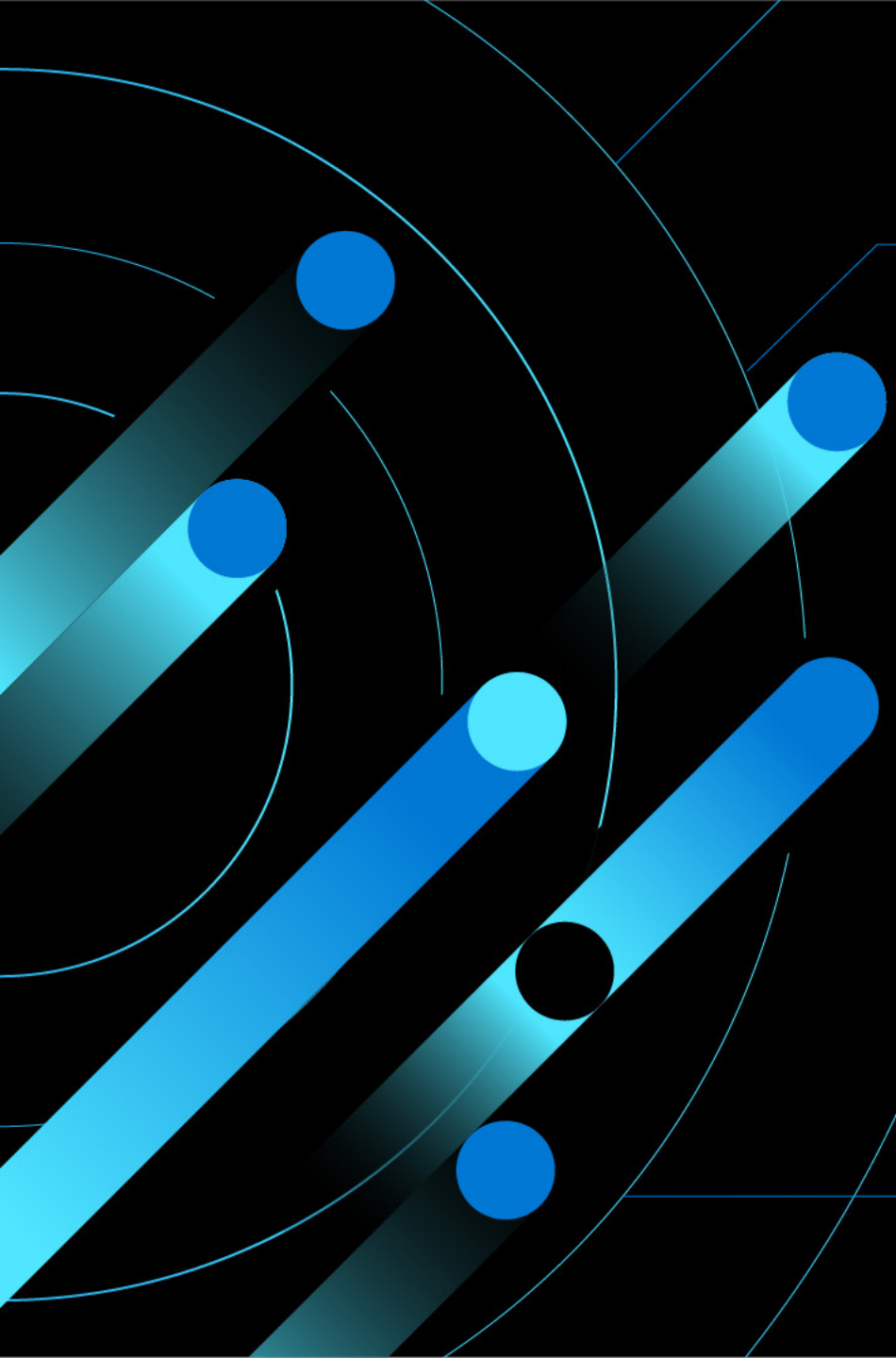


apr ay 云原生运行时开发者日

Developing Apps with Dapr

Artur Souza Engineering Manager

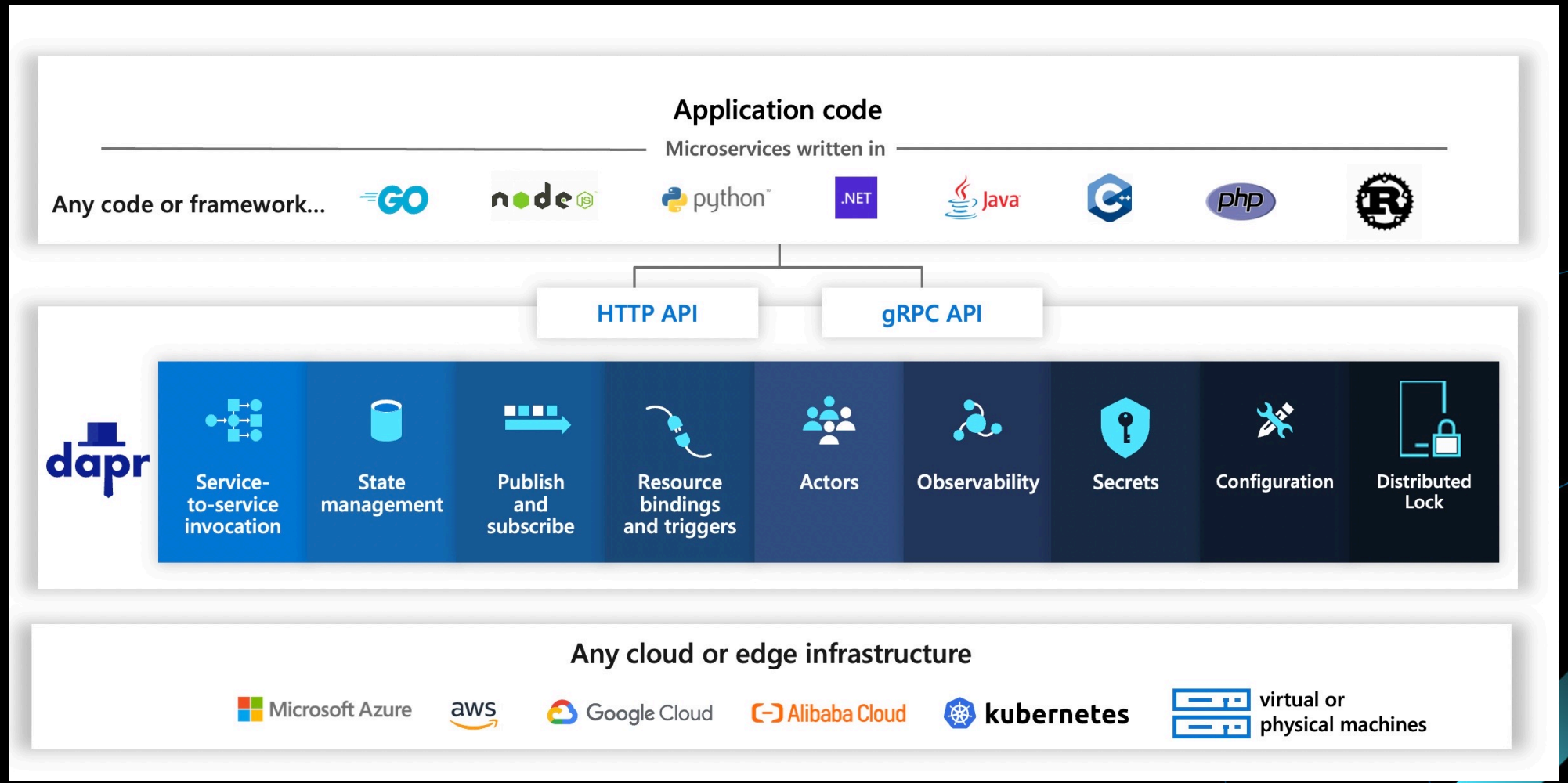
联合主办：

Microsoft Azure intel Dapr China Community

What is Dapr?

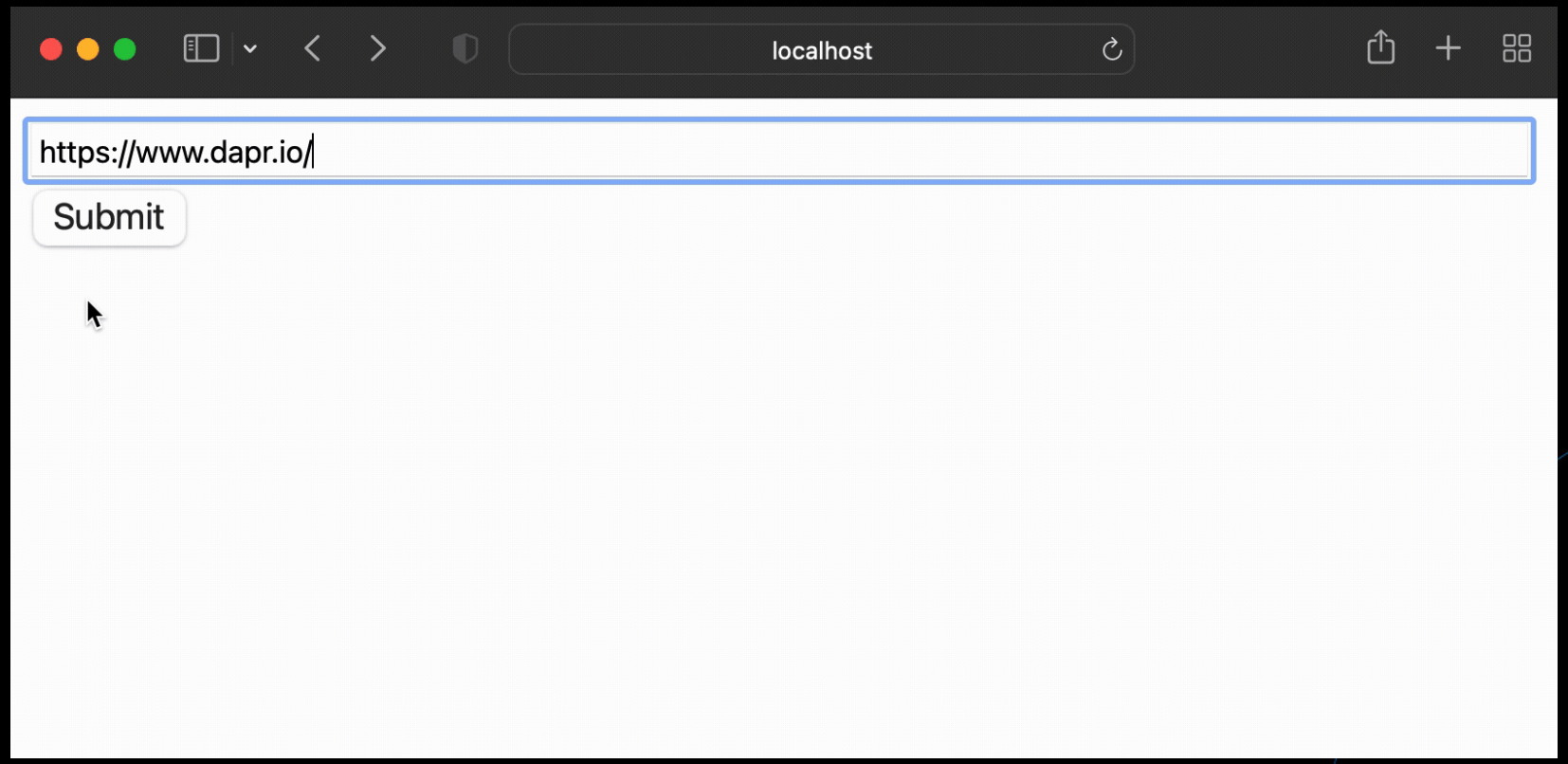
The Distributed Application Runtime (Dapr) provides APIs that simplify microservice connectivity.

Any language, any framework, anywhere



What are we building?

URL Shortener



https://github.com/artursouza/dapr-day-2022

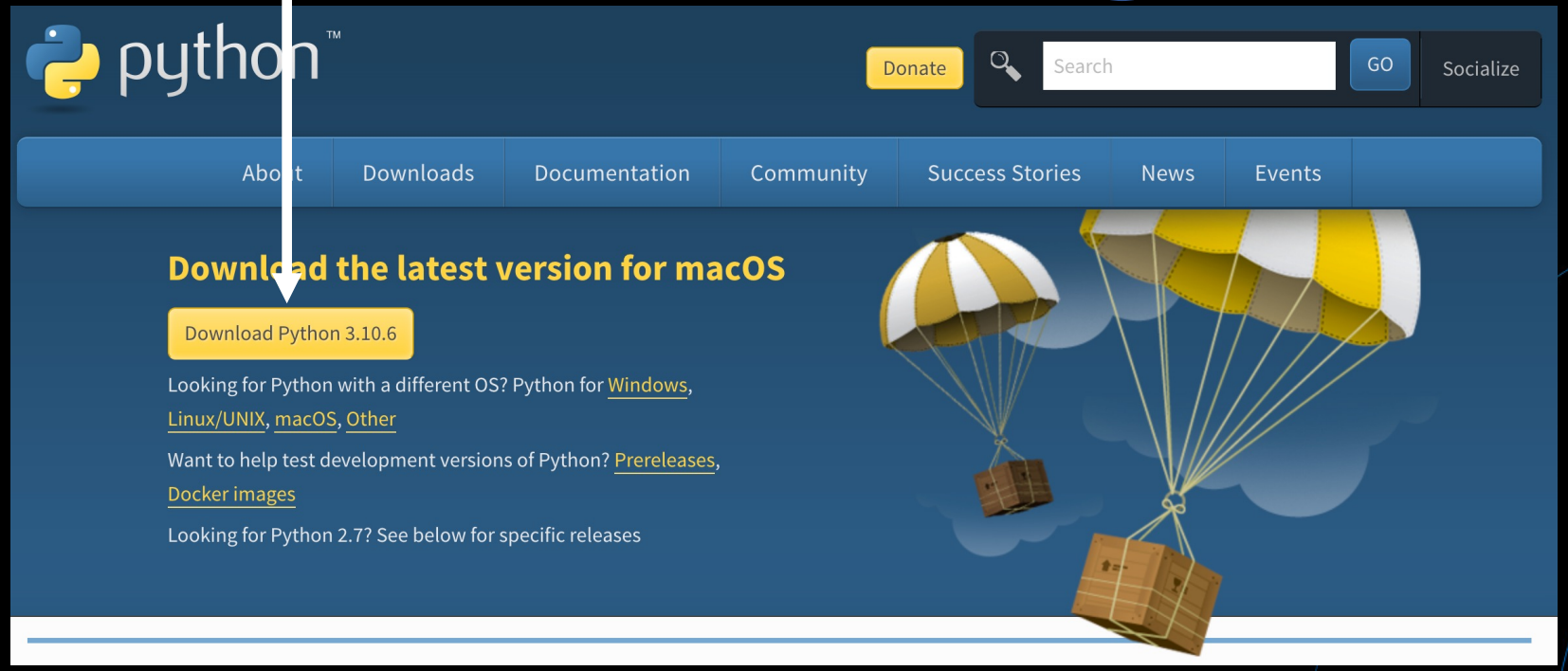
Local Development

From zero to hero

Pre-Requisite for Dapr CLI

Additional pre-requisite for this exercise

! Install version 3.10.x including `pip`



Install Dapr CLI

Linux

wget -q https://spo.ms/dapr-install.sh -O - | /bin/bash

Windows

powershell -Command "iwr -useb https://spo.ms/dapr-install.ps1 | iex"

MacOS

curl -fsSL https://spo.ms/dapr-install.sh | /bin/bash

HomeBrew – MacOS and Linux

brew install dapr/tap/dapr-cli

https://docs.dapr.io/getting-started/install-dapr-cli

```
[$ dapr

      __
 ____/ /___ _____  _____
/ __  / __ '/ __ \/ ___/
/ /_/ / /_/ / /_/ / /
\__,_/\__,_/ .___/_/
          /_/

==============================
Distributed Application Runtime

Usage:
  dapr [command]

Available Commands:
  annotate       Add dapr annotations to a Kubernetes configuration. Supported platforms: Kubernetes
  build-info     Print build info of Dapr CLI and runtime
  completion     Generates shell completion scripts
  components     List all Dapr components. Supported platforms: Kubernetes
  configurations List all Dapr configurations. Supported platforms: Kubernetes
  dashboard      Start Dapr dashboard. Supported platforms: Kubernetes and self-hosted
  help           Help about any command
  init           Install Dapr on supported hosting platforms. Supported platforms: Kubernetes and self-hosted
  invoke         Invoke a method on a given Dapr application. Supported platforms: Self-hosted
  list           List all Dapr instances. Supported platforms: Kubernetes and self-hosted
  logs           Get Dapr sidecar logs for an application. Supported platforms: Kubernetes
  mtls           Check if mTLS is enabled. Supported platforms: Kubernetes
  publish        Publish a pub-sub event. Supported platforms: Self-hosted
  run            Run Dapr and (optionally) your application side by side. Supported platforms: Self-hosted
  status         Show the health status of Dapr services. Supported platforms: Kubernetes
  stop           Stop Dapr instances and their associated apps. Supported platforms: Self-hosted
  uninstall      Uninstall Dapr runtime. Supported platforms: Kubernetes and self-hosted
  upgrade        Upgrades or downgrades a Dapr control plane installation in a cluster. Supported platforms: Kubernetes
  version        Print Dapr runtime and Cli version.

Flags:
  -h, --help          help for dapr
      --log-as-json   Log output in JSON format
  -v, --version       version for dapr

Use "dapr [command] --help" for more information about a command.
```

Initialize

```
$ dapr --version
CLI version: 1.8.1
Runtime version: n/a
```

1

Check the CLI version installed.
Runtime should not be installed yet.

```
$ export DAPR_DEFAULT_IMAGE_REGISTRY=GHCR
```

2

Optionally, use GitHub Container Registry to pull Docker images.

```
$ dapr init
⌛  Making the jump to hyperspace...
ℹ️  Container images will be pulled from Dapr GitHub container registry
ℹ️  Installing runtime version 1.8.4
↙  Downloading binaries and setting up components...
Dapr runtime installed to /Users/asouza/.dapr/bin, you may run the following to add it to your path if you want to run daprd directly:
    export PATH=$PATH:/Users/asouza/.dapr/bin
✅  Downloading binaries and setting up components...
✅  Downloaded binaries and completed components set up.
ℹ️  daprd binary has been installed to /Users/asouza/.dapr/bin.
ℹ️  dapr_placement container is running.
ℹ️  dapr_redis container is running.
ℹ️  dapr_zipkin container is running.
ℹ️  Use `docker ps` to check running containers.
✅  Success! Dapr is up and running. To get started, go here: https://aka.ms/dapr-getting-started
```

3

Now, run `dapr init` to install Dapr's runtime, control plane components, Redis and Zipkin

Installing WebApp dependencies

Create file `requirements.txt`

```
Flask==2.2.2
dapr==1.7.0
```

1 Dapr's Python SDK

Install dependencies

```
pip3 install -r requirements.txt
```

Creating main.py file – basics

```
import random

from dapr.clients import DaprClient
from flask import Flask, request, redirect


SHORT_URL_ID_LENGTH=7

DAPR_STATE_STORE="url-store"

app = Flask(__name__)
dapr = DaprClient()
```

1 Importing for Dapr SDK

2 Name of Dapr's State Store for this application – REMEMBER THIS!

3 Instantiate reusable Dapr Client

Creating main.py file – landing page

```
@app.route("/")
def index():
    return """<form action="" method="POST">
                <input type="text" name="url" size="80">
                <input type="submit" value="Submit">
              </form>"""
```

Creating main.py file – shortening URL

Save State 4

```python
@app.route("/", methods=["POST"])
def shorten():
    """Creates a shortened URL from a full URL."""
    url=request.form["url"]
    if not url:
        return redirect("/")
    short_id=generate_random_short_id()
    print("Setting", short_id, "=>", url, "...", flush=True)
    dapr.save_state(store_name=DAPR_STATE_STORE, key=short_id, value=url)
    print("Set", short_id, "=>", url, flush=True)
    short_url = request.root_url + short_id
    return f"<a href='{short_url}'>{short_url}</a>"
```

Creating main.py file – generating short Id

```
def generate_random_short_id():
    return "".join(map(
        lambda i:chr(i),
        random.choices(range(65,91),
        k=SHORT_URL_ID_LENGTH)))
```

Creating main.py file – resolving short URL

Fetch State 5

```python
@app.route("/<string:short_id>")
def resolve(short_id):
    """Redirects to URL."""
    print("Resolving", short_id, "...", flush=True)
    state = dapr.get_state(store_name=DAPR_STATE_STORE, key=short_id)
    url = state.data.decode()
    print("Resolved", short_id, "=>", url, flush=True)
    return redirect(url)
```

Creating main.py file – start the HTTP server

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080)
```

Creating url_store component

```
$ mkdir components
```

1

Create a folder for components of this application

```
$ vi components/url_store.yml
```

2

Create an YAML file inside the new directory

Component's name must match the name used in the code - did you remember?

```
DAPR_STATE_STORE="url-store"
```

3

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: url-store
spec:
  type: state.redis
  version: v1
  metadata:
  - name: redisHost
    value: localhost:6379
  - name: redisPassword
    value: ""
```

Running the app

App Id for Dapr

1

2

Components folder

3

Command to run the application

```
$ dapr run --app-id=url_shortener --components-path=components/ --  python3 main.py
ℹ️ Starting Dapr with id url_shortener. HTTP Port: 50534. gRPC Port: 50535
ℹ️ Checking if Dapr sidecar is listening on HTTP port 50534
INFO[0000] starting Dapr Runtime -- version 1.8.4 -- commit 18575823c74318c811d6cd61
ance=Arturs-MacBook-Pro.local scope=dapr.runtime type=log ver=1.8.4
INFO[0000] log level set to: info                        app_id=url_shortener instar
ime type=log ver=1.8.4
INFO[0000] metrics server started on :50536/             app_id=url_shortener instar
ics type=log ver=1.8.4
INFO[0000] standalone mode configured                    app_id=url_shortener instar
ime type=log ver=1.8.4
INFO[0000] app id: url_shortener                         app_id=url_shortener instar
ime type=log ver=1.8.4
INFO[0000] mTLS is disabled. Skipping certificate request and tls validation  app_ic
local scope=dapr.runtime type=log ver=1.8.4
INFO[0000] local service entry announced: url_shortener -> 172.16.224.25:50540  app_
o.local scope=dapr.contrib type=log ver=1.8.4
INFO[0000] Initialized name resolution to mdns           app_id=url_shortener instar
```

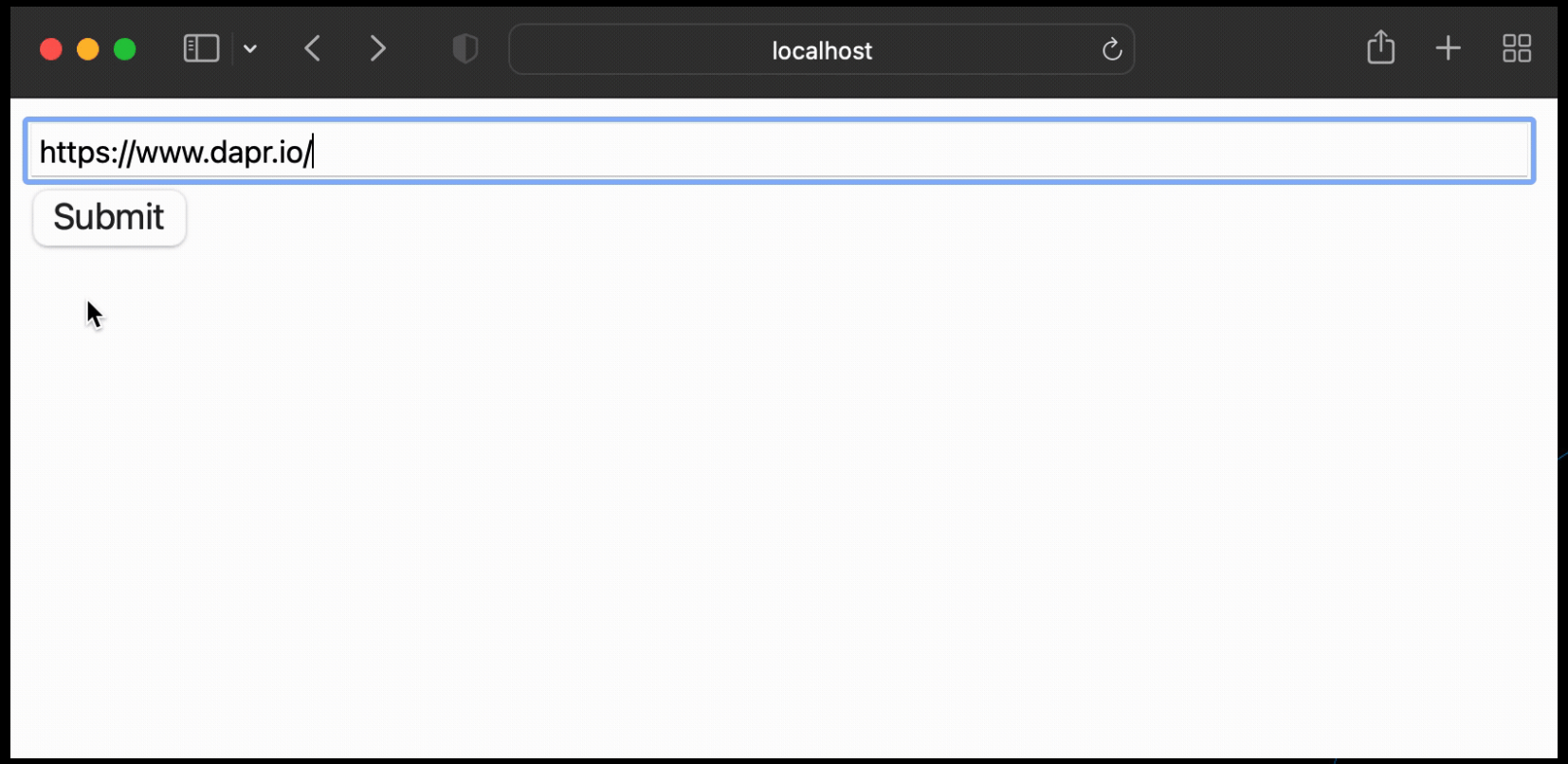
4

Randomly picked ports for Dapr sidecar – those are exposed to the app via environment variables: DAPR_HTTP_PORT and DAPR_GRPC_PORT

Running the app - continuation

```
ℹ️ Checking if Dapr sidecar is listening on GRPC port 50535
ℹ️ Dapr sidecar is up and running.
ℹ️ Updating metadata for app command: python3 main.py
✅  You're up and running! Both Dapr and your app logs will appear here.

== APP ==  * Serving Flask app 'main'
== APP ==  * Debug mode: off
== APP == WARNING: This is a development server. Do not use it in a prodı
.
== APP ==  * Running on http://127.0.0.1:8080
== APP == Press CTRL+C to quit
```



https://github.com/artursouza/dapr-day-2022

Adding Resiliency

- Timeout
- Retries
- Circuit-Breaking

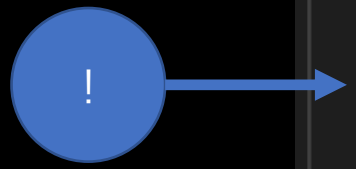
We will add retries when writing to the state store

Adding Resiliency – enabling the feature

```
$ vi ~/.dapr/config.yaml
```

```yaml
apiVersion: dapr.io/v1alpha1
kind: Configuration
metadata:
  name: daprConfig
spec:
  tracing:
    samplingRate: "1"
    zipkin:
      endpointAddress: http://localhost:9411/api/v2/spans
  features:
    - name: Resiliency
      enabled: true
```

Enabling Resiliency, since it is a preview feature



Adding Resiliency

```
$ vi components/resiliency.yaml
```

```yaml
apiVersion: dapr.io/v1alpha1
kind: Resiliency
metadata:
  name: url-resiliency
spec:
  policies:
    retries:
      cosmosdbRetry:
        policy: exponential
        maxInterval: 15s
        maxRetries: 10
  targets:
    components:
      url-store:
        outbound:
          retry: cosmosdbRetry
```

1 Any unique name

2 Policy as a template

3 Target is where the policy is applied. It can be a component, actor type or invoked service

Homework

How to detect duplicate when saving to state store?

The "short Id" is randomly generated but there is no guarantee of uniqueness. Dapr can help with that via the "eTag" feature and fail to save the state if there is an existing key. Change the code to make use of that feature.

https://docs.dapr.io/developing-applications/building-blocks/state-management/state-management-overview#concurrency

Simulate a Redis outage and observe the retry policy being applied.

Run the app again after enabling the Resiliency feature and creating the Resiliency policy file in the same folder as the "url-store" component. After the app starts successfully, stop the Redis container and try to create a new short URL – you should see the page hang. At this point, quickly start the Redis container again and the request will complete successfully.

https://docs.dapr.io/operations/resiliency/resiliency-overview/

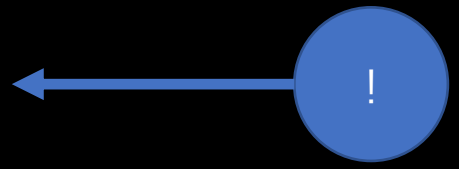
Deploying to Kubernetes

Pre-Requisites

- Docker
- Dapr CLI
- A Kubernetes Cluster
- kubectl command line utility
- CosmosDB account

Creating Dockerfile , building and pushing image

```
FROM python:3.10-alpine
WORKDIR /webapp
ADD requirements.txt /webapp
RUN pip install -r requirements.txt
ADD main.py /webapp
EXPOSE 8080
CMD ["python","main.py"]
```



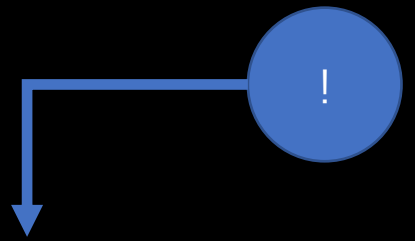
Dockerfile - no Dapr here!

```
$ docker build -t artursouza/daprday:edge .
$ docker login
$ docker push artursouza/daprday:edge
```

Installing Dapr on Kubernetes Cluster

```
$ dapr init -k
⌛  Making the jump to hyperspace...
ℹ️  Note: To install Dapr using Helm, see here: https://docs.dapr.io/getting-started/
install-dapr-kubernetes/#install-with-helm-advanced

ℹ️  Container images will be pulled from Dapr GitHub container registry
✅  Deploying the Dapr control plane to your cluster...
✅  Success! Dapr has been installed to namespace dapr-system. To verify, run `dapr
status -k' in your terminal. To get started, go here: https://aka.ms/dapr-getting-st
arted
```



Kubernetes namespace for Dapr's control plane

```
$ kubectl get pods -n dapr-system
NAME                                 READY   STATUS    RESTARTS   AGE
dapr-dashboard-6b4b6f8579-6sxjl      1/1     Running   0          61s
dapr-operator-755867f4b6-kq8t4       1/1     Running   0          61s
dapr-placement-server-0              1/1     Running   0          61s
dapr-sentry-549b55bdb5-jqtv7         1/1     Running   0          61s
dapr-sidecar-injector-7b8ddf6b7f-hpr4d   1/1     Running   0          61s
```

Configuring CosmosDB state store component on Kubernetes Cluster

```
$ kubectl create secret generic cosmosdb \
>   --from-literal=daprday2022-masterkey=SECRET_GOES_HERE

$ kubectl get secret cosmosdb
NAME       TYPE     DATA   AGE
cosmosdb   Opaque   1      3m25s

$ mkdir azure

$ vi azure/url_store.yml

$ kubectl apply -f azure/url_store.yml
component.dapr.io/url-store created
```

See file content in the next slide.

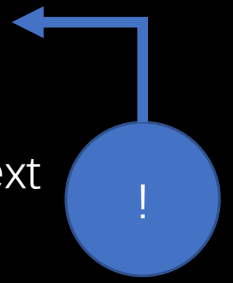
```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: url-store
spec:
  type: state.azure.cosmosdb
  version: v1
  metadata:
  - name: url
    value: https://daprday-2022.documents.azure.com:443/
  - name: masterKey
    secretKeyRef:
      name: cosmosdb
      key:  daprday2022-masterkey
  - name: database
    value: daprday2022
  - name: collection
    value: url_store
```

1 Component name remains the same from local development

2 Reference to secret on Kubernetes

Deploying app to Kubernetes

```
$ mkdir kubernetes
$ vi kubernetes/deployment.yml
$ kubectl apply -f kubernetes/deployment.yml
deployment.apps/url-shortener created
```



See file content in the next slide.

```
$ kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
url-shortener-5bc979599b-6v7fc   2/2     Running   0          50s
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: url-shortener
  labels:
    app: url-shortener
spec:
  replicas: 1
  selector:
    matchLabels:
      app: url-shortener
```

1

Dapr Annotations

```yaml
template:
  metadata:
    labels:
      app: url-shortener
    annotations:
      dapr.io/enabled: "true"
      dapr.io/app-id: "url-shortener"
      dapr.io/app-port: "8080"
      dapr.io/enable-api-logging: "true"
  spec:
    containers:
    - name: webapp
      image: artursouza/daprday:edge
      ports:
      - containerPort: 8080
      imagePullPolicy: Always
```

Deploying ingress to Kubernetes

```
$ vi kubernetes/service.yml
$ kubectl apply -f kubernetes/service.yml
service/url-shortener created
```

See file content in the next slide.

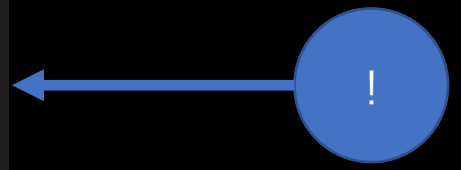
1

2

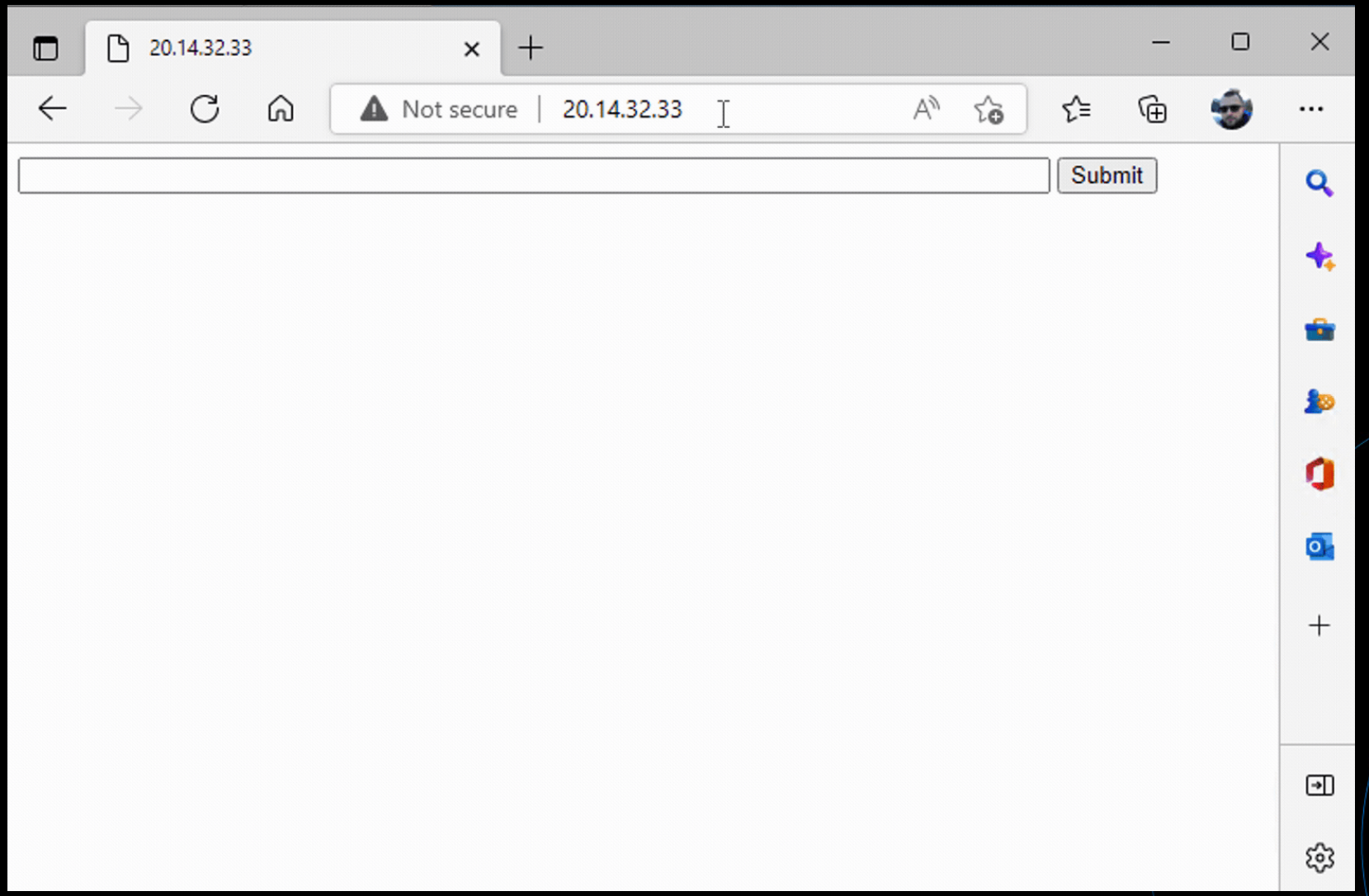
We will invoke http://20.14.32.33/

```
$ kubectl get service
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)
                    AGE
kubernetes          ClusterIP      10.0.0.1       <none>        443/TCP
                    178m
url-shortener       LoadBalancer   10.0.103.185   20.14.32.33   80:32303/TCP
                    57s
url-shortener-dapr  ClusterIP      None           <none>        80/TCP,50001/TCP,
50002/TCP,9090/TCP  100m
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: url-shortener
  labels:
    app: url-shortener
spec:
  selector:
    app: url-shortener
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
  type: LoadBalancer
```

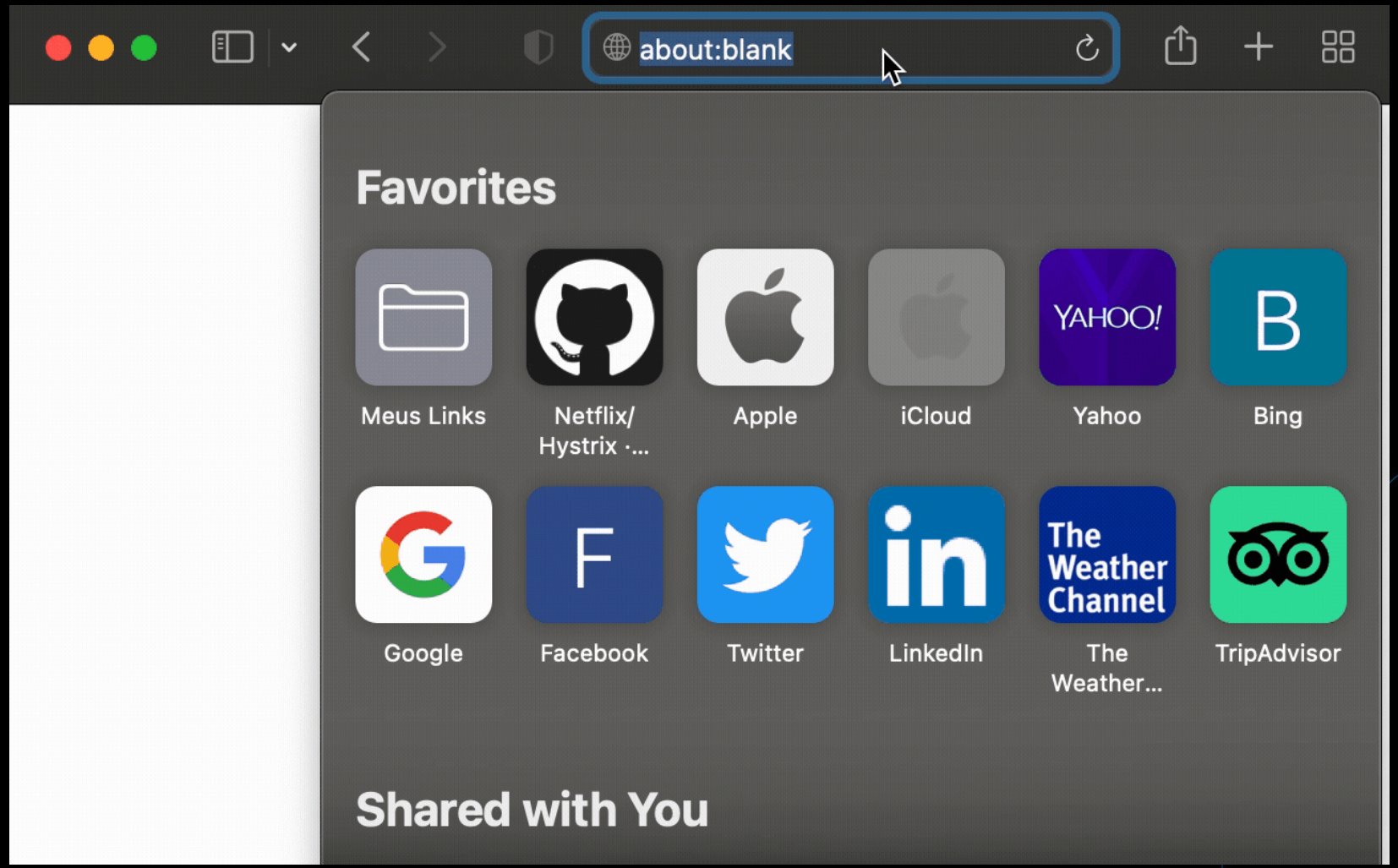


No Dapr here!


20.14.32.33
Not secure
20.14.32.33
Submit

apr Day

Microsoft Azure intel Dapr China Community



apr ay

Microsoft Azure intel Dapr China Community

Homework

Enable Resiliency feature and apply the same Resiliency policy from local development

Create a custom config YAML and apply it to the application's deployment object. Then, apply the same Resiliency object but on the Kubernetes cluster this time. You should now be able to simulate downtime on your state store component and see the application hang when trying to save a new short URL.

https://docs.dapr.io/operations/configuration/preview-features/

https://docs.dapr.io/operations/resiliency/resiliency-overview/

Custom Dapr Installation on Kubernetes via Helm

Fetch Dapr charts

```
$ helm repo add dapr https://dapr.github.io/helm-charts/
$ helm repo update
```

High Availability Mode

```
$ helm upgrade --install dapr dapr/dapr \
  --version=1.8 \
  --namespace dapr-system \
  --create-namespace \
  --set global.ha.enabled=true \
  --wait
```

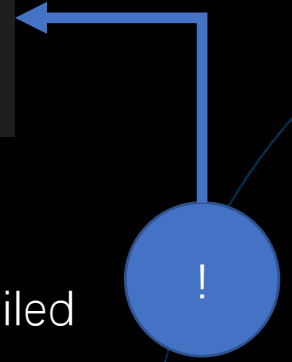
3 replicas of each control plane service

WARNING: if existing installation already exists, placement service must be deleted before moving to HA mode

```
$ k delete -n dapr-system statefulset dapr-placement-server
```

Sidecar Injector WatchDog

```
$ helm upgrade --install dapr dapr/dapr \
  --version=1.8 \
  --namespace dapr-system \
  --create-namespace \
  --set global.ha.enabled=true \
  --set dapr_operator.watchInterval="2m" \
  --wait
```



Kills Kubernetes PODs
where sidecar injector failed

Custom Container Registry

```
$ helm upgrade --install dapr dapr/dapr \
  --version=1.8 \
  --namespace dapr-system \
  --create-namespace \
  --set global.ha.enabled=true \
  --set dapr_operator.watchInterval="2m" \
  --set global.registry="ghcr.io/dapr" \
  --wait
```

Could also be a private registry !

Dapr extension for AKS

Installing Dapr on AKS using the extension

Azure CLI Copy Try It

```
az extension add --name k8s-extension
```

Azure CLI Copy Try It

```
az provider register --namespace Microsoft.KubernetesConfiguration
```

azure-cli Copy

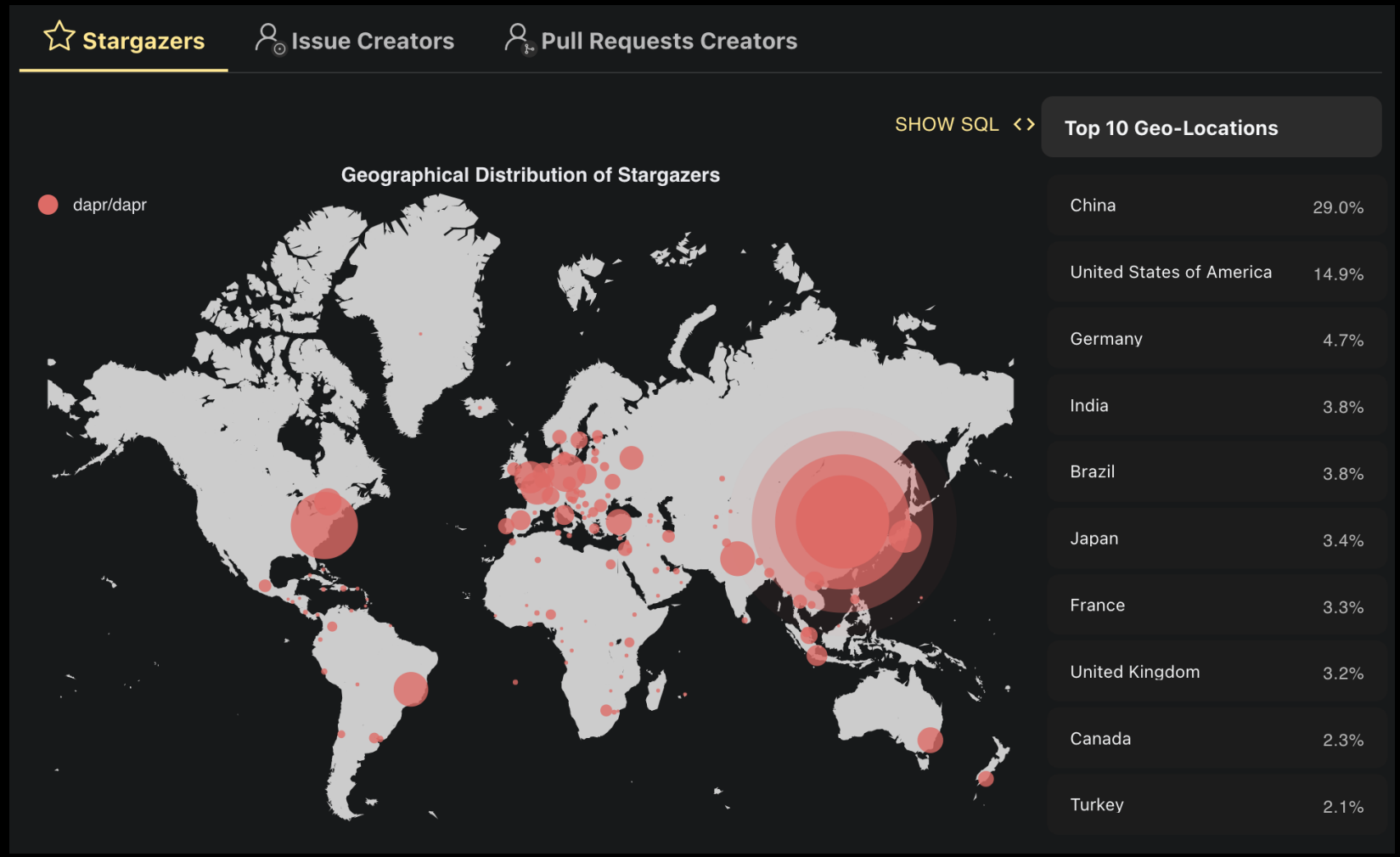
```
az k8s-extension create --cluster-type managedClusters \
--cluster-name myAKSCluster \
--resource-group myResourceGroup \
--name myDaprExtension \
--extension-type Microsoft.Dapr \
--auto-upgrade-minor-version true \
--configuration-settings "global.ha.enabled=true" \
--configuration-settings "dapr_operator.replicaCount=2"
```

!

Offers same configuration options as using Helm

apr ay

Microsoft Azure intel Dapr China Community



https://ossinsight.io/analyze/dapr/dapr


apr ay
云原生运行时开发者日
Thank You
联合主办：
Microsoft Azure
intel
Dapr China Community