteriormente mencionadas (wordpress:latest y mysql:5.7). Únicamente existe una url pública. La comunicación entre los servicios se hace de forma interna dentro del contenedor desplegado. Por lo tanto, solamente está expuesto el puerto 80 del servicio wordpress al exterior.

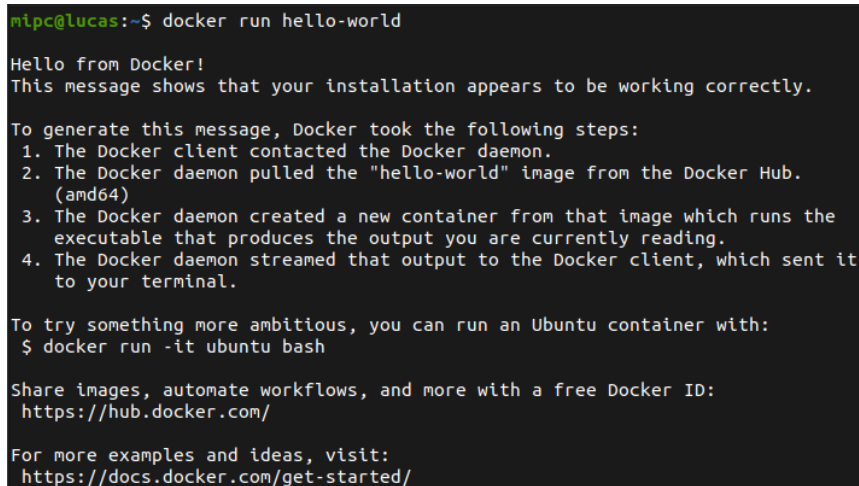
En esta memoria se describe como ha sido el proceso para realizar la práctica de PaaS de Azure paso a paso. Para poder empezar a trabajar se ha configurado el entorno descargando docker en el sistema y se ha habilitado su uso sin *sudo* con las órdenes:

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

Para comprobar que todo funciona correctamente ejecutamos el **hola mundo** de docker

```
docker run hello-world
```

Cuya salida es:



```
mipc@lucas:~$ docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Además, se ha instalado la consola de azure en ubuntu con la orden:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

De igual manera es necesario instalar *Helm*, la dependencia para el despliegue de contenedores, ejecutando el comando:

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3  
| bash
```

Login en azure

En primer lugar se ha creado una cuenta de Azure asociada a la de la udc para poder obtener una serie de servicios gratuitos. A continuación se ha iniciado sesión.

```
az login
```

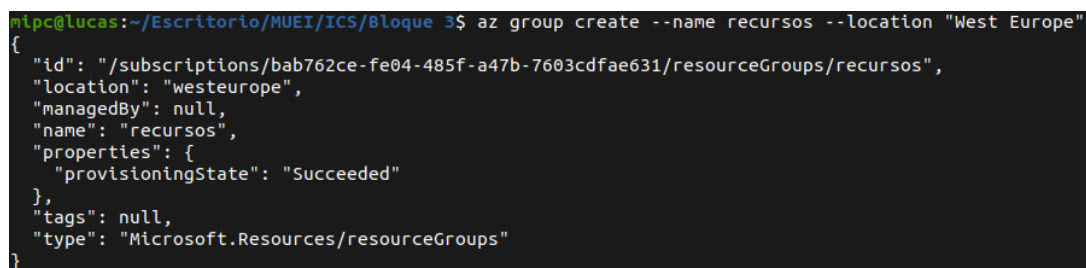


```
mipc@lucas: ~  
mipc@lucas:~$ az login  
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in  
the web browser. If no web browser is available or if the web browser fails to open, use device code flow with `az login --use-devi  
ce-code`.  
{  
  "cloudName": "AzureCloud",  
  "homeTenantId": "defae9d9-59fe-4fa9-b151-12ed7438e099",  
  "id": "bab762ce-fe04-485f-a47b-7603cdfae631",  
  "isDefault": true,  
  "managedByTenants": [],  
  "name": "Azure for Students",  
  "state": "Enabled",  
  "tenantId": "defae9d9-59fe-4fa9-b151-12ed7438e099",  
  "user": {  
    "name": "thehaias@outlook.es",  
    "type": "user"  
  }  
}
```

Creación del Container Registry

Seguidamente se ha creado un nuevo recurso *"recursos"*. Este recurso es el registro de imágenes Docker de Azure:

```
az group create --name recursos --location "West Europe"
```



```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ az group create --name recursos --location "West Europe"  
{  
  "id": "/subscriptions/bab762ce-fe04-485f-a47b-7603cdfae631/resourceGroups/recursos",  
  "location": "westeurope",  
  "managedBy": null,  
  "name": "recursos",  
  "properties": {  
    "provisioningState": "Succeeded"  
  },  
  "tags": null,  
  "type": "Microsoft.Resources/resourceGroups"  
}
```

Y a continuación se ha creado el *Container Registry "serviciolucasics"* sobre el grupo de recursos creado en el paso anterior:

```
az acr create --name serviciolucasics --resource-group recursos --sku Basic
```

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ az acr create --name serviciolucasics --resource-group recursos --sku Basic
{
  "adminUserEnabled": false,
  "anonymousPullEnabled": false,
  "creationDate": "2022-11-15T13:37:11.845869+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/bab762ce-fe04-485f-a47b-7603cdfae631/resourceGroups/recursos/providers/Microsoft.ContainerRegistry/r
egistries/serviciolucasics",
  "identity": null,
  "location": "westeurope",
  "loginServer": "serviciolucasics.azurecr.io",
  "name": "serviciolucasics",
  "networkRuleBypassOptions": "AzureServices",
  "networkRuleSet": null,
  "policies": {
    "azureAdAuthenticationAsArmPolicy": {
      "status": "enabled"
    },
    "exportPolicy": {
      "status": "enabled"
    },
    "quarantinePolicy": {
      "status": "disabled"
    },
    "retentionPolicy": {
      "days": 7,
      "lastUpdatedTime": "2022-11-15T13:37:21.850030+00:00",
      "status": "disabled"
    },
    "softDeletePolicy": {
      "lastUpdatedTime": "2022-11-15T13:37:21.850030+00:00",
      "retentionDays": 7,
      "status": "disabled"
    },
    "trustPolicy": {
      "status": "disabled",
      "type": "Notary"
    }
  },
  "privateEndpointConnections": [],
  "provisioningState": "Succeeded",
  "publicNetworkAccess": "Enabled",
  "resourceGroup": "recursos",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "status": null,

```

Push de la imagen al Container registry de Azure

En primer lugar, hacemos login en nuestro plan de servicios con la orden:

```
az acr login --name serviciolucasics
```

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ az acr login --name serviciolucasics
Login Succeeded
```

En segundo lugar, hacemos pull de las imágenes con las versiones correspondientes que se nos requerían en el enunciado:

```
docker image pull wordpress:latest
docker image pull mysql:5.7
```

El resultado es el siguiente:

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker image pull wordpress:latest
latest: Pulling from library/wordpress
a603fa5e3b41: Pull complete
c428f1a49423: Pull complete
156740b07ef8: Pull complete
fb5a4c8af82f: Pull complete
25f85b498fd5: Pull complete
9b233e420ac7: Pull complete
fe42347c4ecf: Pull complete
d14eb2ed1e17: Pull complete
66d98f73acb6: Pull complete
d2c43c5efbc8: Pull complete
ab590b48ea47: Pull complete
80692ae2d067: Pull complete
05e465aaa99a: Pull complete
5e1d260f5864: Pull complete
1f1b92fc1af4: Pull complete
ef98b4d7e310: Pull complete
b8532900ff77: Pull complete
34541aa160dd: Pull complete
7ce8d947d989: Pull complete
e4a678b65168: Pull complete
a06af02714d7: Pull complete
Digest: sha256:d6b628b9f3caf5910c9e7284e9cbb86b46f1495680cfc806fa98d73c90fa1c6
Status: Downloaded newer image for wordpress:latest
docker.io/library/wordpress:latest
```

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker image pull mysql:5.7
5.7: Pulling from library/mysql
9a0b9cd2dfe6: Pull complete
c637408ee7df: Pull complete
4c517093c276: Pull complete
301cc7d68c2a: Pull complete
17ca9bf9231a: Pull complete
9ae101e5c786: Pull complete
04baa409344e: Pull complete
f0b6015bf853: Pull complete
6005bb052ef8: Pull complete
99f303d57050: Pull complete
307a9a80c1df: Pull complete
Digest: sha256:0e3435e72c493aec752d8274379b1eac4d634f47a7781a7a92b8636fa1dc94c1
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
```

Después, para hacer el push de las imágenes hay que crear un **tag** y además debe tener un formato específico, en concreto:

```
<nombreRegistro>.azurecr.io/<nombre>:<version>
```

A continuación creamos el tag para las imágenes previamente creadas.

```
docker tag wordpress:latest serviciolucasics.azurecr.io/wordpress:latest
docker tag mysql:5.7 serviciolucasics.azurecr.io/mysql:5.7
```

Y seguidamente para comprobarlo listamos las imágenes Docker del sistema:

```
docker images
```

El resultado es el siguiente:

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker tag wordpress:latest serviciolucasics.azurecr.io/wordpress:latest
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker tag mysql:5.7 serviciolucasics.azurecr.io/mysql:5.7
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker images
```

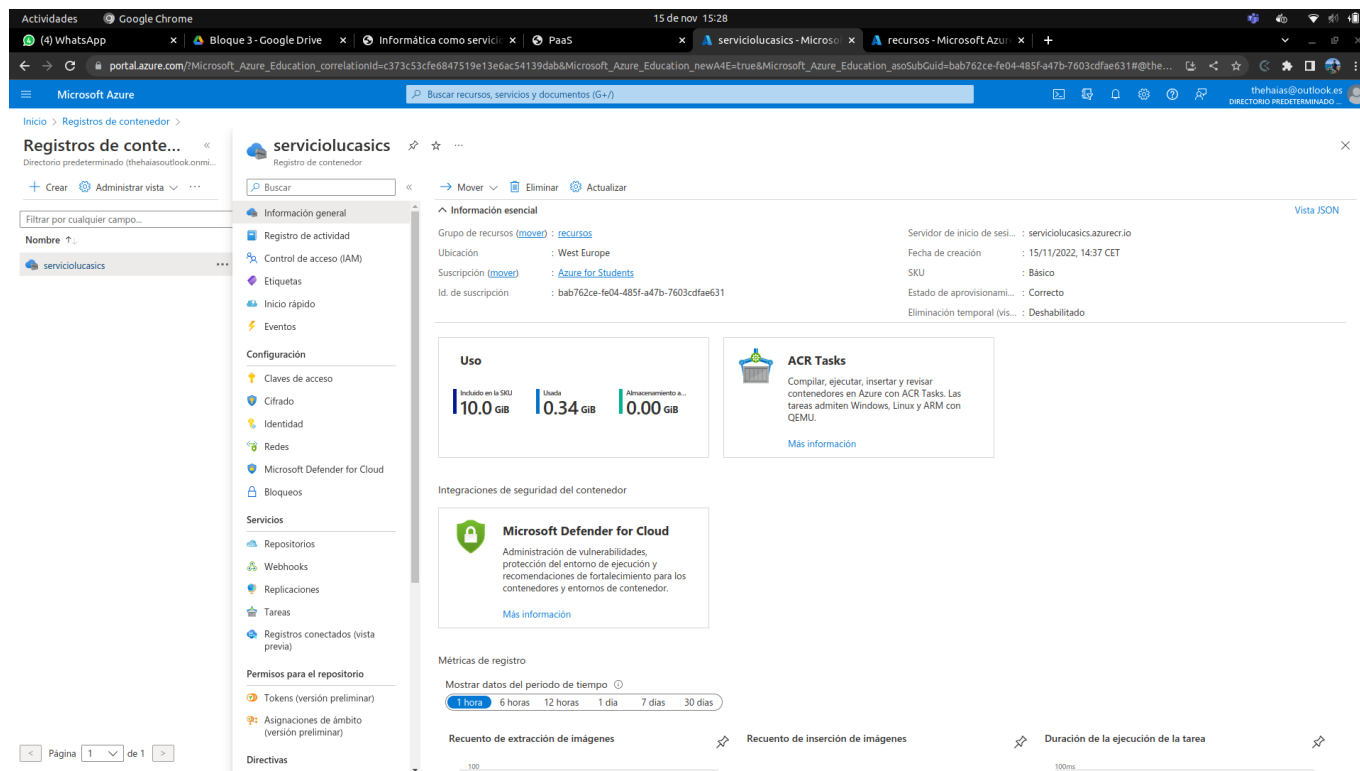
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
wordpress	latest	4473a9685e0b	4 hours ago	612MB
serviciolucasics.azurecr.io/wordpress	latest	4473a9685e0b	4 hours ago	612MB
bulletinboard	1.0	0c2fe10d24ce	4 days ago	688MB
planserviciolucasics.azurecr.io/bulletinboard	v1	0c2fe10d24ce	4 days ago	688MB
serviciolucasics.azurecr.io/mysql	5.7	eef0fab001e8	10 days ago	495MB
mysql	5.7	eef0fab001e8	10 days ago	495MB
hello-world	latest	feb5d9fea6a5	13 months ago	13.3kB

Seguidamente, al tratarse de un repositorio privado, hay que habilitar el login ejecutando el comando:

```
az acr update -n serviciolucasics --admin-enabled true
```

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ az acr update -n serviciolucasics --admin-enabled true
{
  "adminUserEnabled": true,
  "anonymousPullEnabled": false,
  "creationDate": "2022-11-15T13:37:11.845869+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/bab762ce-fe04-485f-a47b-7603cdfae631/resourceGroups/recursos/providers/Microsoft.ContainerRegistry/r
egistries/serviciolucasics",
  "identity": null,
  "location": "westeurope",
  "loginServer": "serviciolucasics.azurecr.io",
  "name": "serviciolucasics",
  "networkRuleBypassOptions": "AzureServices",
  "networkRuleSet": null,
  "policies": {
    "azureAdAuthenticationAsArmPolicy": {
      "status": "enabled"
    },
    "exportPolicy": {
      "status": "enabled"
    },
    "quarantinePolicy": {
      "status": "disabled"
    },
    "retentionPolicy": {
      "days": 7,
      "lastUpdatedTime": "2022-11-15T13:37:21.850030+00:00",
      "status": "disabled"
    },
    "softDeletePolicy": {
      "lastUpdatedTime": "2022-11-15T13:37:21.850030+00:00",
      "retentionDays": 7,
      "status": "disabled"
    },
    "trustPolicy": {
      "status": "disabled",
      "type": "Notary"
    }
  },
  "privateEndpointConnections": [],
  "provisioningState": "Succeeded",
  "publicNetworkAccess": "Enabled",
  "resourceGroup": "recursos",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "status": null,
  "systemData": {
    "createdAt": "2022-11-15T13:37:11.845869+00:00"
```

Y si accedemos al portal de azure a la sección *registro de contenedor* se puede ver el contenedor creado:



Posteriormente hacemos login por consola con el siguiente comando:

```
docker login serviciolucasics.azurecr.io
```

El resultado se muestra en la siguiente imagen:

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker login serviciolucasics.azurecr.io
Username (00000000-0000-0000-0000-000000000000): serviciolucasics
Password:
WARNING! Your password will be stored unencrypted in /home/mipc/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

Finalmente ya podemos hacer el push de las imagenes Docker para su despliegue en Azure con el siguiente comando:

```
docker push serviciolucasics.azurecr.io/wordpress:latest
docker push serviciolucasics.azurecr.io/mysql:5.7
```

El resultado se muestra en la siguiente imagen:

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker push serviciolucasics.azurecr.io/wordpress:latest
The push refers to repository [serviciolucasics.azurecr.io/wordpress]
69d070ebb713: Pushed
4b3329c6d445: Pushed
df4502d16efc: Pushed
4ed0a84639ff: Pushed
12e5af3e723e: Pushed
f1e08374f9f8: Pushed
038757fef684: Pushed
982159df618a: Pushed
3d33242bf117: Pushed
529016396883: Pushed
5464bcc3f1c2: Pushed
28192e867e79: Pushed
d173e78df32e: Pushed
0be1ec4fbfdc: Pushed
30fa0c430434: Pushed
a538c5a6e4e0: Pushed
e5d40f64dcb4: Pushed
44148371c697: Pushed
797a7c0590e0: Pushed
f60117696410: Pushed
ec4a38999118: Pushed
latest: digest: sha256:ae83387028e987b6937ee6c201b1ae1b3da8d7e4dcf0392e0ec96678ed310a2c size: 4710
```

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ docker push serviciolucasics.azurecr.io/mysql:5.7
The push refers to repository [serviciolucasics.azurecr.io/mysql]
a5e6b19bb0ea: Pushed
4fe9b42ec433: Pushed
d5ca9f973f82: Pushed
2e59a86f1a56: Pushed
463c1e5f5bdf: Pushed
0a30c1d12b5d: Pushed
2ff54c49223b: Pushed
af0fb2439f4c: Pushed
e186b8fba002: Pushed
ac4f520e60d7: Pushed
ba7464c70f4e: Pushed
5.7: digest: sha256:55d2f4aa17fd27821a7fc575d2921485681a8aa5ac8411b75d7a163d895dfba1 size: 2618
```

Docker compose

El siguiente paso es crear el archivo **docker-compose.yml** con las imágenes creadas y donde se indican los puertos y algunas credenciales como el usuario y contraseña de wordpress y de la base de datos. Tiene el siguiente contenido:

```
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 1234
      MYSQL_DATABASE: wordpress
      MYSQL_USER: lucas
      MYSQL_PASSWORD: 1234

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
```

```
WORDPRESS_DB_USER: lucas
WORDPRESS_DB_PASSWORD: 1234
volumes:
  db_data:
```

Creación del app service y container

Con la siguiente orden creamos el app service asociado al grupo de recursos creado previamente:

```
az appservice plan create --name weblucas \
--resource-group recursos \
--sku B1 \
--is-linux
```

Este es el resultado de la ejecución:

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ az appservice plan create --name weblucas \
--resource-group recursos \
--sku B1 \
--is-linux
Resource provider 'Microsoft.Web' used by this operation is not registered. We are registering for you.
Registration succeeded.
{
  "elasticScaleEnabled": false,
  "extendedLocation": null,
  "freeOfferExpirationTime": "2023-05-15T16:56:12.936666",
  "geoRegion": "West Europe",
  "hostingEnvironmentProfile": null,
  "hyperv": false,
  "id": "/subscriptions/bab762ce-fe04-485f-a47b-7603cdfae631/resourceGroups/recursos/providers/Microsoft.Web/serverfarms/web
lucas",
  "isSpot": false,
  "isXenon": false,
  "kind": "linux",
  "kubeEnvironmentProfile": null,
  "location": "westeurope",
  "maximumElasticWorkerCount": 1,
  "maximumNumberOfWorkers": 0,
  "name": "weblucas",
  "numberOfSites": 0,
  "numberOfWorkers": 1,
  "perSiteScaling": false,
  "provisioningState": "Succeeded",
  "reserved": true,
  "resourceGroup": "recursos",
  "sku": {
    "capabilities": null,
    "capacity": 1,
    "family": "B",
    "locations": null,
    "name": "B1",
    "size": "B1",
    "skuCapacity": null,
    "tier": "Basic"
  },
  "spotExpirationTime": null,
  "status": "Ready",
  "subscription": "bab762ce-fe04-485f-a47b-7603cdfae631",
  "tags": null,
  "targetWorkerCount": 0,
  "targetWorkerSizeId": 0,
  "type": "Microsoft.Web/serverfarms",
  "workerTierName": null,
  "zoneRedundant": false
}
```

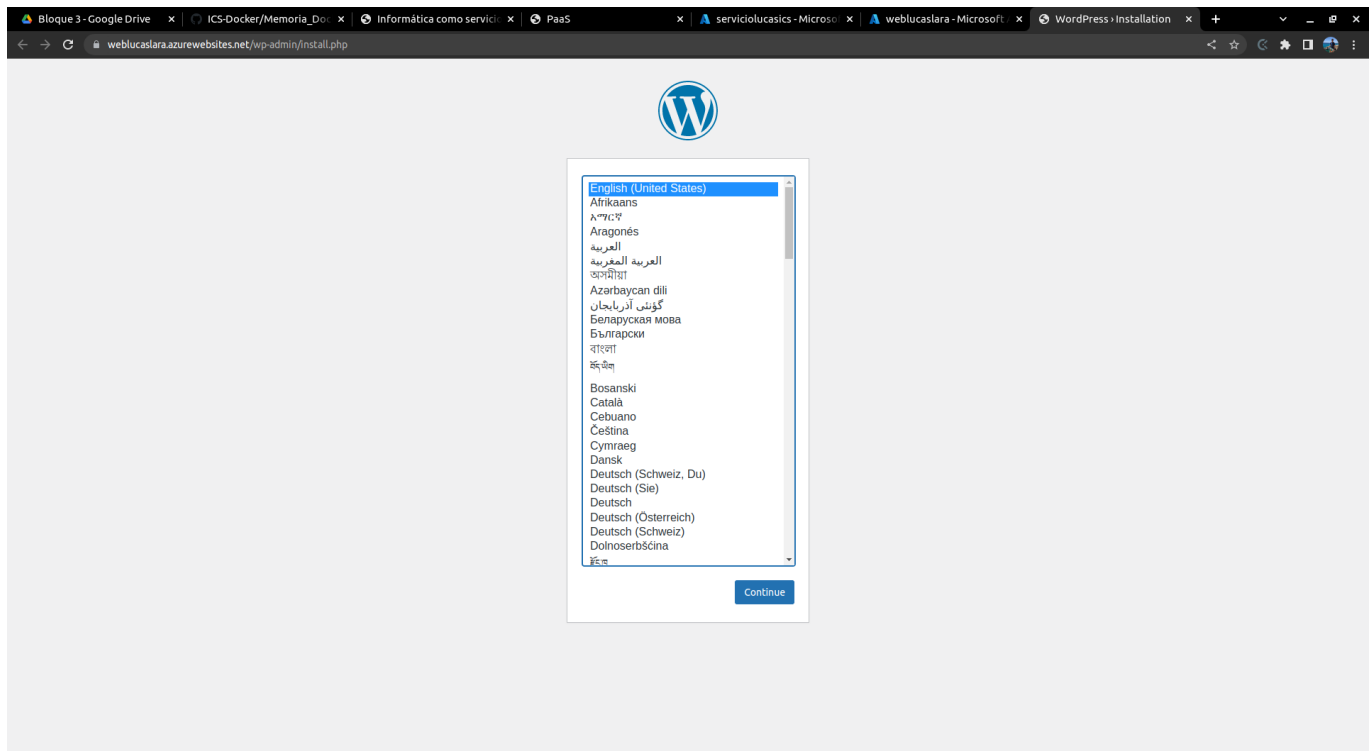
Y seguidamente creamos el contenedor que contiene dos servicios con las dos imágenes anteriormente creadas:


```
az webapp create -n weblucaslara -p weblucas -g recursos --multicontainer-
config-file docker-compose.yml --multicontainer-config-type compose
```

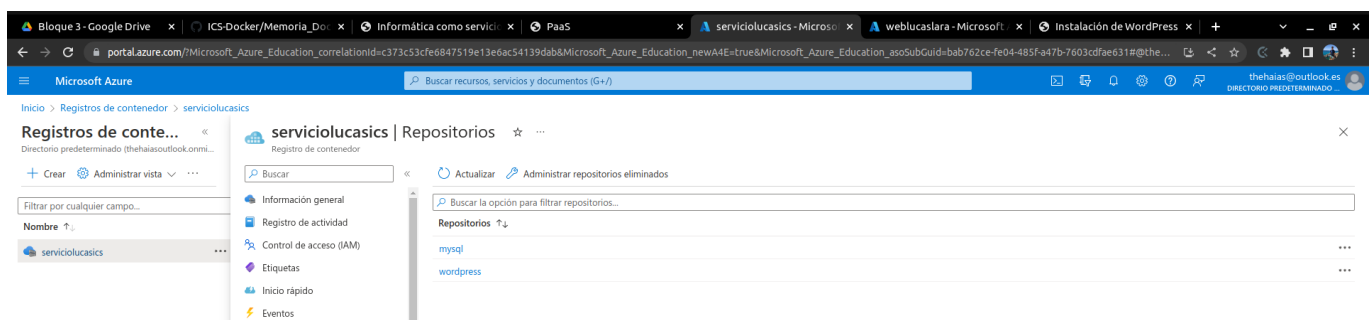
El resultado se muestra a continuación:

```
mipc@lucas:~/Escritorio/MUEI/ICS/Bloque 3$ az webapp create -n weblucaslara -p weblucas -g recursos --multicontainer-config-
file docker-compose.yml --multicontainer-config-type compose
{
  "availabilityState": "Normal",
  "clientAffinityEnabled": true,
  "clientCertEnabled": false,
  "clientCertExclusionPaths": null,
  "clientCertMode": "Required",
  "cloningInfo": null,
  "containerSize": 0,
  "customDomainVerificationId": "92D8D95E3E8DB3974A105AE087F0E507559AFB13DE3385656475E4C4C7F923FB",
  "dailyMemoryTimeQuota": 0,
  "defaultHostName": "weblucaslara.azurewebsites.net",
  "enabled": true,
  "enabledHostNames": [
    "weblucaslara.azurewebsites.net",
    "weblucaslara.scm.azurewebsites.net"
  ],
  "extendedLocation": null,
  "ftpPublishingUrl": "ftp://waws-prod-am2-597.ftp.azurewebsites.windows.net/site/wwwroot",
  "hostNameSslStates": [
    {
      "hostType": "Standard",
      "ipBasedSslResult": null,
      "ipBasedSslState": "NotConfigured",
      "name": "weblucaslara.azurewebsites.net",
      "sslState": "Disabled",
      "thumbprint": null,
      "toUpdate": null,
      "toUpdateIpBasedSsl": null,
      "virtualIp": null
    },
    {
      "hostType": "Repository",
      "ipBasedSslResult": null,
      "ipBasedSslState": "NotConfigured",
      "name": "weblucaslara.scm.azurewebsites.net",
      "sslState": "Disabled",
      "thumbprint": null,
      "toUpdate": null,
      "toUpdateIpBasedSsl": null,
      "virtualIp": null
    }
  ],
  "hostNames": [
    "weblucaslara.azurewebsites.net"
  ],
  "hostNamesDisabled": false,
  "hostingEnvironmentProfile": null,
  "httpsOnly": false,
  "hyperV": false,
  "id": "/subscriptions/bab762ce-fe04-485f-a47b-7603cdfae631/resourceGroups/recursos/providers/Microsoft.Web/sites/weblucasl
ara",
  "identity": null,
  "inProgressOperationId": null,
  "isDefaultContainer": null
}
```

Y como vemos se muestra en el portal de azure en el apartado *App Services*:



Dentro del portal de azure, si accedemos a *Registros de contenedor* > *Repositorios* podemos ver las imágenes de nuestro servicio:



Y para finalizar, una vez completado el proceso de inicio en wordpress ya podemos acceder a nuestra página y añadir aquello que queramos, en mi caso un titular y una nueva entrada, como se muestra en la imagen a continuación:

