Using gitRepo To Mount a Git Repository

In this lesson, we will explore the gitRepo Volume type.

WE'LL COVER THE FOLLOWING

- The gitRepo Volume Type
  - Looking into the Definition
  - Creating the Pod
  - Destroying the Pod

The gitRepo Volume Type

The `gitRepo` Volume type is probably not going to be on your list of top three Volume types. Or, maybe it will. It all depends on your use cases. We like it since it demonstrates how a concept of a Volume can be extended to a new and innovative solution.

Looking into the Definition

Let's see it in action through the `volume/github.yml` definition.

```
cat volume/github.yml
```

The **output** is as follows.

```
apiVersion: v1
kind: Pod
metadata:
  name: github
spec:
  containers:
  - name: github
    image: docker:17.11
    command: ["sleep"]
    args: ["100000"]
    volumeMounts:
    - mountPath: /var/run/docker.sock
```

```
      name: docker-socket
    - mountPath: /src
      name: github
  volumes:
  - name: docker-socket
    hostPath:
      path: /var/run/docker.sock
      type: Socket
  - name: github
    gitRepo:
      repository: https://github.com/vfarcic/go-demo-2.git
      directory: .
```

This Pod definition is very similar to `volume/docker.yml`. The only significant difference is that we added the second `volumeMount`. It will mount the directory `/src` inside the container, and will use the Volume named `github`. The Volume definition is straightforward. The `gitRepo` type defines the Git `repository` and the `directory`. If we skipped the latter, we'd get the repository mounted as `/src/go-demo-2`.

> The `gitRepo` Volume type allows a third field which we haven't used. We could have set a specific `revision` of the repository. But, for demo purposes, the `HEAD` should do.

Creating the Pod

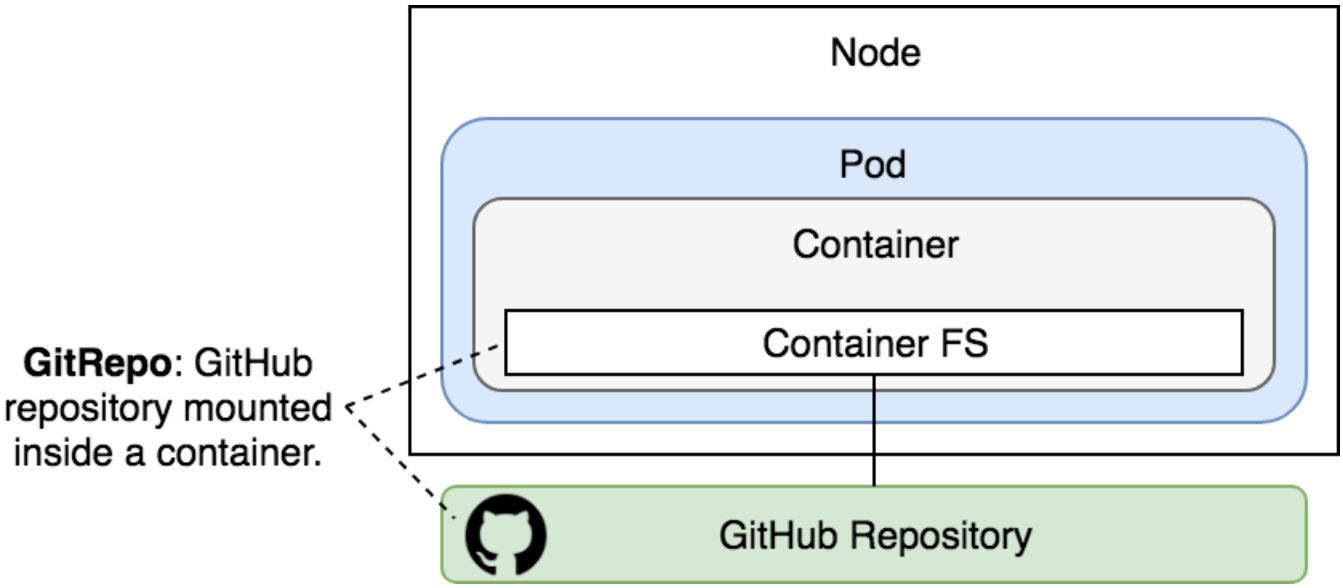
Let's create the Pod.

```
kubectl create \
    -f volume/github.yml
```

Now that we created the Pod, we'll enter its only container, and check whether `gitRepo` indeed works as expected.

```
kubectl exec -it github sh
cd /src
ls -l
```

We entered into the container of the Pod, switched to the `/src` directory, and listed all the files and directories inside it. That proved that `gitRepo` mounted a Volume with the contents of the `vfarcic/go-demo-2` GitHub repository.



GitHub repository mounted inside a container

Since the Pod container is based on the `docker` image, and the socket is mounted as well, we should be able to build the image using the source code provided by the `gitRepo` Volume.

```
docker image build \
    -t vfarcic/go-demo-2:beta .
```

This time, the build should be very fast since we already have the same image on the host, and the source code did not change in the meantime. You should see a `Using cache` notification for each layer of the image we're building.

Destroying the Pod

Since we now proved the point, let's get out of the container and remove the Pod.

```
exit

kubectl delete \
    -f volume/github.yml
```

`gitRepo` is a nifty little addition to the Volume types. It does not save us a lot of work, nor does it provide something truly exceptional. We could accomplish the same result by using an image with `git` and execute a simple `git clone` command.

Still, the Volume type might come in handy on a few occasions. The more we have defined in YAML files, the less we depend on ad-hoc commands. That way, we can aim towards fully documented processes.

> The `gitRepo` Volume type helps us move `git` commands (e.g., `git clone`) into the YAML definition. It also removes the need for the `git` binary inside containers. While `gitRepo` might not always be the best option, it is indeed something worth considering.

This was it about the gitRepo Volume type.

In the next lesson, we will discuss the non-persisting state of a Deployment.