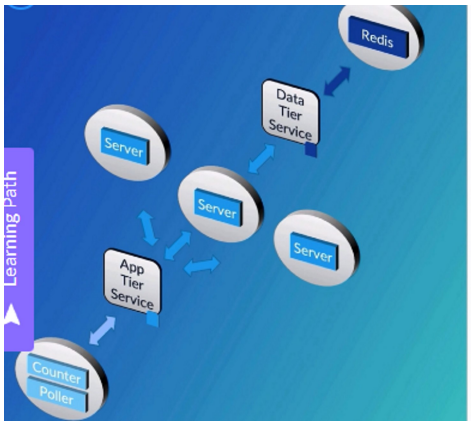
5. Service Discovery

27 February 2022 15:26

Service discovery



Compare with the previous design why this is best

Why Services?

- Supports multi-Pod design
- Provides static endpoint for each tier
- Handles Pod IP changes
- Load balancing

Service Discovery Mechanisms

Environment Variables	DNS
• Services address automatically injected in containers	• DNS records automatically created in cluster's DNS
• Environment variables follow naming conventions based on service name	• Containers automatically configured to query cluster DNS

```
admin@Jamess-MBP src % kubectl create -f 4.1-namespace.yaml
namespace/service-discovery created
admin@Jamess-MBP src % cat
```

→ creat new namespace.

```
admin@Jamess-MBP src % cat 4.2-data_tier.yaml
apiVersion: v1
kind: Service
metadata:
  name: data-tier
  labels:
    app: microservices
spec:
  ports:
  - port: 6379
    protocol: TCP # default
    name: redis # optional when only 1 port
  selector:
    tier: data
  type: ClusterIP # default
---
apiVersion: v1
kind: Pod
metadata:
  name: data-tier
  labels:
    app: microservices
    tier: data
spec:
  containers:
  - name: redis
```

→ name same for service & pod

→ naming part is mandatory if we have multiple ports.

we can create as many resources as we want seperated by 3 - - -

selector tier → data

This is how linking happens.

label tier : data

type : cluster ip → default type, creates virtual ip inside cluster for internal access only.

```
in@Jamess-MBP src % kubectl create -f 4.2-data_tier.yaml -n service-discovery
```

Cresources are created in the order they are listed in the file.

Resources are created in the order they are "run"

```
admin@Jamess-MBP src % kubectl describe service -n service-discovery data-tier
Name:              data-tier
Namespace:         service-discovery
Labels:            app=microservices
Annotations:       <none>
Selector:          tier=data
Type:              ClusterIP
IP:                10.98.110.202
Port:              redis  6379/TCP
TargetPort:        6379/TCP
Endpoints:         172.17.0.3:6379
Session Affinity:  None
Events:            <none>
admin@Jamess-MBP src %
```

→ cluster IP

```
tier: app
---
apiVersion: v1
kind: Pod
metadata:
  name: app-tier
  labels:
    app: microservices
    tier: app
spec:
  containers:
    - name: server
      image: lrakai/microservices:server-v1
      ports:
        - containerPort: 8080
      env:
        - name: REDIS_URL
          # Environment variable service discovery
          # Naming pattern:
          #   IP address: <all_caps_service_name>_SERVICE_HOST
          #   Port: <all_caps_service_name>_SERVICE_PORT
          #   Named Port: <all_caps_service_name>_SERVICE_PORT_<all_caps_port_name>
          value: redis://$(DATA_TIER_SERVICE_HOST):$(DATA_TIER_SERVICE_PORT_REDIS)
          # In multi-container example value was
          # value: redis://localhost:6379
in@Jamess-MBP src %
```

environment variables are auto created by the K8s we just need naming convention proper.

ex:- NAME_SERVICE_HOST

↳ data-tier

↳ hyphen replaced by underscore

$(DATA_TIER_SERVICE_HO → ALL CAPS.

$(DATA_TIER_SERVICE_PORT_REDIS)

↳ Port name.

To use environment variables the service should be created first before using/defining.

service should also be in same namespace where the env variables are used

```
admin@Jamess-MBP src % cat 4.4-support_tier.yaml
apiVersion: v1
kind: Pod
metadata:
  name: support-tier
  labels:
    app: microservices
    tier: support
spec:
  containers:

    - name: counter
      image: lrakai/microservices:counter-v1
      env:
```

→ support tier.

```
spec:
  containers:

    - name: counter
      image: lrakai/microservices:counter-v1
      env:
        - name: API_URL
          # DNS for service discovery
          # Naming pattern:
          #   IP address: <service_name>.<service_namespace>
          #   Port: needs to be extracted from SRV DNS record
          value: http://app-tier.service-discovery:8080
    - name: poller
      image: lrakai/microservices:poller-v1
      env:
        - name: API_URL
          # omit namespace to only search in the same namespace
```

k8's create DNS for every service.

if in same namespace no need to mention.

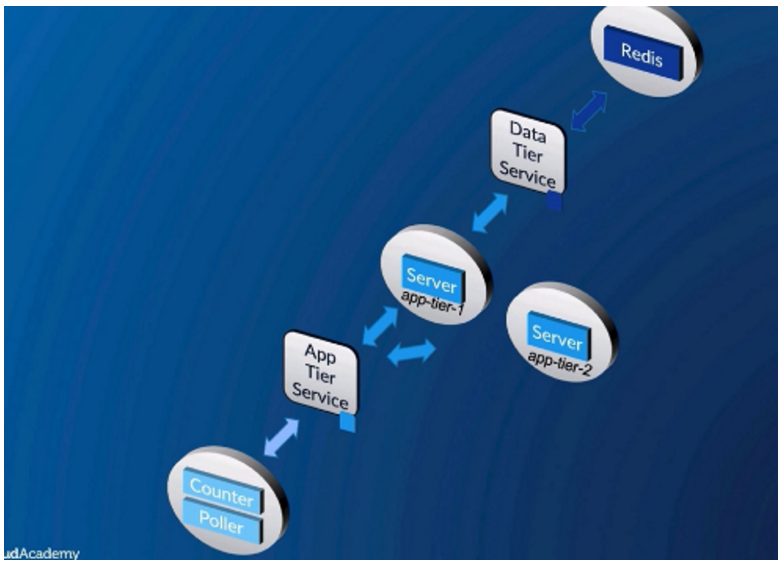
```
    - name: poller
      image: lrakai/microservices:poller-v1
      env:
        - name: API_URL
          # omit namespace to only search in the same namespace
          value: http://app-tier:$(APP_TIER_SERVICE_PORT)
in@Jamess-MBP src %
```

continuation hard code (or) use env. variable.

namespace omitted

DNS does not require underscore or all CAPS.

```
admin@Jamess-MBP src % kubectl create -f 4.4-support_tier.yaml -n service-discover
pod/support-tier created
in@Jamess-MBP src % kubectl get pods -n service-discovery
E            READY   STATUS    RESTARTS   AGE
-tier        1/1     Running   0          4m3s
a-tier       1/1     Running   0          12m
port-tier    2/2     Running   0          11s
```



-f on logs will give log on trail -