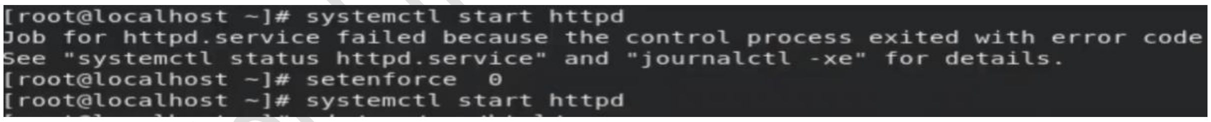
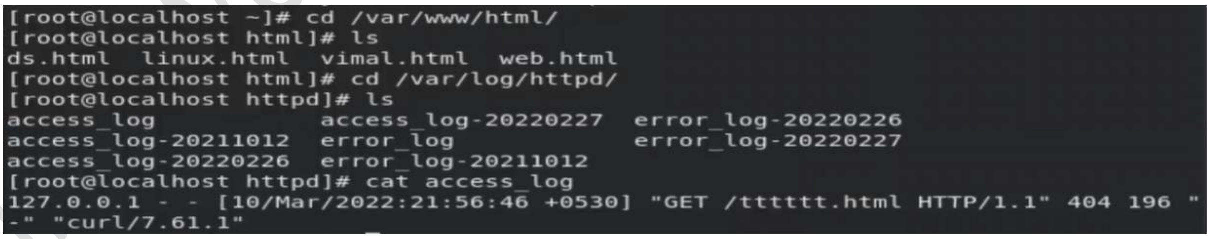
Summary

- Design patterns for multi-container pods are
    - Sidecar pattern:- In the sidecar pattern pod, the container that runs our app, and the container that runs the agent runs parallel
    - Adaptor pattern:-In the adaptor pattern the agent can extract transform & load data
    - Ambassador pattern:- In the ambassador pattern there are two containers, one works as a server & other works as a proxy container.
- Logs management tools are used for centralizing, analyzing & visualization of log data
- One of the duties of the agent is to collect recode & send recode in real-time to centralized storage
- A good practice is running a single server on one system
- Multi-container pod usually has a server & agent program
- Practical;- Logs in Apache webserver
    - Running a web server in a local system



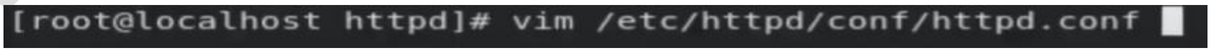
```
[root@localhost ~]# systemctl start httpd
Job for httpd.service failed because the control process exited with error code
See "systemctl status httpd.service" and "journalctl -xe" for details.
[root@localhost ~]# setenforce  0
[root@localhost ~]# systemctl start httpd
```

    - Accessing logs in the webserver

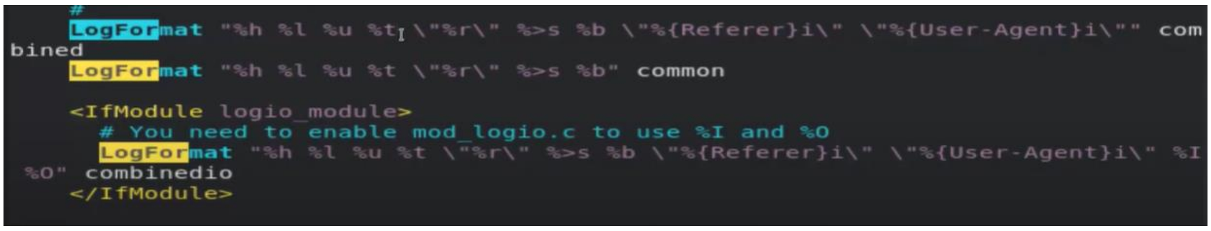


```
[root@localhost ~]# cd /var/www/html/
[root@localhost html]# ls
ds.html  linux.html  vimal.html  web.html
[root@localhost html]# cd /var/log/httpd/
[root@localhost httpd]# ls
access_log            access_log-20220227  error_log-20220226
access_log-20211012   error_log            error_log-20220227
access_log-20220226   error_log-20211012
[root@localhost httpd]# cat access_log
127.0.0.1 - - [10/Mar/2022:21:56:46 +0530] "GET /tttttt.html HTTP/1.1" 404 196 "
-" "curl/7.61.1"
```

    - File to change log formats



```
[root@localhost httpd]# vim /etc/httpd/conf/httpd.conf
```



```
    #
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" com
bined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
      # You need to enable mod_logio.c to use %I and %O
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I
 %O" combinedio
    </IfModule>
```

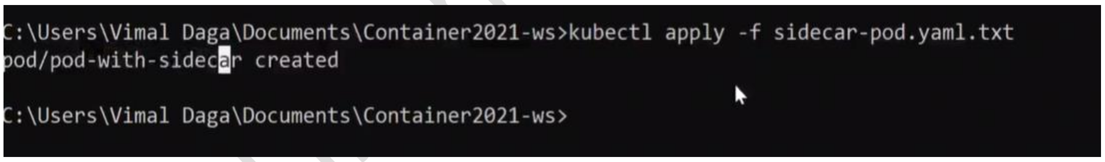
        - Here every character has a different meaning for example %h means host name, %t means time stamp

- **Side Car pattern example**
    - Manifest file for sidecar pattern

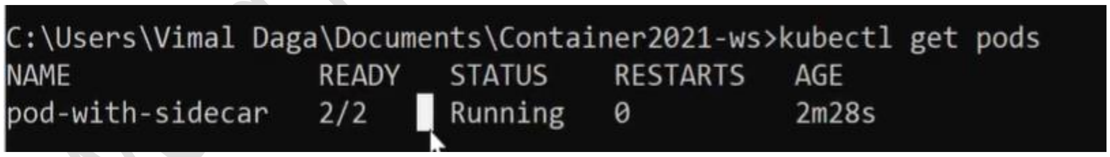
```
# Example YAML configuration for the sidecar pattern.

# It defines a main application container which writes
# the current date to a log file every five seconds.

# The sidecar container is nginx serving that log file.
# (In practice, your sidecar is likely to be a log collection
# container that uploads to external storage.)

# To run:
#   kubectl apply -f pod.yaml

# Once the pod is running:
#
#   (Connect to the sidecar pod)
#   kubectl exec pod-with-sidecar -c sidecar-container -it bash
#
#   (Install curl on the sidecar)
#   apt-get update && apt-get install curl
#
#   (Access the log file via the sidecar)
#   curl 'http://localhost:80/app.txt'

apiVersion: v1
kind: Pod
metadata:
  name: pod-with-sidecar
spec:
  # Create a volume called 'shared-logs' that the
  # app and sidecar share.
  volumes:
  - name: shared-logs
    emptyDir: {}

  # In the sidecar pattern, there is a main application
  # container and a sidecar container.
  containers:
```

```
# Main application container
- name: app-container
  # Simple application: write the current date
  # to the log file every five seconds
  image: alpine # alpine is a simple Linux OS image
  command: ["/bin/sh"]
  args: ["-c", "while true; do date >> /var/log/app.txt; sleep 5;done"]

  # Mount the pod's shared log file into the app
  # container. The app writes logs here.
  volumeMounts:
  - name: shared-logs
    mountPath: /var/log

# Sidecar container
- name: sidecar-container
  # Simple sidecar: display log files using nginx.
  # In reality, this sidecar would be a custom image
  # that uploads logs to a third-party or storage service.
  image: nginx:1.7.9
  ports:
    - containerPort: 80

  # Mount the pod's shared log file into the sidecar
  # container. In this case, nginx will serve the files
  # in this directory.
  volumeMounts:
  - name: shared-logs
    mountPath: /usr/share/nginx/html # nginx-specific mount path
```

- Creating a pod from a manifest file

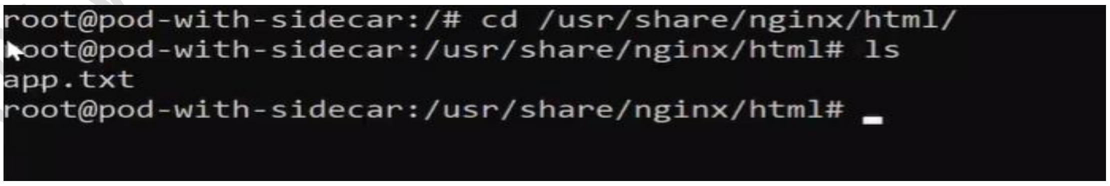


```
:\Users\Vimal Daga\Documents\Container2021-ws>kubectl apply -f sidecar-pod.yaml.txt
pod/pod-with-sidecar created

:\Users\Vimal Daga\Documents\Container2021-ws>
```



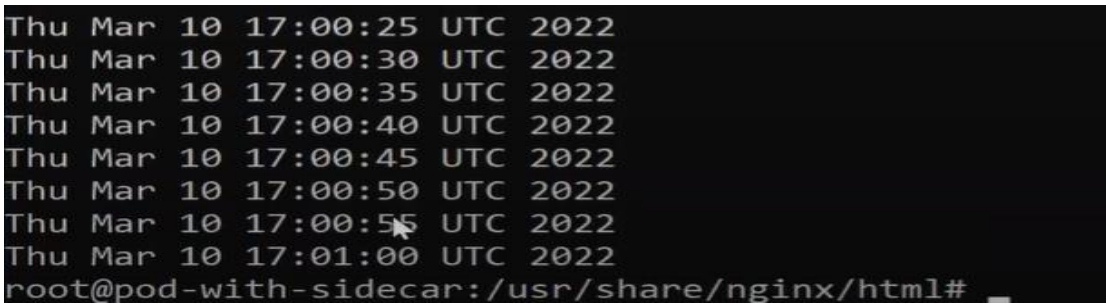
```
C:\Users\Vimal Daga\Documents\Container2021-ws>kubectl get pods
NAME               READY   STATUS    RESTARTS   AGE
pod-with-sidecar   2/2     Running   0          2m28s
```

- Accessing log file



```
root@pod-with-sidecar:/# cd /usr/share/nginx/html/
root@pod-with-sidecar:/usr/share/nginx/html# ls
app.txt
root@pod-with-sidecar:/usr/share/nginx/html# _
```

- Checking logs in a log file



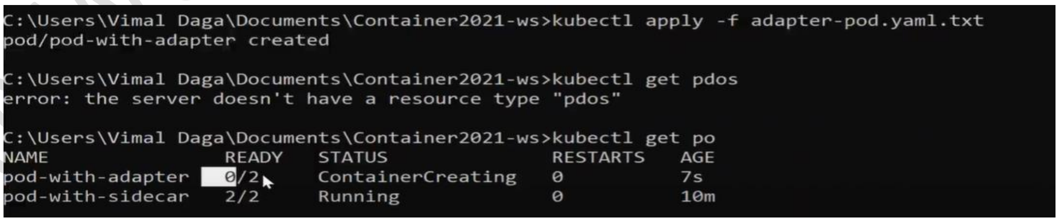
```
Thu Mar 10 17:00:25 UTC 2022
Thu Mar 10 17:00:30 UTC 2022
Thu Mar 10 17:00:35 UTC 2022
Thu Mar 10 17:00:40 UTC 2022
Thu Mar 10 17:00:45 UTC 2022
Thu Mar 10 17:00:50 UTC 2022
Thu Mar 10 17:00:55 UTC 2022
Thu Mar 10 17:01:00 UTC 2022
root@pod-with-sidecar:/usr/share/nginx/html# _
```

- Adaptor pattern example
    - Manifest file for adaptor pattern pod

```
# Example YAML configuration for the adapter pattern.

# It defines a main application container which writes
# the current date and system usage information to a log file
# every five seconds.

# The adapter container reads what the application has written and
# reformats it into a structure that a hypothetical monitoring
# service requires.

# To run:
#   kubectl apply -f pod.yaml

# Once the pod is running:
#
#   (Connect to the application pod)
#   kubectl exec pod-with-adapter -c app-container -it sh
#
#   (Take a look at what the application is writing.)
#   cat /var/log/top.txt
#
#   (Take a look at what the adapter has reformatted it to.)
#   cat /var/log/status.txt


apiVersion: v1
kind: Pod
metadata:
  name: pod-with-adapter
spec:
  # Create a volume called 'shared-logs' that the
  # app and adapter share.
  volumes:
  - name: shared-logs
    emptyDir: {}

  containers:
```

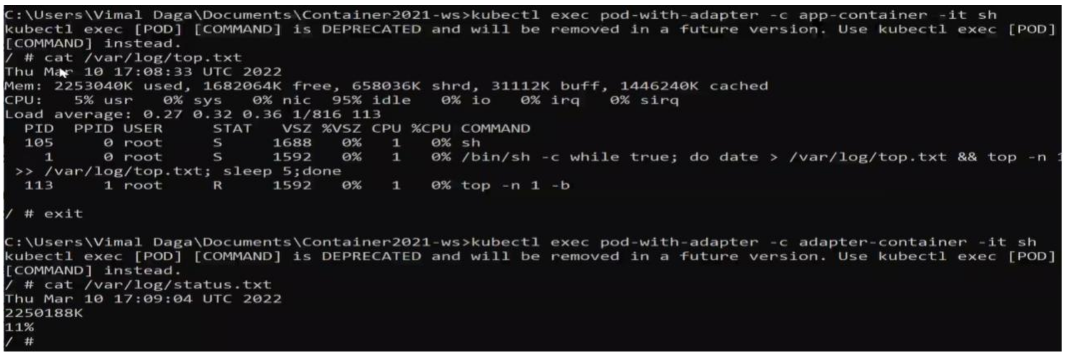
```
  # Main application container
  - name: app-container
    # This application writes system usage information (`top`) to a status
    # file every five seconds.
    image: alpine
    command: ["/bin/sh"]
    args: ["-c", "while true; do date > /var/log/top.txt && top -n 1 -b >> /var/log/top.txt; sleep 5;done"

    # Mount the pod's shared log file into the app
    # container. The app writes logs here.
    volumeMounts:
    - name: shared-logs
      mountPath: /var/log

  # Adapter container
  - name: adapter-container
    # This sidecar container takes the output format of the application
    # (the current date and system usage information), simplifies
    # and reformats it for the monitoring service to come and collect.

    # In this example, our monitoring service requires status files
    # to have the date, then memory usage, then CPU percentage each
    # on a new line.

    # Our adapter container will inspect the contents of the app's top file,
    # reformat it, and write the correctly formatted output to the status file.
    image: alpine
    command: ["/bin/sh"]

    # A long command doing a simple thing: read the `top.txt` file that the
    # application wrote to and adapt it to fit the status file format.
    # Get the date from the first line, write to `status.txt` output file.
    # Get the first memory usage number, write to `status.txt`.
    # Get the first CPU usage percentage, write to `status.txt`.

    args: ["-c", "while true; do (cat /var/log/top.txt | head -1 > /var/log/status.txt)
&& (cat /var/log/top.txt | head -2 | tail -1 | grep
 -o -E '\\d+\\w' | head -1 >> /var/log/status.txt) && (cat /var/log/top.txt | head -3 | tail -1 | grep
-o -E '\\d+%' | head -1 >> /var/log/status.txt); sleep 5; done"]


    # Mount the pod's shared log file into the adapter
    # container.
    volumeMounts:
    - name: shared-logs
      mountPath: /var/log
```

- Creating pod from a manifest file



```
C:\Users\Vimal Daga\Documents\Container2021-ws>kubectl apply -f adapter-pod.yaml.txt
pod/pod-with-adapter created

C:\Users\Vimal Daga\Documents\Container2021-ws>kubectl get pdos
error: the server doesn't have a resource type "pdos"

C:\Users\Vimal Daga\Documents\Container2021-ws>kubectl get po
NAME               READY   STATUS              RESTARTS   AGE
pod-with-adapter   0/2     ContainerCreating   0          7s
pod-with-sidecar   2/2     Running             0          10m
```

- Accessing the log file



```
C:\Users\Vimal Daga\Documents\Container2021-ws>kubectl exec pod-with-adapter -c app-container -it sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD]
[COMMAND] instead.
/ # cat /var/log/top.txt
Thu Mar 10 17:08:33 UTC 2022
Mem: 2253040K used, 1682064K free, 658036K shrd, 31112K buff, 1446240K cached
CPU:   5% usr   0% sys   0% nic  95% idle   0% io   0% irq   0% sirq
Load average: 0.27 0.32 0.36 1/816 113
  PID  PPID USER     STAT   VSZ %VSZ CPU %CPU COMMAND
  105     0 root     S     1688   0%   1   0% sh
    1     0 root     S     1592   0%   1   0% /bin/sh -c while true; do date > /var/log/top.txt && top -n
 >> /var/log/top.txt; sleep 5;done
  113     1 root     R     1592   0%   1   0% top -n 1 -b

/ # exit

C:\Users\Vimal Daga\Documents\Container2021-ws>kubectl exec pod-with-adapter -c adapter-container -it sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD]
[COMMAND] instead.
/ # cat /var/log/status.txt
Thu Mar 10 17:09:04 UTC 2022
2250188K
11%
/ #
```