Fedora Containers Lab: Episodio II

Alex Callejas
Services Content Architect @ Red Hat

@dark_axl
darkaxl017.fedorapeople.org/slides/
rutil.io/social

¿Y la primera parte?

Slides:

- Fedora México meetup v1.0
- FLISoL 2019
- Fedora Containers Lab v2
- Containers Lab - Red Hat
- SG Virtual 2020
- TecNM Campus Iztapalapa III 2021

Videos:

- Fedora México meetup
- SG Virtual 2020

FliSol 2021

¿Porque una segunda parte?

Podman

Podman Pod → Kubernetes

Creamos el contenedor y lo validamos

```
# podman run -dt -p 8000:80 --name demo quay.io/libpod/alpine_nginx:latest
```

```
# curl localhost:8000
podman rulez
```

Generamos un snapshot para crear el archivo YAML de kubernetes

```
# podman generate kube demo > demo.yml
```

Eliminamos el pod y comprobamos el YAML creado

```
# podman play kube demo.yml
```

```
...
a container exists with the same name ("demo") as the pod in your YAML
file; changing pod name to demo_pod
...
```

Major Hayden @ Devconf.cz 🇨🇿
@majorhayden

Ready to take your container to Kubernetes? Use:

podman generate kube

to create Kubernetes yaml once you have everything working the way you want! #devconfcz

okd

docs.okd.io/minishift/getting-started

FliSol 2021

Podman

Podman Pod → Kubernetes

Editamos el YAML y lo validamos

```
# vi demo.yml
```

```
...
  labels:
    app: demo
  name: demo-pod
...
    image: quay.io/libpod/alpine_nginx:latest
    name: demo-container
...
```

```
# podman play kube demo.yml
```

```
# podman pod list
```

```
# curl localhost:8000
podman rulez
```

Podman

Podman Pod → Kubernetes

Descargamos el YAML y lo cargamos en minishift

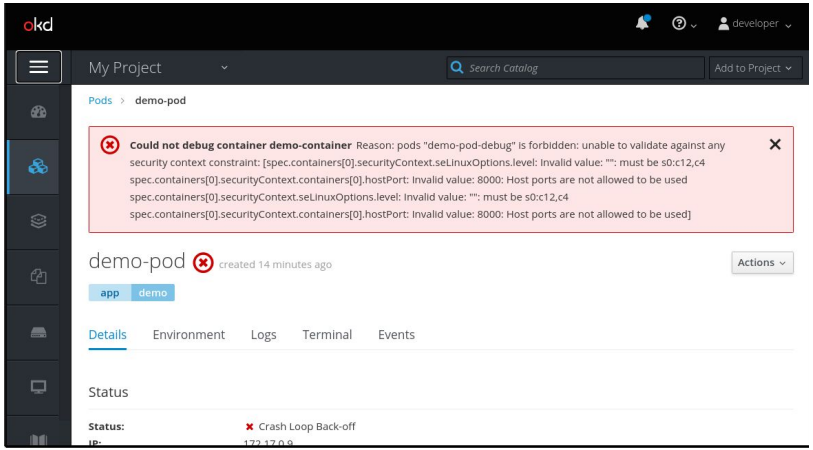
```
$ oc create -f demo.yml -n myproject
```

Checamos el status

```
$ oc status
```

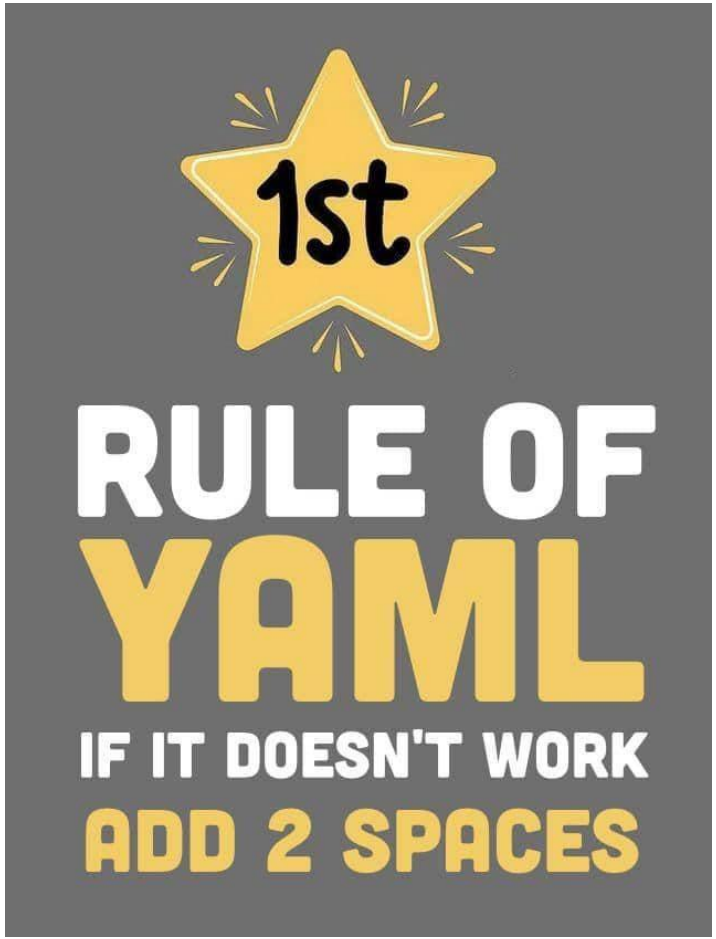
```
...
Errors:
  * pod/demo-pod is crash-looping
...
```

Checamos el status en la consola web →



FliSol 2021

Errores en YAML?



FliSol 2021

Podman

Podman Pod → Kubernetes

Corregidos los errores, lo cargamos y exponemos el servicio

```
$ oc create -f demo.yml -n myproject
```

```
$ oc get pods
NAME       READY     STATUS    RESTARTS   AGE
demo-pod   1/1       Running   0          54s
```

```
$ oc expose pod demo-pod
service/demo-pod exposed
```

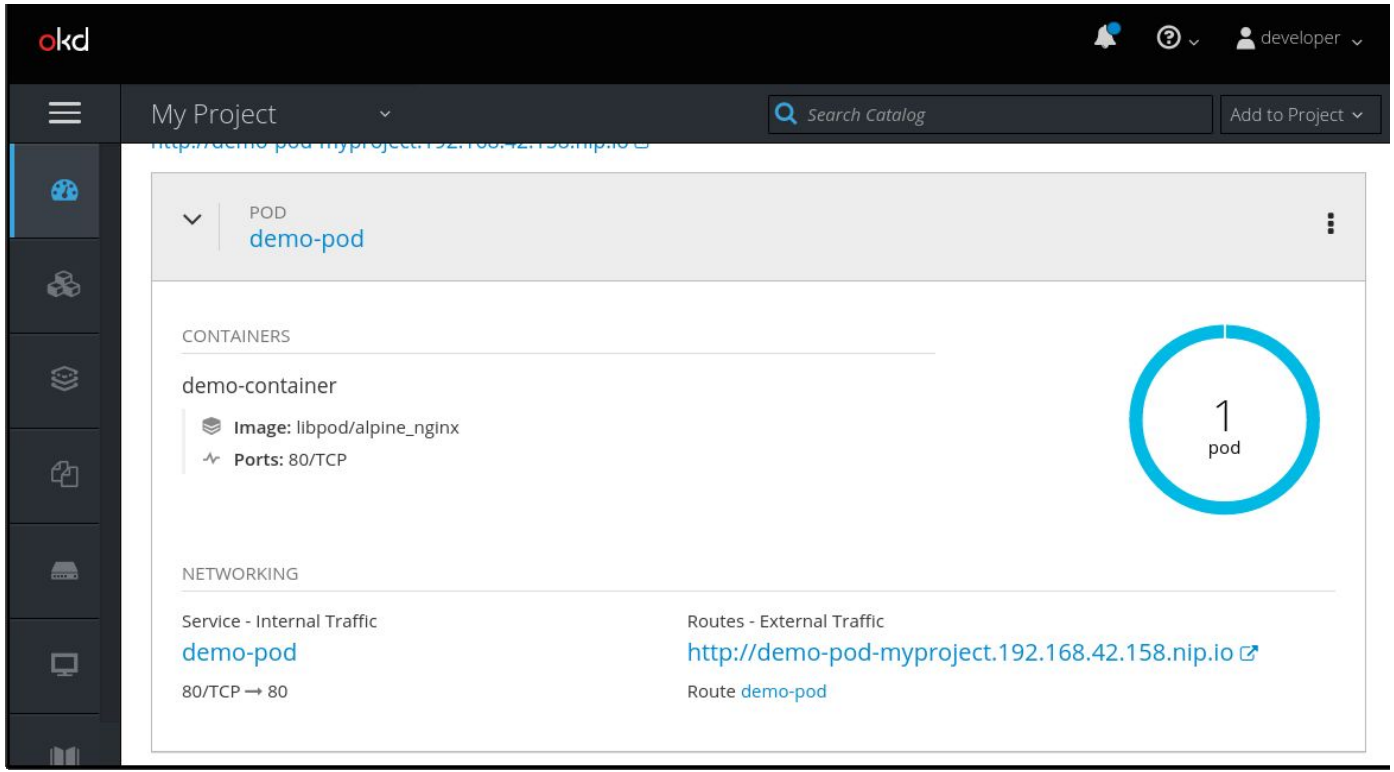
```
$ oc get svc
NAME       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
demo-pod   ClusterIP   172.30.140.79   <none>        80/TCP     6s
```

```
$ oc port-forward demo-pod 8000:80
Forwarding from 127.0.0.1:8000 -> 80
Forwarding from [::1]:8000 -> 80
Handling connection for 8000
```

Podman

Podman Pod → Kubernetes

Creamos la ruta en la consola web



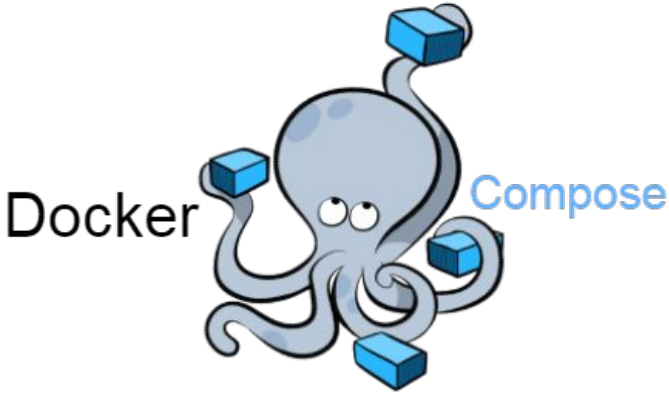
FliSol 2021

Podman Compose

Podman Compose es un proyecto cuyo objetivo es ser utilizado como una alternativa a **Docker Compose** sin necesidad de realizar ningún cambio en el archivo `docker-compose.yaml`.

> *Un **Pod** es un grupo de uno o más contenedores, con recursos de almacenamiento y red compartidos, además una especificación de cómo ejecutar dichos contenedores.*
>
> **- PODS - KUBERNETES DOCUMENTATION**

La idea básica detrás de **Podman Compose** es que recoge los servicios definidos dentro del archivo `docker-compose.yaml` y crea un contenedor *rootless* para cada servicio. **Podman Compose** añade los *rootless* contenedores a un único pod para todo el proyecto, y todos los *rootless* contenedores comparten la misma red.



Docker Compose logo

Instalación

```
# dnf install podman-compose
```

FliSol 2021

Podman Compose

/home/student/awx3/docker-compose.yml

```
version: '3'
services:
  postgres:
    image: "postgres:9.6"
    volumes:
      - pgsql:/var/lib/postgresql/data
    environment:
      POSTGRES_USER: awx
      POSTGRES_PASSWORD: awxpass
      POSTGRES_DB: awx

  rabbitmq:
    image: "rabbitmq:3"
    volumes:
      - rabbitmq:/var/lib/rabbitmq
    environment:
      RABBITMQ_DEFAULT_VHOST: awx

  memcached:
    image: "memcached:alpine"
```

```
  awx_web:
    # image: "geerlingguy/awx_web:latest"
    image: "ansible/awx_web:3.0.1"
    volumes:
      - nginxweb:/var/lib/nginx
    links:
      - rabbitmq
      - memcached
      - postgres
    ports:
      - "8080:8052"
    hostname: awxweb
    user: root
    environment:
      SECRET_KEY: aabbcc
      DATABASE_USER: awx
      DATABASE_PASSWORD: awxpass
      DATABASE_NAME: awx
      DATABASE_PORT: 5432
      DATABASE_HOST: postgres
      RABBITMQ_USER: guest
      RABBITMQ_PASSWORD: guest
      RABBITMQ_HOST: rabbitmq
      RABBITMQ_PORT: 5672
      RABBITMQ_VHOST: awx
      MEMCACHED_HOST: memcached
      MEMCACHED_PORT: 11211
```

```
  awx_task:
    # image: "geerlingguy/awx_task:latest"
    image: "ansible/awx_task:3.0.1"
    volumes:
      - nginxtask:/var/lib/nginx
    links:
      - rabbitmq
      - memcached
      - awx_web:awxweb
      - postgres
    hostname: awx
    user: root
    environment:
      SECRET_KEY: aabbcc
      DATABASE_USER: awx
      DATABASE_PASSWORD: awxpass
      DATABASE_NAME: awx
      DATABASE_PORT: 5432
      DATABASE_HOST: postgres
      RABBITMQ_USER: guest
      RABBITMQ_PASSWORD: guest
      RABBITMQ_HOST: rabbitmq
      RABBITMQ_PORT: 5672
      RABBITMQ_VHOST: awx
      MEMCACHED_HOST: memcached
      MEMCACHED_PORT: 11211
```

Podman Compose

Desde el directorio donde se encuentra el archivo `docker-compose.yaml`, ejecutamos

```
$ podman-compose up -d
```

← se crea el pod con el nombre del directorio

Checamos el status

```
$ podman-compose ps
```

```
$ podman pod list
```

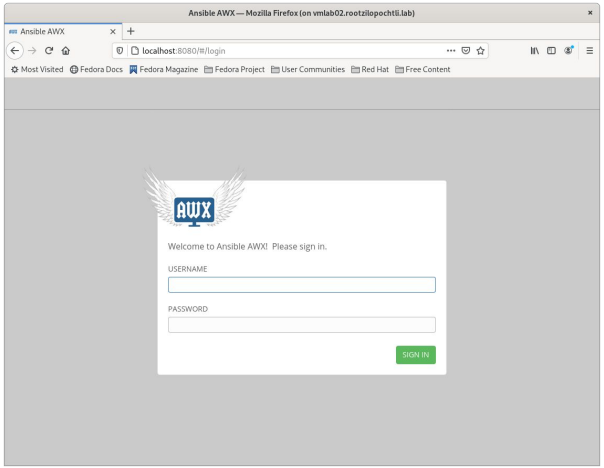
```
$ podman volume ls
```

```
$ podman logs <container_name>
```

Validamos la aplicación

```
$ curl localhost:8080
```

```
$ firefox &
```



Modificamos la contraseña de **admin**

```
$ podman exec -ti awx3_awx_web_1 /bin/bash
```

```
# awx-manage changepassword admin
```

Podman Compose

Para detener el servicio, ejecutamos

```
$ podman-compose down
```

Checamos el status

```
$ podman-compose ps
```

```
$ podman pod list
```

Eliminamos el servicio

```
$ podman pod prune
```

```
$ podman volume prune
```

FliSol 2021

Podman Compose

Docker Compose → Podman Pod → Kubernetes

/home/student/blog/docker-compose.yml

```
version: "3.8"
services:
  web:
    image: wordpress
    restart: always
    volumes:
      - wordpress:/var/www/html
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: magazine
      WORDPRESS_DB_NAME: magazine
      WORDPRESS_DB_PASSWORD:
1maGazine!
      WORDPRESS_TABLE_PREFIX: cz
      WORDPRESS_DEBUG: 0
    depends_on:
      - db
    networks:
      - wpnet
```

```
  db:
    image: mariadb:10.5
    restart: always
    ports:
      - 6603:3306

    volumes:
      - wpdbvol:/var/lib/mysql

    environment:
      MYSQL_DATABASE: magazine
      MYSQL_USER: magazine
      MYSQL_PASSWORD: 1maGazine!
      MYSQL_ROOT_PASSWORD: 1maGazine!
    networks:
      - wpnet
volumes:
  wordpress: {}
  wpdbvol: {}

networks:
  wpnet: {}
```

FliSol 2021

Podman Compose

Docker Compose → Podman Pod → Kubernetes

Desde el directorio donde se encuentra el archivo `docker-compose.yaml`, ejecutamos

```
$ podman-compose up -d
```

Checamos el status

```
$ podman-compose ps
```

```
$ podman pod list
```

```
$ podman volume ls
```

```
$ podman logs <container_name>
```

Validamos la aplicación

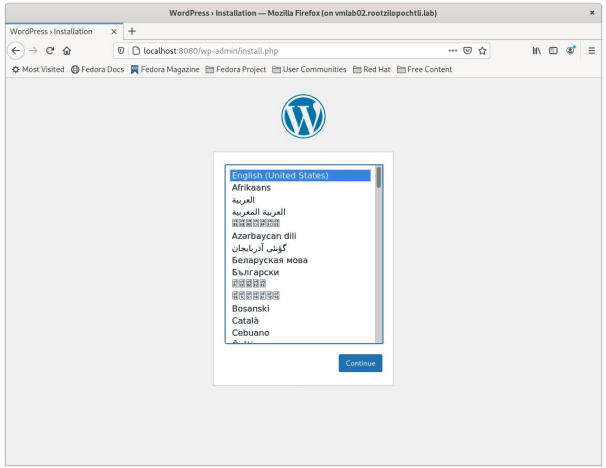
```
$ curl localhost:8080
```

```
$ firefox &
```

Generamos un snapshot para crear el archivo YAML y lo comprobamos

```
$ podman generate kube -s blog > blog.yaml
```

```
$ podman play kube blog.yaml
```



fedora

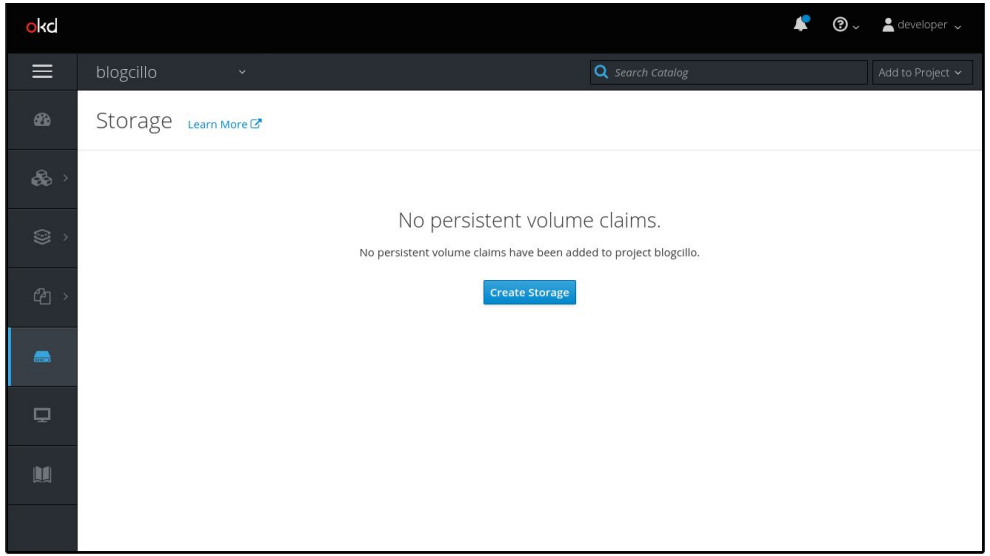
Podman Compose

Docker Compose → Podman Pod → Kubernetes

Descargamos el YAML y lo cargamos en minishift para checar errores

```
$ oc create -f blog.yaml -n blogcillo
```

Creamos los volúmenes en la consola web →




```
$ grep claimName blog.yaml
      claimName: blog-wordpress
      claimName: blog-wpdbvol
```

Nota: No tenemos especificación del tamaño, por lo que los crearemos con el mínimo (**2 GiB**).

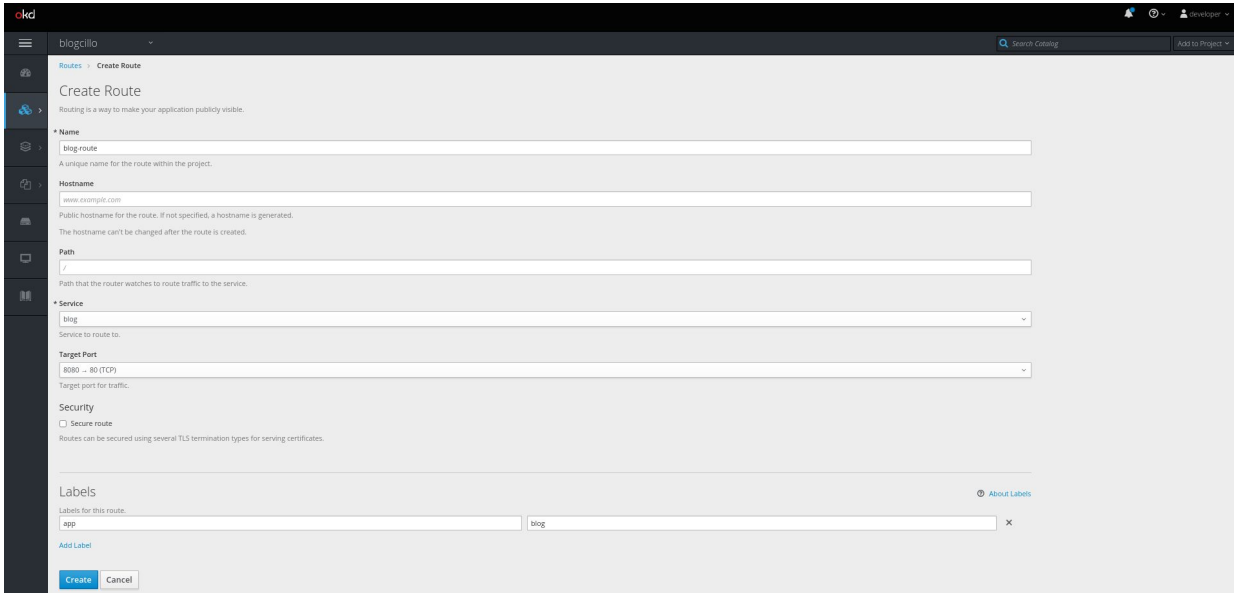
Podman Compose

Docker Compose → Podman Pod → Kubernetes

Corregidos los errores, lo cargamos y exponemos el servicio

```
$ oc create -f blog.yaml -n blogcillo
```

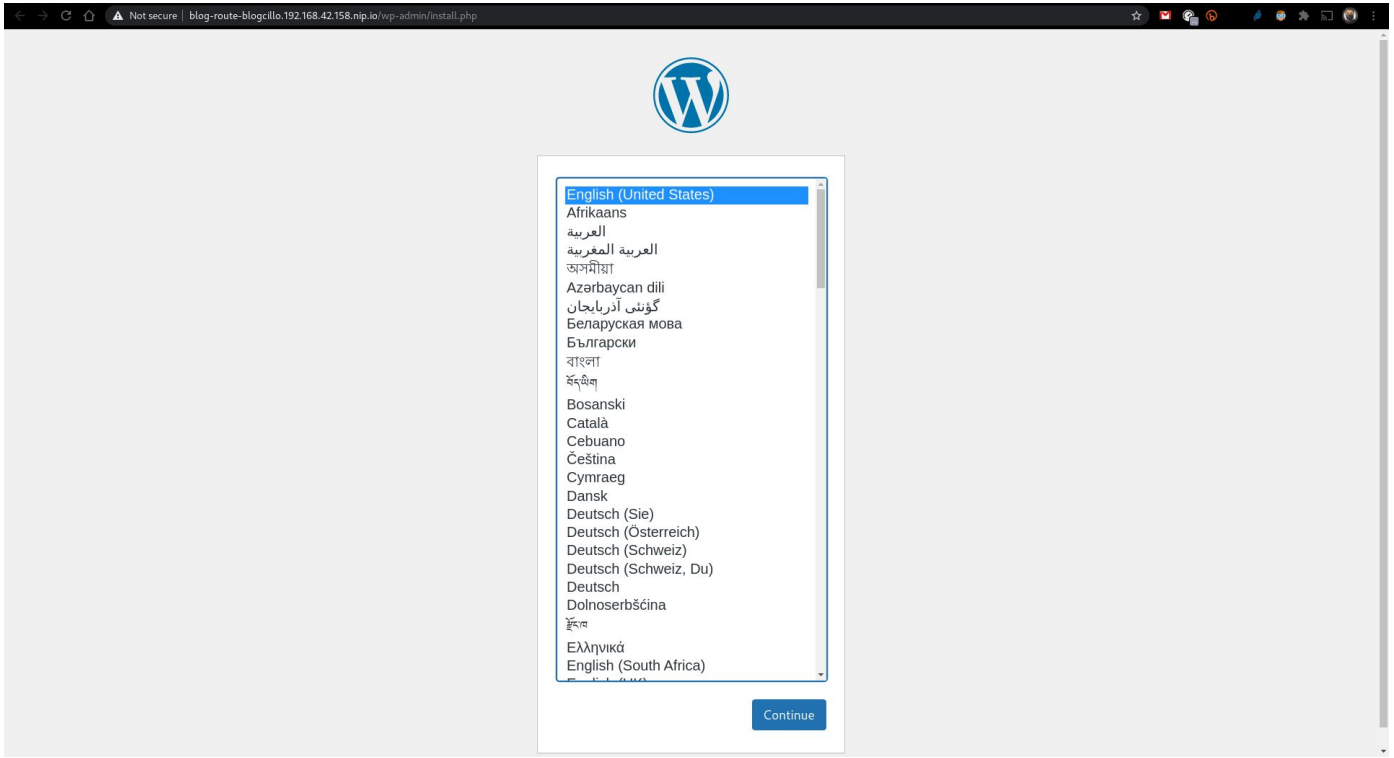
Creamos la ruta en la consola web →



Podman Compose

Docker Compose → Podman Pod → Kubernetes

Validamos el servicio



Podman

Podman Compose → Podman Pod → Kubernetes

Tips

- La generación de YAML's de Kubernetes aún está en desarrollo, por lo que:
  - Muchos de los *values* y *options* aún no están liberados o no están soportadas en *k8s*
  - Las variables de los archivos **Compose** apuntan al hostname, hay que dirigirlos al *container_name*
  - En algunos casos se requiere modificar el contexto de SELinux (Si, ya se...)
  - Leer, modificar y probar

FliSol 2021

Podman Secrets

Podman v3.0

Permite utilizar fácilmente el contenido *sensible* dentro de un contenedor, pero evita que termine en algún lugar fuera del contenedor, como en un registro de imágenes.

Creamos un archivo con la información *sensible*

```
$ echo "secretdata" > secretfile
```

Creamos el secreto y lo inspeccionamos

```
$ podman secret create secretname secretfile
```

```
$ podman secret inspect secretname
```

FliSol 2021

Podman Secrets

Podman v3.0

Ejecutamos un contenedor con *secrets*

```
$ podman run --secret secretname --name foo alpine cat /run/secrets/secretname
```

Creamos una imagen a partir del contenedor con *secrets*

```
$ podman commit foo secrimg
```

Ejecutamos el contenedor para validar el *secret*

```
$ podman run secrimg cat /run/secrets/secretname
cat: can't open '/run/secrets/secretname': No such file or directory
```

FliSol 2021

Podman in Podman

Running a container inside a container

Nota:

- Por el momento sólo está disponible para **RHEL 8.3**
- Se requieren todos los privilegios por lo que sólo al iniciar el primer contenedor como **root** es posible acceder a esta función

Ejecutamos el contenedor de forma interactiva y privilegiada

```
# podman run --privileged -it registry.redhat.io/rhel8/podman /bin/bash
```

Dentro del contenedor, creamos un directorio y nos cambiamos a él

```
[root@dbec61e1f9fc /]# mkdir container && cd container/
```

Creamos el **`Containerfile`**

```
[root@dbec61e1f9fc container]# vi Containerfile
```

```
FROM registry.access.redhat.com/ubi8/ubi
RUN yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
RUN yum -y install moon-buggy && yum clean all
CMD ["/usr/bin/moon-buggy"]
```

FliSol 2021

Podman in Podman

Running a container inside a container

Construimos la imagen

```
[root@dbec61e1f9fc container]# podman build -t moon-buggy .
```

Verificamos la creación de la imagen

```
[root@dbec61e1f9fc container]# podman images
```

Ejecutamos un contenedor basado en la imagen y lo verificamos

```
[root@dbec61e1f9fc container]# podman run -it --name moon moon-buggy
```

Esta imagen la podríamos etiquetar para enviarla a un repositorio remoto

```
[root@dbec61e1f9fc container]# podman tag moon-buggy registry.example.com/moon-buggy
```

```
[root@dbec61e1f9fc container]# podman push registry.example.com/moon-buggy
```

FliSol 2021

Referencias

Links y Documentación

- Manage containers with Podman Compose
- From Docker Compose to Kubernetes with Podman
- Using Podman and Docker Compose
- Adapting Docker and Kubernetes containers to run on Red Hat OpenShift Container Platform
- Technology preview: Running a container inside a container

- Fedora Containers Lab Examples

- registry.fedoraproject.org
- kubernetesbyexample.com

- podman.io
- github.com/containers/podman-compose

- Daniel Walsh - @rhatdan

https://hopin.com/events/fedora-linux-34-release-party

Gracias!

Únete a la conversación!

@fedoramexico