Socket application with MicroK8s on Ubuntu 20.04

"MicroK8s is a powerful, lightweight, reliable production-ready Kubernetes distribution. It is an enterprise-grade Kubernetes distribution that has a small disk and memory footprint while offering carefully selected add-ons out-the-box, such as Istio, Knative, Grafana, Cilium and more. Whether you are running a production environment or interested in exploring K8s, MicroK8s serves your needs," see Introduction to MicroK8s.

Check whether ar-statefulset, gl-statefullset and cnt-statefullset pods are running

Run the next command from the terminal:

microk8s.kubectl get pods -n socket-ns

```
NAME                 READY   STATUS    RESTARTS   AGE
cnt-statefulset-0    2/2     Running   0          160m
gl-statefulset-0     2/2     Running   0          159m
gl-statefulset-1     2/2     Running   0          159m
ar-statefulset-1     2/2     Running   0          98m
ar-statefulset-0     2/2     Running   0          97m
```

If pods are not running use next commands from the terminal

- Install microk8s: sudo snap install microk8s –classic
- Enable microk8s build-in registry: microk8s enable registry
- Clone files from GitHub: git clone https://github.com/rbontekoe/ar.git
- Enter the folder with the downloaded files: cd ar
- Download Julia 1.6.0: curl -O https://julialang-s3.julialang.org/bin/linux/x64/1.6/julia-1.6.0-linux-x86_64.tar.gz
- Create accounts receivable image: docker build –no-cache -f ar.Dockerfile -t localhost:32000/i_ar:v1.0.16 .
- Copy to local registry: docker push localhost:32000/i_ar:v1.0.16
- Create general ledger image: docker build –no-cache -f gl.Dockerfile -t localhost:32000/i_gl:v1.0.3 .
- Copy to local registry: docker push localhost:32000/i_gl:v1.0.3
- Create counter image: docker build –no-cache -f cnt.Dockerfile -t localhost:32000/i_cnt:v1.0.1 .
- Copy to local registry: docker push localhost:32000/i_cnt:v1.0.1
- microk8s.kubectl apply -f ar-storage.yaml
- microk8s.kubectl apply -f cnt-storage.yaml
- microk8s.kubectl apply -f gl-storage.yaml

Import AppliAR.jl

```
import Pkg; Pkg.add(url="https://github.com/rbontekoe/AppliAR.jl")
```

Load the files

```
using Sockets, Serialization, AppliSales, AppliAR, AppliGeneralLedger, DataFrames,
Query
```

Delete the old data files

Go to the terminal and erase the next files:

sudo rm /var/data-ar/unpaid-invoices.txt /var/data-ar/paid-invoices.txt /var/data-gl/journal.txt /var/data-gl/generalledger.txt /var/data-cnt/seqnbr.txt

Connect to AppliAR.jl, create and process the orders

```
clientside = TCPSocket(RawFD(19) open, 0 bytes waiting)
```

```
clientside = connect(ip"127.0.0.1", 30012) # connect to accounts receivable pod
```

```
begin
    sales = AppliSales.process() # create sales orders
    serialize(clientside, sales) # send orders to account receivable
end
```

	accountid	customerid	invoice_nbr	debit	credit	descr
1	1300	"Scrooge Investment Bank"	"1001"	1210.0	0.0	"Learn Smiling'
2	1300	"Duck City Chronicals"	"1002"	2420.0	0.0	"Learn Smiling'
3	1300	"Donalds Hardware Store"	"1003"	1210.0	0.0	"Learn Smiling'

```
begin
    r = AppliGeneralLedger.read_from_file("/var/data-gl/generalledger.txt")
    df = r |> @filter(_.accountid == 1300) |> DataFrame
    df[:, [:accountid, :customerid, :invoice_nbr, :debit, :credit, :descr]]
end
```

Load and process the bank statements

```
stms =
  [BankStatement(2020-01-15, "Duck City Chronicals Invoice 1002", "NL93INGB", 2420.0), Bar
```

```
stms = AppliAR.read_bank_statements("./bank-kubernetes.csv") # retrieve the
bankstatements
```

```
serialize(clientside, stms) # create paid invoices and update general ledger
```

Display Accounts Receivable

Other accounts are:

```
8000 - Sales
1150 - Bank
4000 - VAT
1300 - Accounts Receivable
```

```
accountid = 1300;
```

	accountid	customerid	invoice_nbr	debit	credit	descr
1	1300	"Scrooge Investment Bank"	"1001"	1210.0	0.0	"Learn Smiling'
2	1300	"Duck City Chronicals"	"1002"	2420.0	0.0	"Learn Smiling'
3	1300	"Donalds Hardware Store"	"1003"	1210.0	0.0	"Learn Smiling'
4	1300	"Duck City Chronicals"	"1002"	0.0	2420.0	"Learn Smiling'
5	1300	"Donalds Hardware Store"	"1003"	0.0	1210.0	"Learn Smiling'

```
begin
    r2 = AppliGeneralLedger.read_from_file("/var/data-gl/generalledger.txt")
    df2 = r2 |> @filter(_.accountid == accountid) |> DataFrame
    df2[:, [:accountid, :customerid, :invoice_nbr, :debit, :credit, :descr]]
end
```

Display the status of the unpaid invoices

	id_inv	csm	inv_date	amount	days
1	"1001"	"Scrooge Investment Bank"	2022-01-19	1210.0	0 days

```
begin
    r1 = AppliAR.aging("/var/data-ar/unpaid-invoices.txt", "/var/data-ar/paid-invoices.txt")
    result = DataFrame(r1)
end
```