

Creando y Provisionando Containers Docker

Preparar el entorno de trabajo

Verificar que docker está funcionando:

```
docker run hello-world
```

Creando una imagen Docker con Puppet

Construiremos una imagen Docker basada en la imagen utilizada en laboratorios anteriores (ubuntu) y le agregaremos la herramienta de provisionamiento Puppet.

Creamos un directorio y dentro de él un archivo llamado Dockerfile con el siguiente contenido:

```
FROM ubuntu
RUN apt-get -y update
RUN apt-get -y install ruby
RUN echo "gem: --no-ri --no-rdoc" > ~/.gemrc RUN gem install
puppet librarian-puppet
```

Ahora ejecutamos dentro del directorio el comando:

```
docker build -t="usuarioDocker/puppetbase" .
```

Listamos las imágenes y debería aparecer la que recién creamos

```
docker images
```

Ejecutemos un shell dentro de un container que usa la imagen

```
docker run -it arellanog/puppetbase bash
```

Para salir de la consola Presionar CTRL+p y luego CTRL+q

Para ver el container en ejecución

```
docker ps
```

Provisionado una imagen Docker usando Puppet

Construiremos una imagen Docker basada en la anterior que mediante puppet instalará nginx.

Creamos una carpeta y dentro, el siguiente Dockerfile:

```
FROM usuarioDocker/puppetbase
RUN apt-get -y -q install wget git-core
ADD Puppetfile /
RUN librarian-puppet install
RUN puppet apply --modulepath=/modules -e "class { 'nginx': }"
RUN echo "daemon off;" >> /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx"]
```

Creamos un manifiesto para Puppet llamado Puppetfile con el siguiente contenido:

```
mod "nginx",
    :git => "https://github.com/jfryman/puppet-nginx"
```

Creamos la imagen con:

```
docker build -t="usuarioDocker/nginx" .
```

Listamos las imágenes y debería aparecer la que recién creamos

```
docker images
```

Ejecutamos un container basado en la imagen recién creada

```
docker run -P -d usuarioDocker/nginx
```