Dapr Introduction

This sample introduces Dapr and how to use it to build microservices and is based on the Dapr quickstarts.

It contains two projects:

- `food-dapr-backend` - A .NET Core Web API project that uses Entity Framework and Dapr to store and retrieve food orders.
- `food-dapr-frontend` - An MVC project that uses Dapr to call the backend API.

Dapr configuration is stored in the components folder and container the following files:

- `statestore.yaml` - Configures the state store to use Azure Blob Storage.

Readings

Dapr CLI

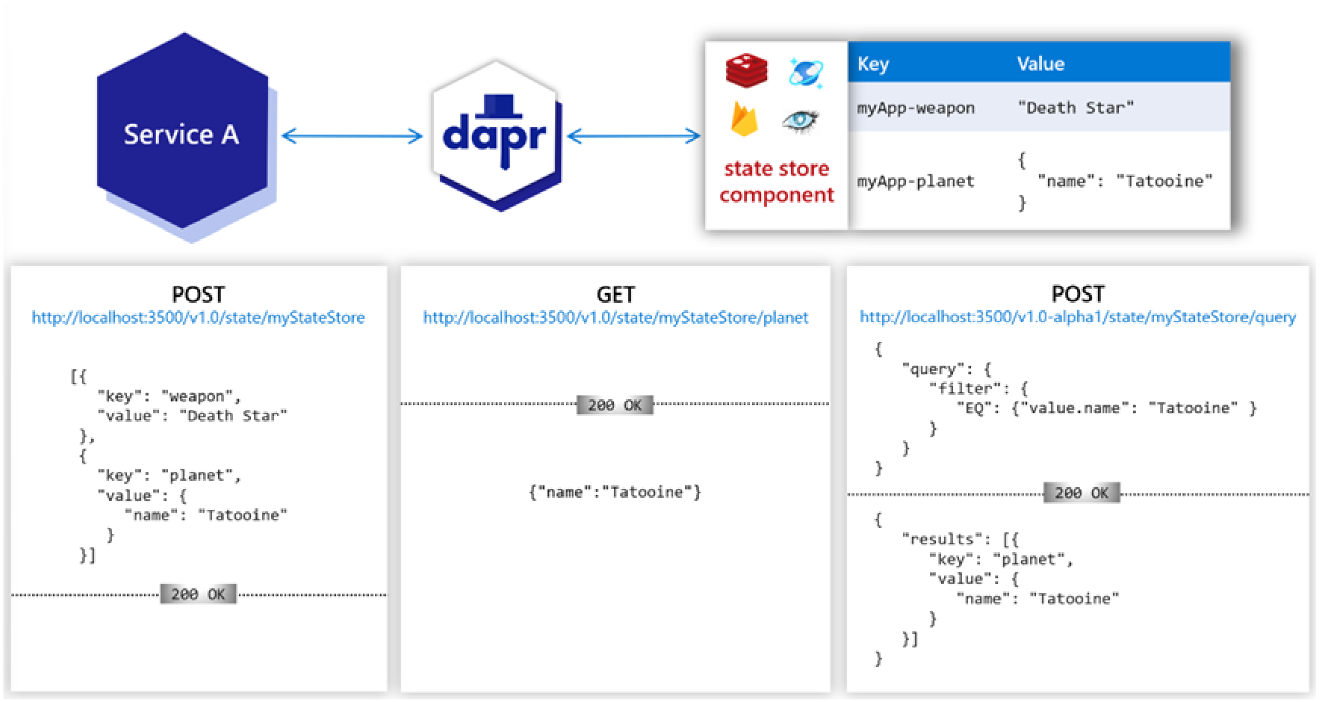
Dapr Visual Studio Code extension

Developing Dapr applications with Dev Containers

Getting started, Basic State & Deployment to Azure Container Apps

> Note: This demo assumes that you have created an Azure Container Apps environment. If you haven't done so, please follow the instructions to create one.

Getting started & Basic State


Service A
dapr
state store component
Key
Value
myApp-weapon
"Death Star"
myApp-planet
{ "name": "Tatooine" }
POST
http://localhost:3500/v1.0/state/myStateStore
200 OK
GET
http://localhost:3500/v1.0/state/myStateStore/planet
200 OK
POST
http://localhost:3500/v1.0-alpha1/state/myStateStore/query
200 OK

```
[{
   "key": "weapon",
   "value": "Death Star"
},
{
   "key": "planet",
   "value": {
      "name": "Tatooine"
   }
}]
```

```
{"name":"Tatooine"}
```

```
{
   "query": {
      "filter": {
         "EQ": {"value.name": "Tatooine" }
      }
   }
}
```

```
{
   "results": [{
      "key": "planet",
      "value": {
         "name": "Tatooine"
      }
   }]
}
```

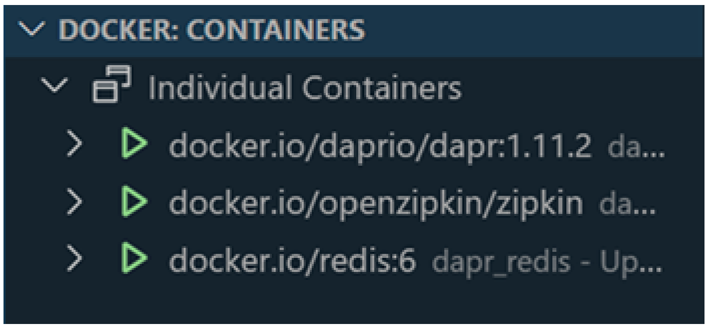
- Install Dapr CLI

```
Set-ExecutionPolicy RemoteSigned -scope CurrentUser
powershell -Command "iwr -useb
https://raw.githubusercontent.com/dapr/cli/master/install/install.ps1 | iex"
```

> Note: Restart the terminal after installing the Dapr CLI

- Initialize default Dapr containers and check running containers:

```
dapr init
```



docker.io/daprio/dapr:1.11.2 da...
docker.io/openzipkin/zipkin da...
docker.io/redis:6 dapr_redis - Up...

> Note: To remove the default Dapr containers run `dapr uninstall`

- Run project `food-dapr-backend`

```
cd food-dapr-backend
dapr run --app-id food-backend --app-port 5001 --dapr-http-port 5010 dotnet
run --launch-profile https
```

- Test the API by invoking `http://localhost:5000/food` using the dapr sidecar:

```
GET http://localhost/<dapr-http-port>/v1.0/invoke/<app-id>/method/<method-
name>
GET http://localhost:5010/v1.0/invoke/food-backend/method/food
```

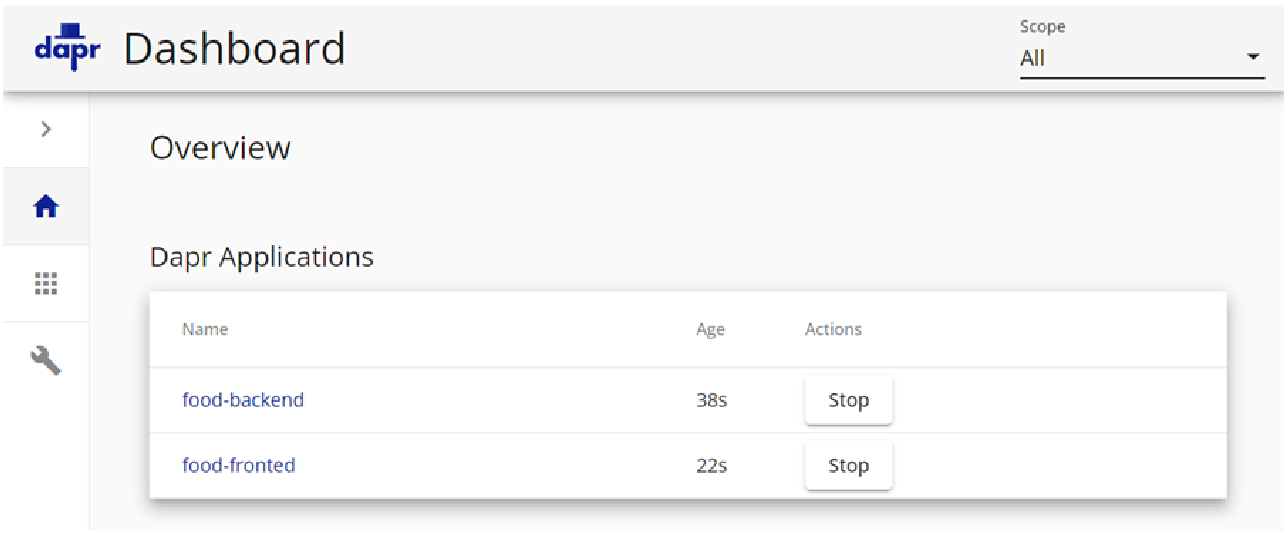
- Run project `food-dapr-fronted`

```
cd food-dapr-fronted
dapr run --app-id food-fronted --app-port 5002 --dapr-http-port 5011 dotnet
run
```

- Show Dapr Dashboard

```
dapr dashboard
```

- Examine Dapr Dashboard on http://localhost:8080:



Dapr Applications

Name	Age	Actions
food-backend	38s	Stop
food-fronted	22s	Stop

Running with Tye

- Install Tye. Project Tye is an experimental developer tool that makes developing, testing, and deploying microservices and distributed applications easier

```
dotnet tool install -g Microsoft.Tye --version "0.11.0-alpha.22111.1"
```

- A typical tye file could look like this:

```
name: dapr-services
services:
- name: food-dapr-backend
project: food-dapr-backend/food-dapr-backend.csproj
bindings:
- port: 5000
- name: food-dapr-frontend
project: food-dapr-frontend/food-dapr-frontend.csproj
bindings:
- port: 5002
```

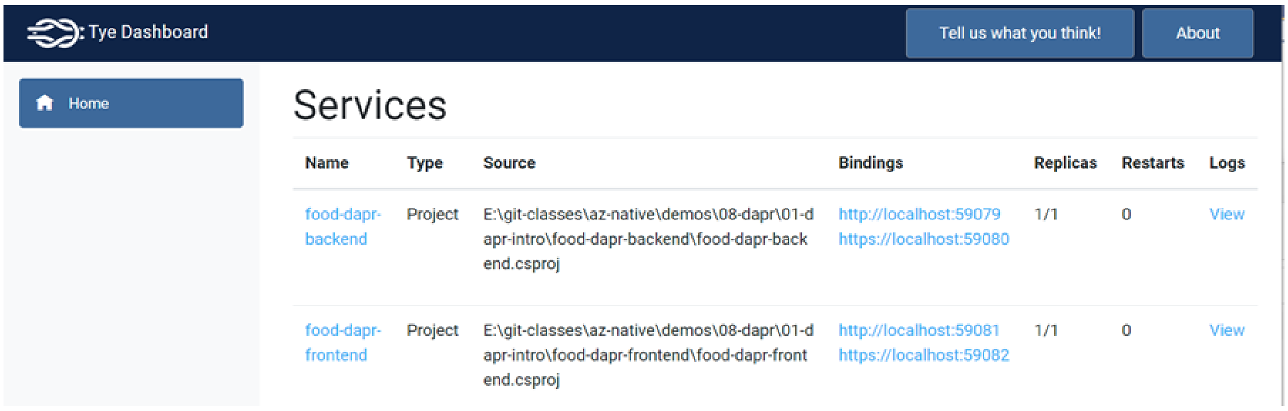
- Create a `tye.yaml` file in the root of the solution by running:

```
tye init
```

> Note: You can skip this step as the `tye.yaml` file is already included in the solution.

- Run the two projects with Tye

```
tye run
```



Services

Name	Type	Source	Bindings	Replicas	Restarts	Logs
food-dapr-backend	Project	E:\git-classes\az-native\demos\08-dapr\01-dapr-intro\food-dapr-backend\food-dapr-backend.csproj	http://localhost:59079 https://localhost:59080	1/1	0	View
food-dapr-frontend	Project	E:\git-classes\az-native\demos\08-dapr\01-dapr-intro\food-dapr-frontend\food-dapr-frontend.csproj	http://localhost:59081 https://localhost:59082	1/1	0	View

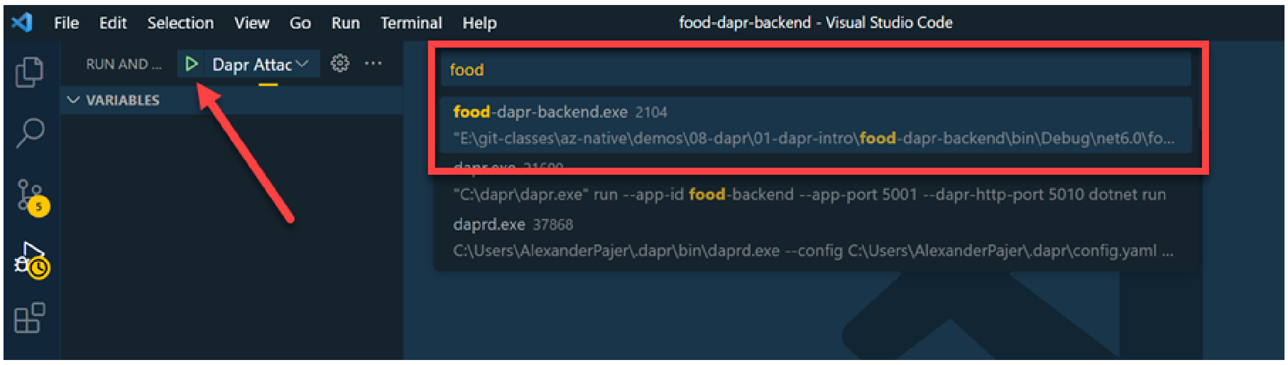
Using Default State Store

- Examine `CountController.cs` and call it multiple times to increment the counter:

```
[HttpGet("getCount")]
public async Task<int> Get()
{
    var daprClient = new DaprClientBuilder().Build();
    var counter = await daprClient.GetStateAsync<int>(storeName, key);
    await daprClient.SaveStateAsync(storeName, key, counter + 1);
    return counter;
}
```

- Examine the `Dapr Attach` config in `launch.json` and use it to attach the debugger to the `food-dapr-backend` process and debug the state store code:

```
{
    "name": "Dapr Attach",
    "type": "coreclr",
    "request": "attach",
    "processId": "${command:pickProcess}"
}
```


File Edit Selection View Go Run Terminal Help
food-dapr-backend - Visual Studio Code
RUN AND ...
Dapr Attac
VARIABLES

```
food
food-dapr-backend.exe 2104
"E:\git-classes\az-native\demos\08-dapr\01-dapr-intro\food-dapr-backend\bin\Debug\net6.0\fo...
dapr.exe 21600
"C:\dapr\dapr.exe" run --app-id food-backend --app-port 5001 --dapr-http-port 5010 dotnet run
daprd.exe 37868
C:\Users\AlexanderPajer\.dapr\bin\daprd.exe --config C:\Users\AlexanderPajer\.dapr\config.yaml ...
```

Deploy to Azure Container Apps

- Build the food-dapr-backend image

```
env=dev
grp=az-native-$env
loc=westeurope
acr=aznative$env
imgBackend=food-dapr-backend:v1
az acr build --image $imgBackend --registry $acr --file dockerfile .
```

- Create a storage account to be used as state store

```
stg=aznative$env
az storage account create -n $stg -g $grp -l $loc --sku Standard_LRS
```

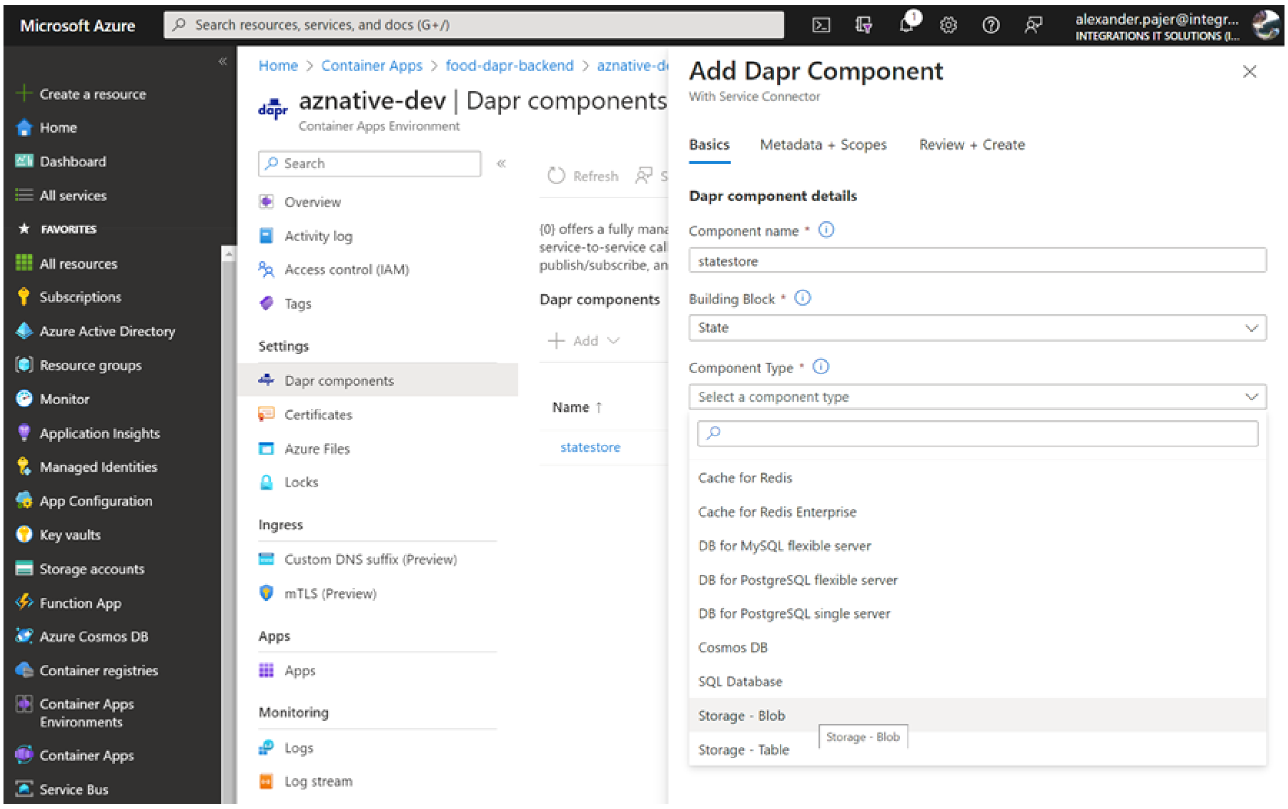
- Update its values in `components/statestore.yml`

```
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
name: statestore
spec:
type: state.azure.blobstorage
metadata:
    - name: storageAccount
    value: aznative$env
    - name: storageAccessKey
    value: <storage-account-key>
```

- Add the Dapr component to the Azure Container Apps environment

```
az containerapp env dapr-component set -n $acaenv -g $grp \
--dapr-component-name statestore \
--yaml './components/statestore.yml'
```

> Note. In Azure Portal you can see the Dapr component in the Azure Container Apps environment:



Component name * statestore

Building Block * State

Component Type * Select a component type

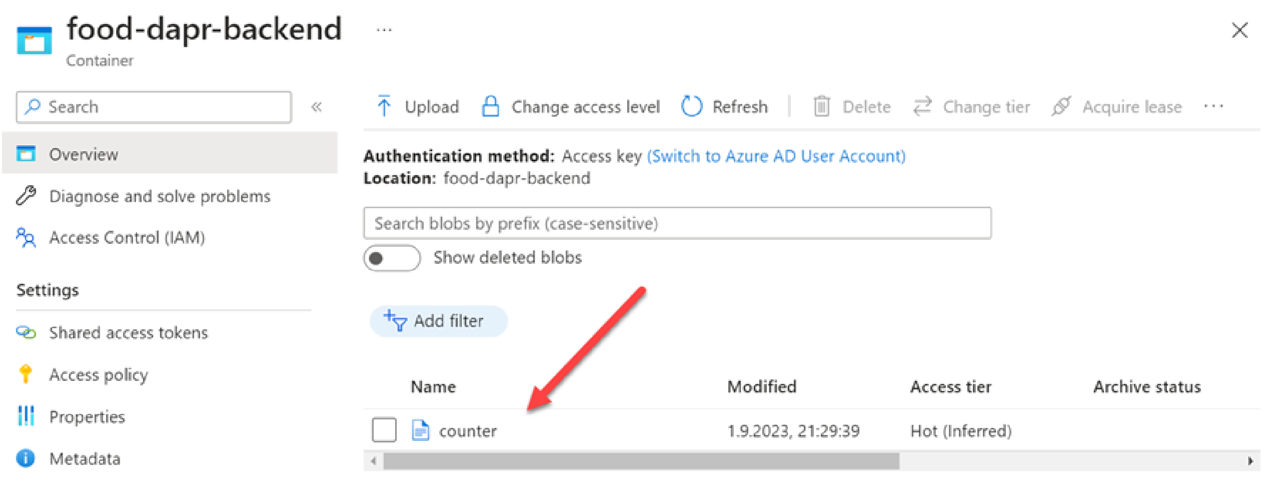
- Execute deploy-app.azcli to create the container app

```
az containerapp create -n $appBackend -g $grp \
--image $imgBackend \
--environment $acaenv \
--target-port 80 --ingress external \
--min-replicas 0 --max-replicas 1 \
--enable-dapr \
--dapr-app-port 80 \
--dapr-app-id $appBackend \
--registry-server $loginSrv \
--registry-username $acr \
--registry-password $pwd
```

- Execute the /count/getCount method multiple times to increment the counter

```
curl -X GET "http://<URL>.$loc.azurecontainer.io/count/getCount" -H
"accept: text/plain"
```

- Examine the storage account to see the state store data



Authentication method: Access key (Switch to Azure AD User Account)
Location: food-dapr-backend

Name	Modified	Access tier	Archive status
counter	1.9.2023, 21:29:39	Hot (Inferred)	