João Luzio

From: Fredy Sierra Umaña via Backbase Community <community@backbase.com>
Sent: 23 de agosto de 2021 09:13
To: João Luzio
Subject: [Backbase Community] Using Telepresence to run services locally

Follow Up Flag: Follow up
Flag Status: Flagged

Fredy Sierra Umaña **Fredy**
August 16

Why use Telepresence?

To develop a Backbase service and run it locally, you also need to run other services like Access Control, Edge Service, Registry Service, and Identity Services. Running that amount of services in a local environment can be challenging for your machine's memory and CPU.

Telepresence 2 can mitigate these challenges by proxying Kubernetes instance communication to your local machine environment. With Telepresence you can run one service locally, while the rest of the application runs in the cloud. This is important because local development loops (build/deploy/test) are fast and allow the use of local tools like IDE and debuggers.

This blog post shows how to use Telepresence, with examples based on the Backbase **Golden Sample**. The Golden Sample is a set of example services we provide to illustrate best practices.

Assumptions

To use Telepresence you must be an AWS user with admin access to the Kubernetes cluster. The only restriction of our current user is `arn:aws:iam::aws:policy/ReadOnlyAccess` . This gives total control of the cluster.

Installation and use

An empty cluster will be used

```
└─ kubectl get ns
NAME              STATUS   AGE
default           Active   12d
kube-node-lease   Active   12d
kube-public       Active   12d
kube-system       Active   12d
```

Set up the Backbase backend platform

When you develop a Backbase Backend service you require other components such as Edge, Kubernetes Discovery, ActiveMQ, and a database. The first step for using Telepresence is to deploy your platform in the cluster. For the purpose of this document, **Golden sample** will be used

The above services are deployed to the default Kubernetes namespace. Note that this deployment includes the service you want to intercept.

```
└─ kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP
activemq        ClusterIP      172.20.18.182    <none>
auth            ClusterIP      172.20.136.166   <none>
edge            LoadBalancer   172.20.177.147   ab4201184519e4b7d8baea439319355d-538893185.eu-west-1.elb.amazonaws.
kubernetes      ClusterIP      172.20.0.1       <none>
mysql           ClusterIP      172.20.27.37     <none>
product         ClusterIP      172.20.54.17     <none>
review          ClusterIP      172.20.92.113    <none>
```

To check if your deployment and setup are correct, use the endpoint **http://yourcluster/api/store/client-api/v1/product-composite/113** .

Set up Telepresence

The next step is to install Telepresence in the cluster. For that, the **First user** installs the Telepresence client, and when trying to connect to the cluster they install the required Telepesence components.

To install on a Mac, run the following command:

```
brew install datawire/blackbird/telepresence
```

```
└─ brew install datawire/blackbird/telepresence
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/cask).
==> Updated Casks
Updated 1 cask.

==> Installing telepresence from datawire/blackbird
==> Downloading https://app.getambassador.io/download/tel2/darwin/amd64/2.3.2/telepresence
```

In the above image, the version installed was “2.3.2“. Note that other Telepresence users for this cluster will have to use the same client version. Once installed, check the installation with the following command:

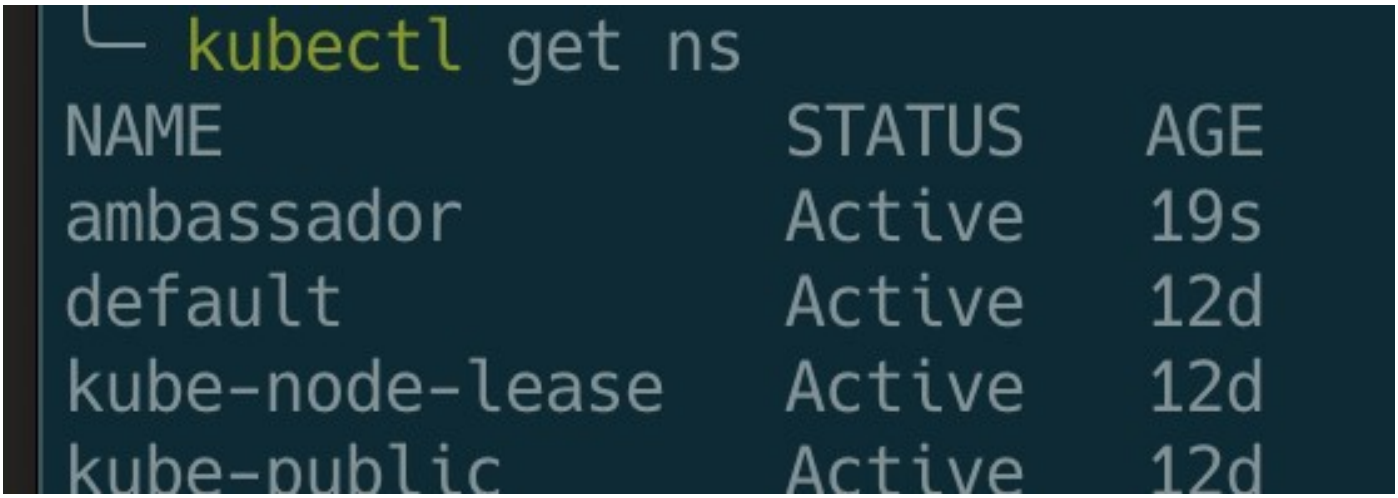
```
└─ telepresence status
Root Daemon: Not runnir
```

Before connecting Telepresence client for the first time, verify that you're in the correct Kubernetes context:

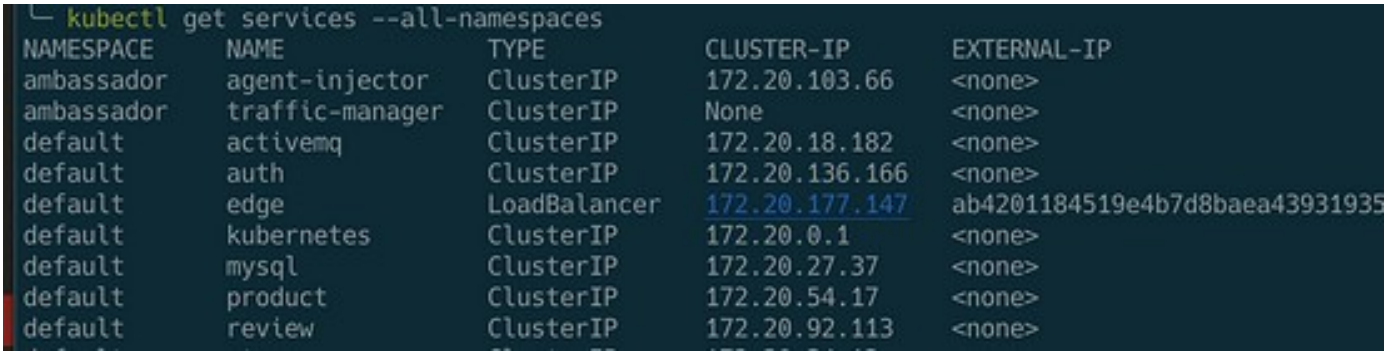
The first user trying to connect using Telepresence will install the required services in the cluster.

```
telepresence connect
```

Notice that after running the previous comment a new namespace (ambassador) has been created. This is only possible if the current user has the appropriate permissions.



Inside the new ambassador namespace, you see two services: the agent-injector and the traffic manager. For more information about those services see **the Telepresence architecture docs**.



Intercepting

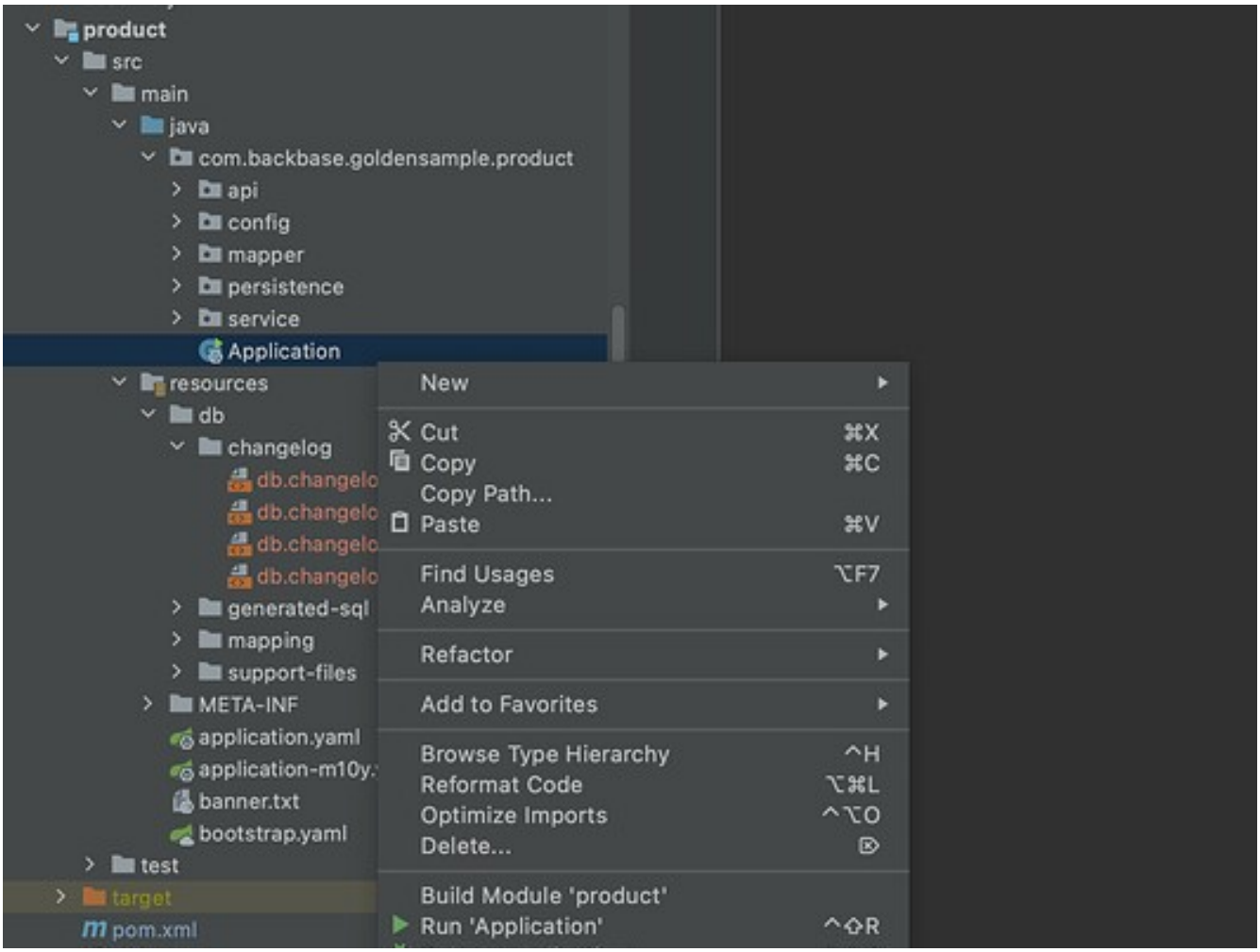
The service you want to run in your local environment should be on the list of services to intercept, so the service should be initially deployed in the Kubernetes cluster.

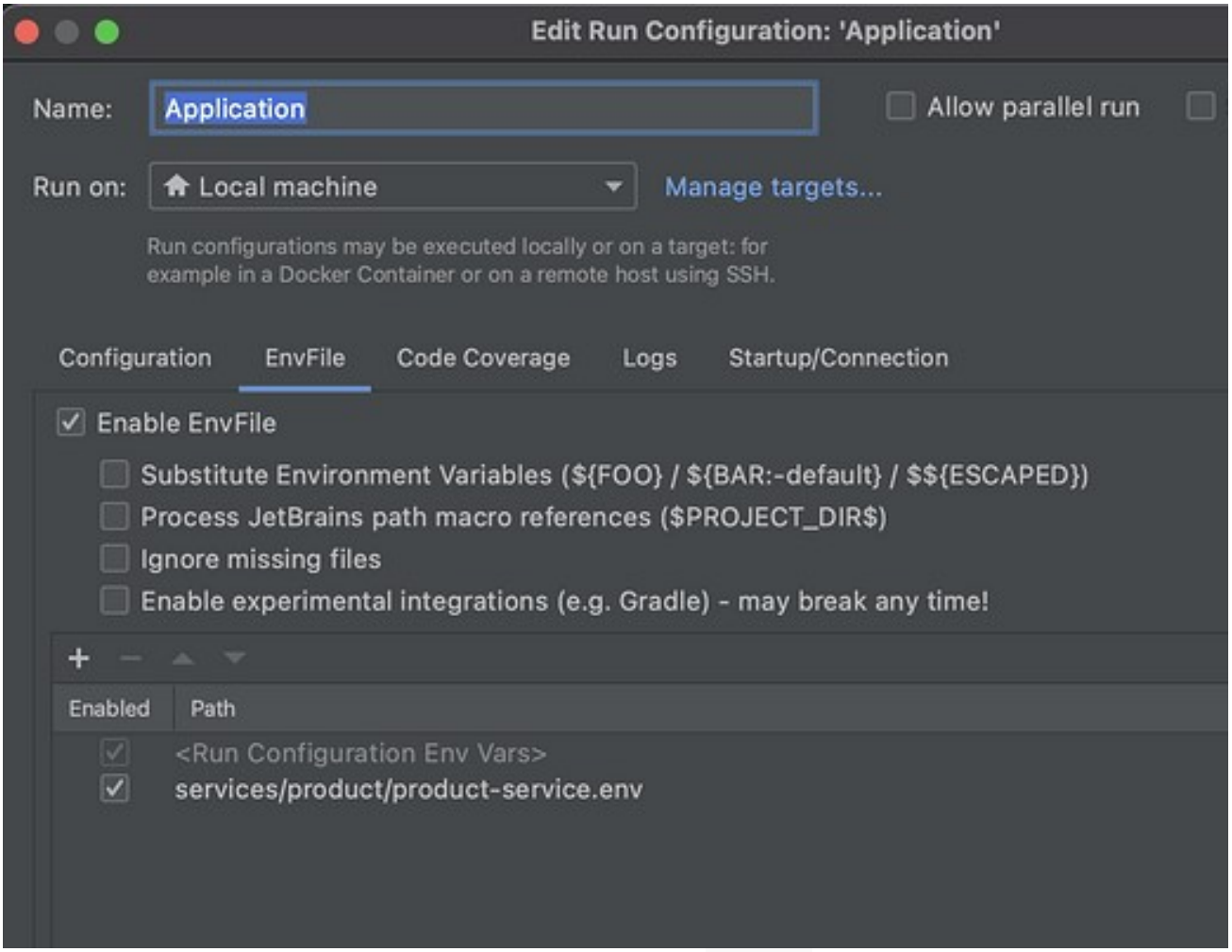
Use `telepresence list` command to obtain the list of services to intercept.

```
telepresence list --namespace default
activemq                          : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-authentication-dev : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-bas-token-converter: ready to intercept (traffic-agent not yet installed)
golden-local-k8s-edge             : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-product          : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-review           : ready to intercept (traffic-agent not yet installed)
```

The previous command generates a file (product-service.env) that Intellij can use to start a local `product` service.

For IntelliJ you can use the **EnvFile** plugin to run the service locally using the environment variables in the file.





Now you can run or debug your service locally. In order to run `product` in your local environement, you can use the `docker-image` profile. This provides you with the required driver dependencies.

With this current configuration, any request to product service will be redirected to your local. You might be blocking requests from other developers or those requests might interfere with your debug process.

When you finish your debugging or development process, use `telepresence leave` to end the service interception.

First, use `telepresence list` to obtain the intercept name. In this example, it's `golden-local-k8s-product-default`

```
└─ telepresence list
activemq                          : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-authentication-dev : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-bas-token-converter: ready to intercept (traffic-agent not yet installed)
golden-local-k8s-edge             : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-product          : intercepted
    Intercept name         : golden-local-k8s-product-default
    State                  : ACTIVE
    Workload kind          : Deployment
    Destination            : 127.0.0.1:8080
    Service Port Identifier: 8080
    Intercepting           : all TCP connections
```

With the `intercept name` you can use the leave command as follows:

```
telepresence leave golden-local-k8s-product-default
fredy ~
telepresence list
activemq                          : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-authentication-dev : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-bas-token-converter: ready to intercept (traffic-agent not yet installed)
golden-local-k8s-edge             : ready to intercept (traffic-agent not yet installed)
golden-local-k8s-product          : ready to intercept (traffic-agent already installed)
golden-local-k8s-review           : ready to intercept (traffic-agent not yet installed)
```

Service to Service communication with Telepresence.

In **Golden sample**, the service `store` aggregates information from the services `product` and `review`

```
telepresence intercept golden-local-k8s-store --port 8080:8080 --env-file store-service5.env --
Using Deployment golden-local-k8s-store
intercepted
    Intercept name         : golden-local-k8s-store-tele
    State                  : ACTIVE
    Workload kind          : Deployment
    Destination            : 127.0.0.1:8080
    Service Port Identifier: 8080
```

The generated file store-service5.env can be used to run the service using Intellij. However, the environment variable `KUBERNETES_NAMESPACE` is required by the Kubernetes client in order to find the required services (product and review). The value is the Kubernetes namespace where the service to intercept is deployed. Add that environment variable to `store-service5.env` file.

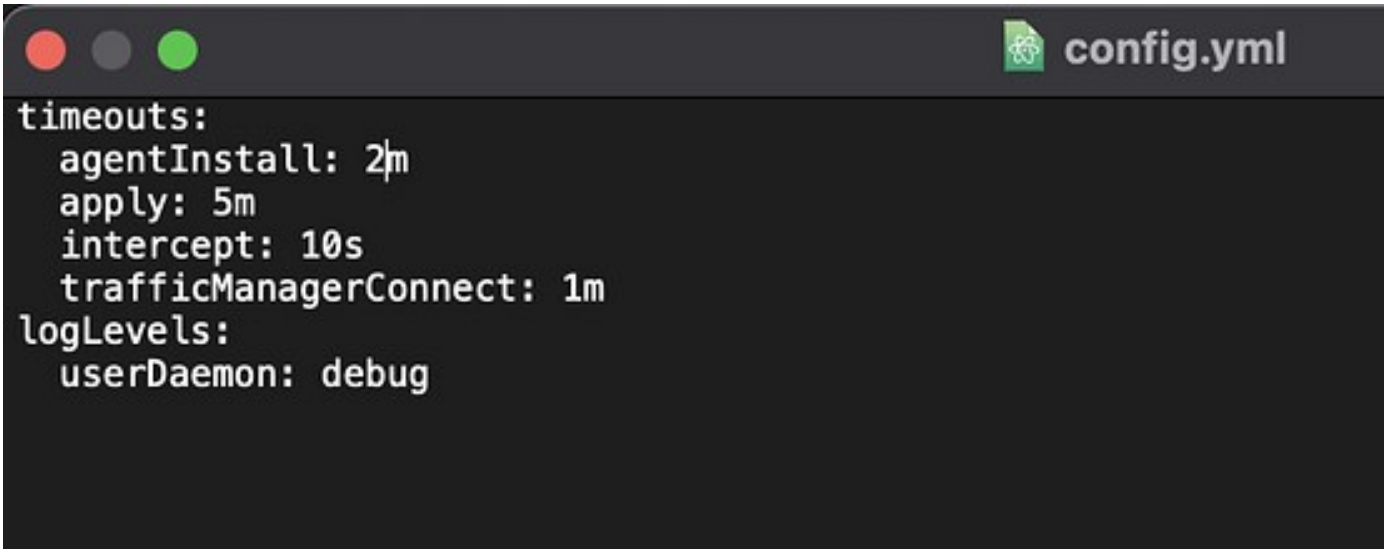
```
backbase.review-service.serviceId=review
backbase.review-service.servicePort=8080
beckbase.product-service.serviceId=product
logging.level.com.backbase.buildingblocks.communication.http.InterServiceLogger=DEBUG
management.endpoint.env.enabled=true
management.endpoints.web.exposure.include=info,health,beans,bindings,gateway,prometheus
server.port=8080
```

Possible Errors

If there's a lot of network changes and instability, `telepresence` can fail:

```
helm git:(golden) x telepresence connect
Connecting to traffic manager...
Disconnecting...done
telepresence: error: connector.Connect: the port-forward connection to the traffic manager timed out.  The current timeout 1m0s can be configured as "timeouts.tra
/Library/Application Support/telepresence/config.yml"
```

Following the provided error advice you can configure timeouts in the `config.yml` file:



If you continue having problems. you can:

- Use `telepresence quit`
- Reset network
- Reconnect VPN

Try To Remain Calm....😇

Visit Topic to respond.

To unsubscribe from these emails, **click here**.