Init Containers

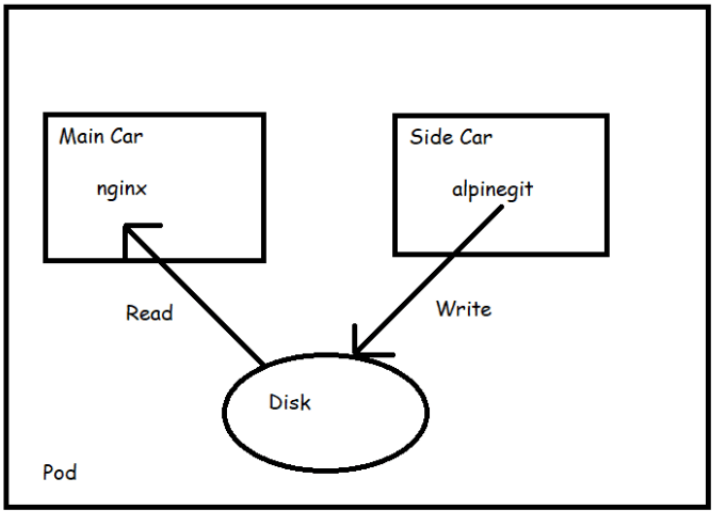
- Overview: Init Containers enable separation of concerns by providing a separate lifecycle for initialization related tasks distinct from main application containers.
- Init Containers in k8s are part of Pod definition, and they separate all containers in the Pods into two groups
    - Init Containers
    - Application Containers
- All init container are executed in sequence & all of them have to terminate successfully before the application containers are started up.

```
---
apiVersion: v1
kind: Pod
metadata:
  name: initcontainer
  labels:
    app: ngnix
spec:
  initContainers:
    - name: download
      image: alpine
      command:
        - git
        - clone
        - https://github.com/qt/frontend.git
        - /var/www/html
      volumeMounts:
        - mountPath: /var/www/html
          name: source
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: /var/www/html
          name: source
  volumes:
    - emptyDir: {}
      name: source
```

Sidecar

- A Sidecar container extends and enhances the functionality of a pre-existing container without changing it.

```
---
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  containers:
    - name: app
      image: nginx
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: /var/www/html
          name: git
    - name: polling
      image: alpinegit
      volumeMounts:
        - mountPath: /var/www/html
          name: git
      env:
        - name: GIT_REPO
          value: https://github.com/qt/frontend.git
      command:
        - "sh"
        - "-c"
        - "git clone $(GIT_REPO) . && watch -n 600 git pull"
  volumes:
    - emptyDir: {}
      name: git
```



Now we can use side car

- to export logs
- synchronize directories

Adapter: Converting from one form to another form.

- Adapter pattern takes a heterogenous containerized system and makes it confirm to consistent unified interface with standard and normalized format to be consumed by outside world.
- Adapter pattern inherits all the characteristics from Sidecar, but has the single purpose of providing adapted access to the application
- We have an application which is composed of multiple microservices written in different languages
- So, we need to monitor the application i.e. we want applications to expose the metrics in the same format to the monitoring tool.