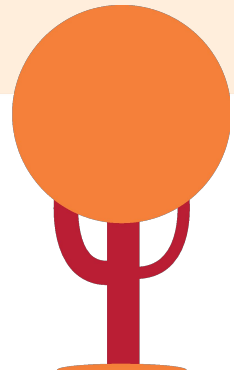
Serverless k8s with Azure Container Apps

Masi Malmi



About me

DevOps Consultant @ Polar Squad

Certified public cloud professional with .NET background

Resident of Málaga since August

Mountaineer at heart

Husband & father

polar squad

Intro

“Microsoft loves Linux.”

Satya Nadella
2014

History of container platforms in Azure

= **Brendan Burns joins MS**

July 2016

Azure Container Service

April 2016

Azure Service Fabric

March 2016

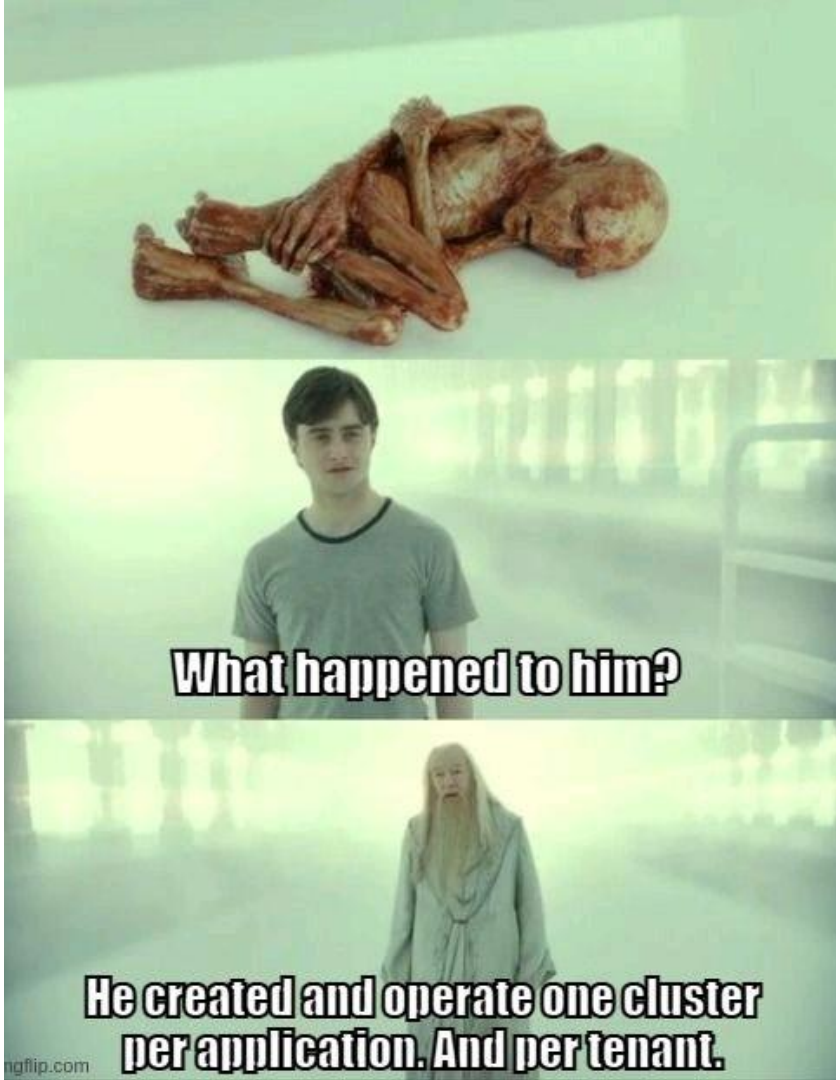
Azure Kubernetes Service (AKS)

June 2018

Azure Container Apps (ACA)

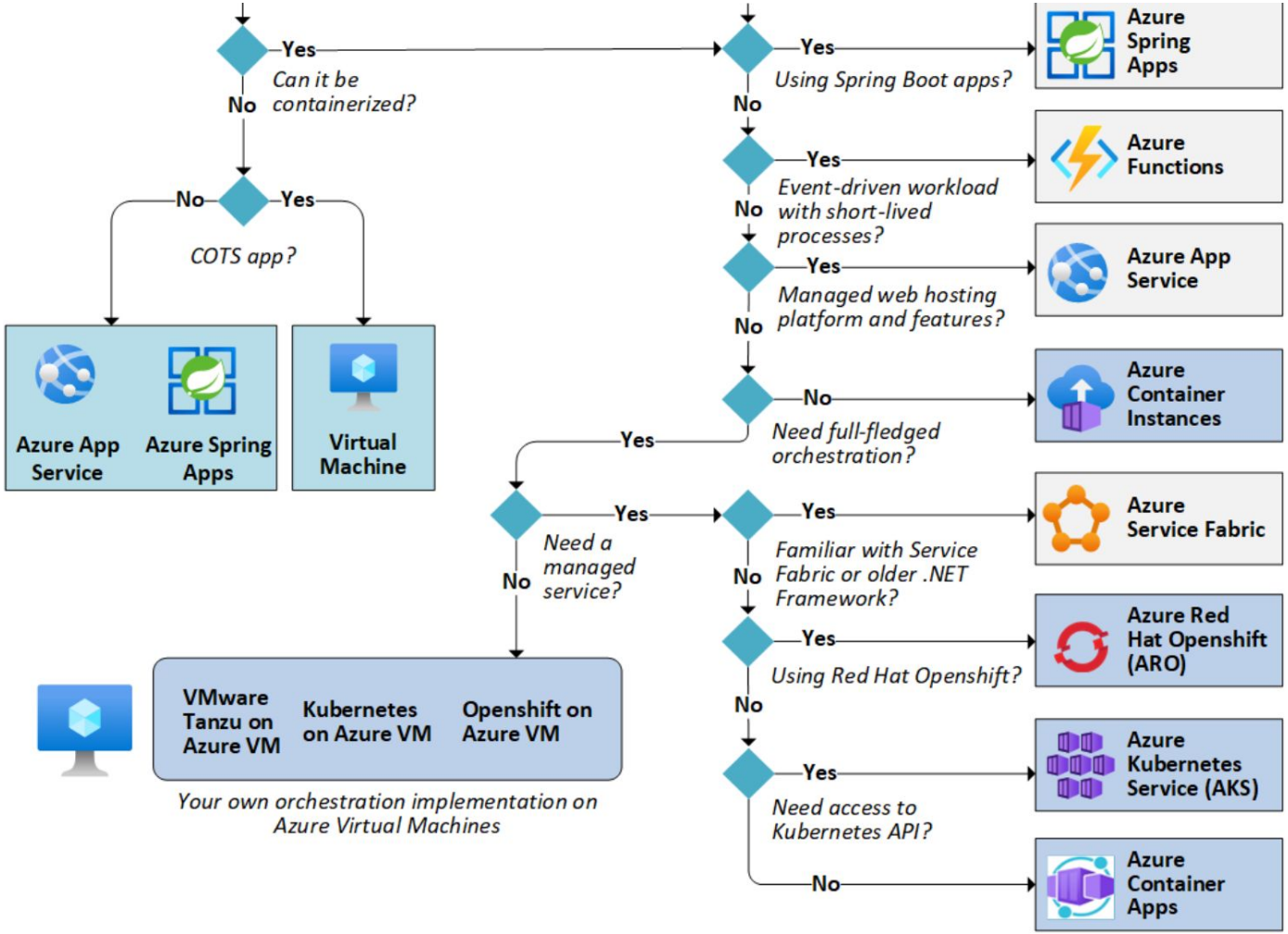
May 2022


AND THIS HOW YOU DEPLOY
A CONTAINER ON KUBERNETES
imgflip.com


What happened to him?
He created and operate one cluster per application. And per tenant.
ngflip.com

Container services in Azure

Azure Compute Services



Azure App Service

A Linux based server side service, that needs characteristics like

- Always on - high availability and auto-healing
- Hybrid connectivity and private networking support
- Authentication and security features out-of-the-box
- Supports also stateful applications

Use cases

- **Mobile app backend**: a GraphQL endpoint
- **A PHP backend**: an admin tool
- **Node.JS or .NET Core application**: a backend for a Single Page Application (SPA)

Azure Functions

Linux based app for processing small tasks

- Triggered based on events
- Scales out quickly
- Programming model portable to other container based platforms (code reuse)

Use cases

- **API endpoints**
- Integration layer between other services: **iPaaS**
- Event-driven **serverless microservices**

Azure Container Instances

A Linux based single workload (one process)

- Scales automatically
- Zero maintenance

Use cases

- **Custom RTMP endpoint** (video streaming)
- Azure DevOps **self-hosted Linux build agent**

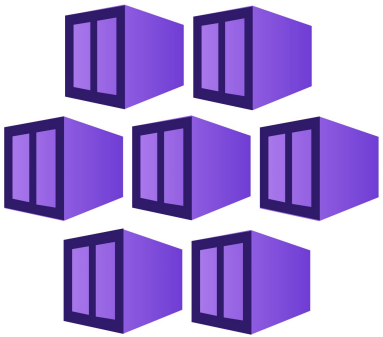
Azure Kubernetes Service (AKS)

Fully managed Kubernetes service

- Integrates with Azure native services
- High availability & CI/CD “out of the box”
- Requires knowledge and know-how of Kubernetes, Linux and Open Source projects
- Responsibility over what happens inside the cluster after initial creation

Use cases

- **SaaS solution** based on microservices architecture
- Self-hosting a **3rd party product**



There was no real option for hosting fully fledged microservices architecture in Azure without diving deep into Kubernetes or Service Fabric internals.

Azure Container Apps

Fully-managed serverless abstraction on top of Kubernetes infrastructure, purpose built for managing and scaling event-driven microservices with a consumption-based pricing model.

Platform capabilities powered by CNCF projects

Envoy: managed ingress and traffic splitting

Kubernetes Event Driven Autoscaler (KEDA): managed, event-driven autoscale

Distributed Application Runtime (Dapr): managed, APIs that simplify microservice connectivity*

* optional

Core features

Revisions: immutable snapshot representative of a specific version of a container app

Health probes: Based on Kubernetes health probes with support for Readiness, Liveness and Startup

Built-in authentication methods

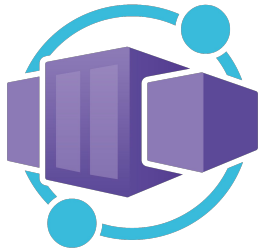
Managed Identities

Custom domain names and certificates

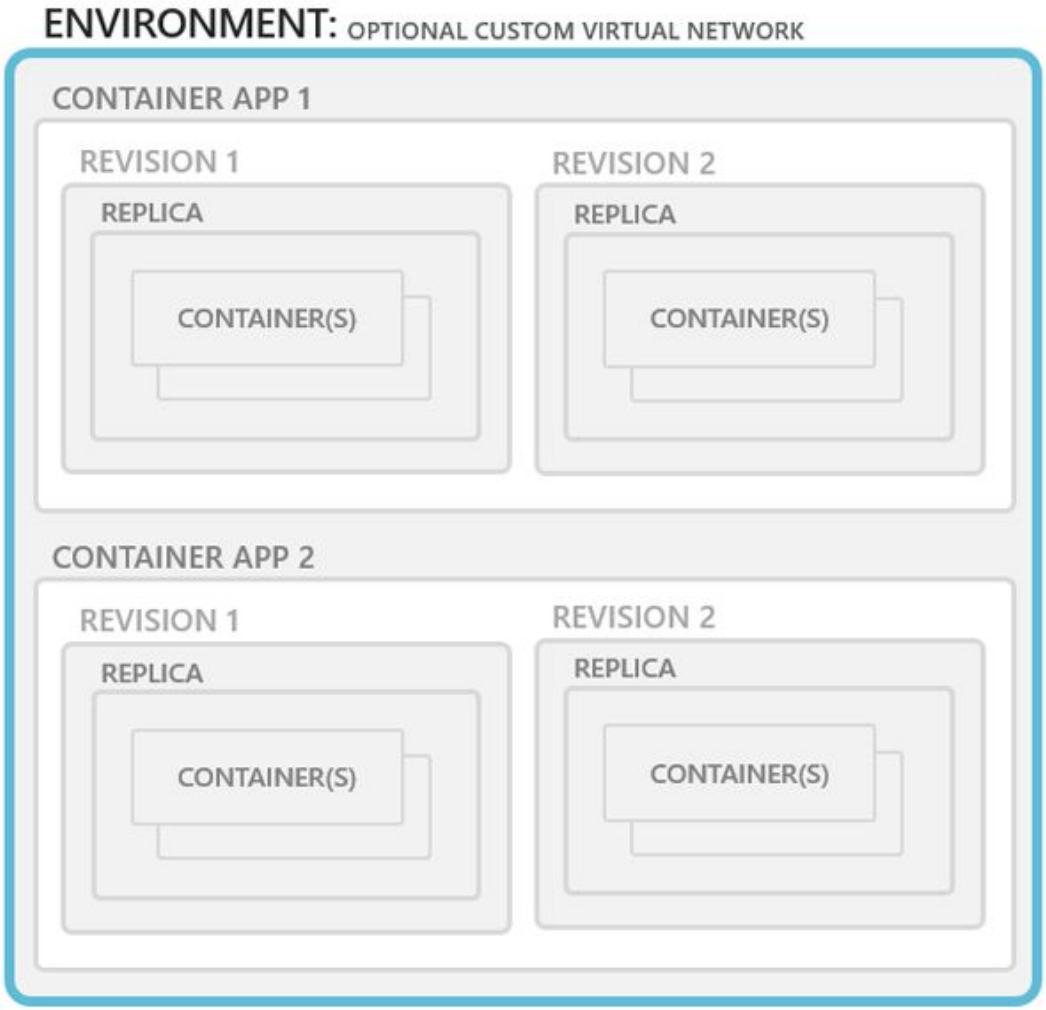
Virtual Network: managed or custom

Platform observability: log streaming, console connect, Azure monitor

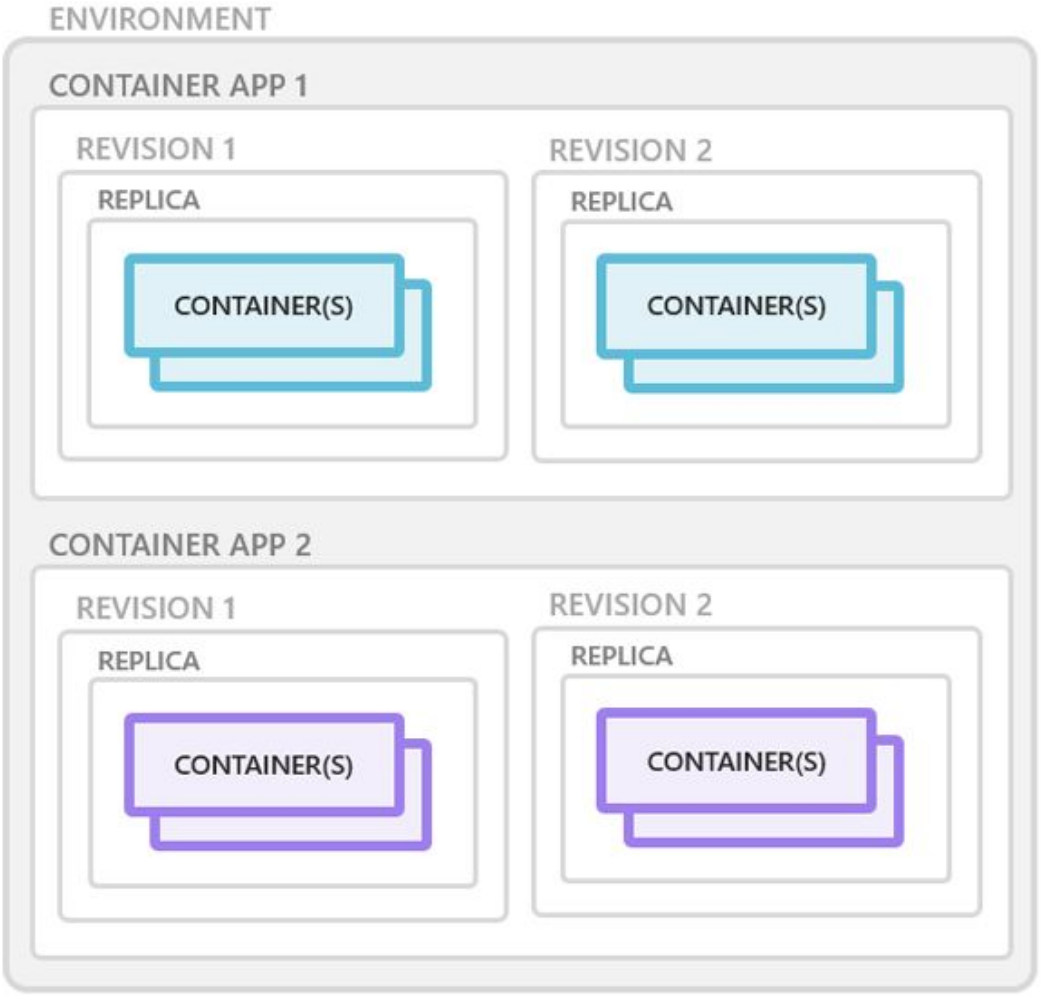
Secrets



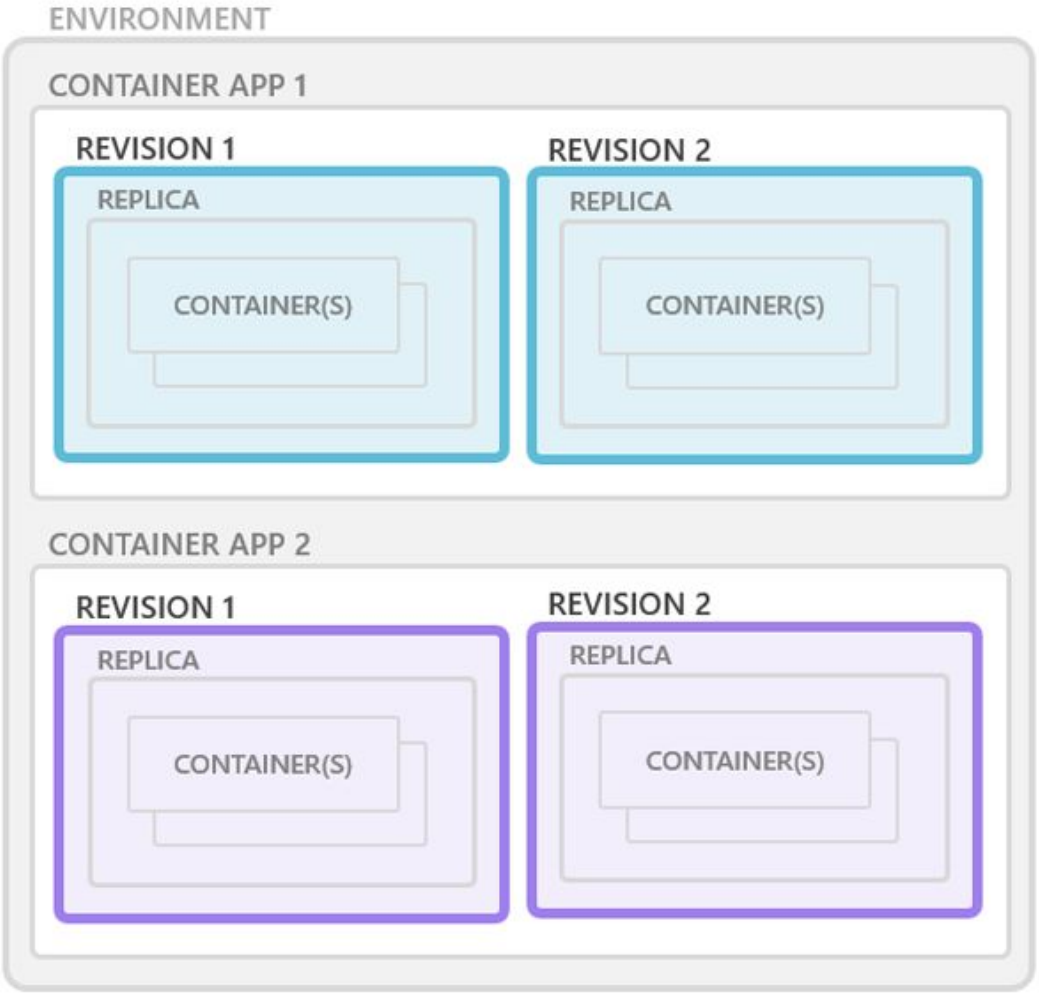
Environments are an isolation boundary around a collection of container apps.

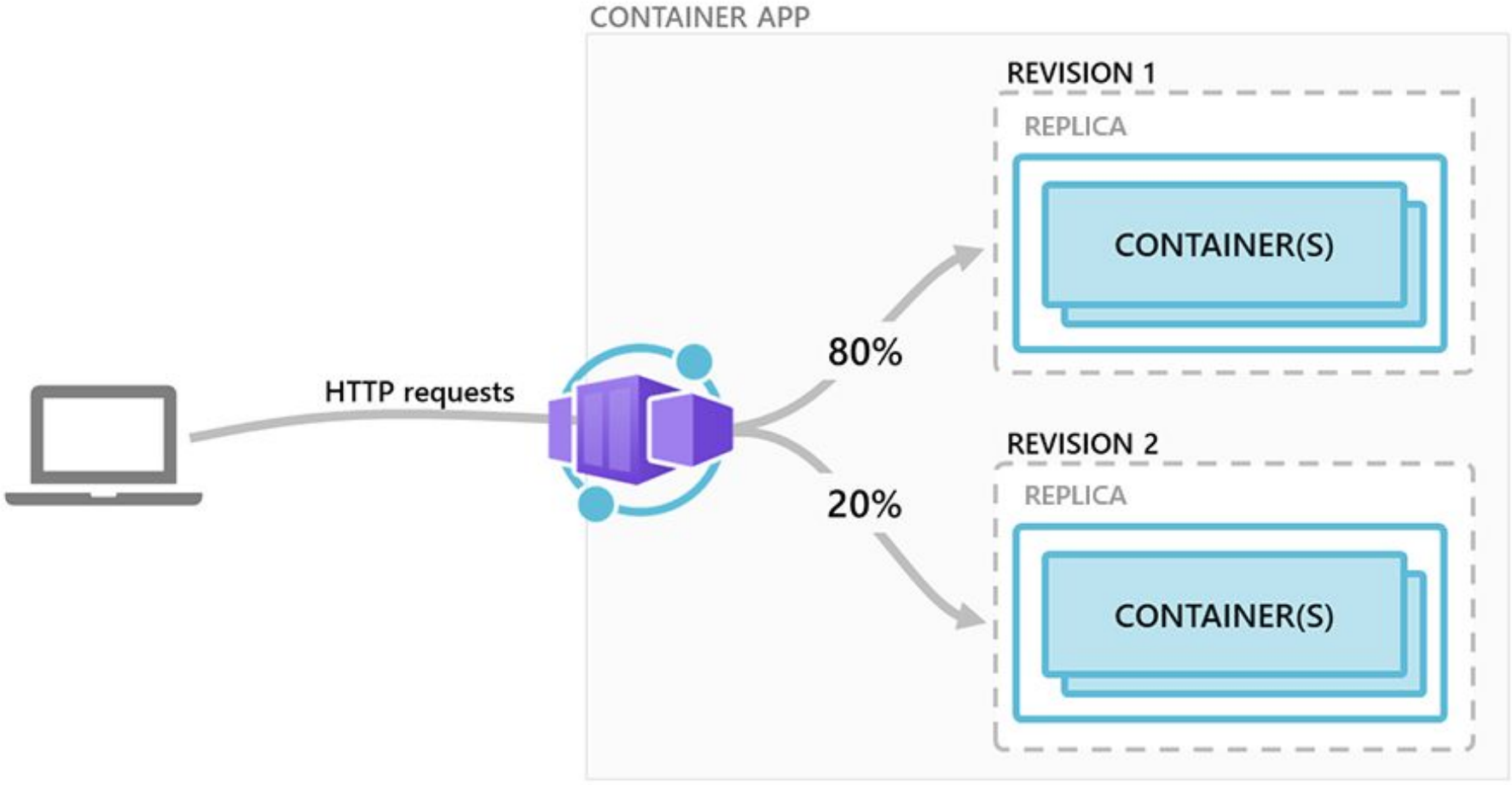


Containers for an Azure Container App are grouped together in pods inside revision snapshots.



Revisions are immutable snapshots of a container app.

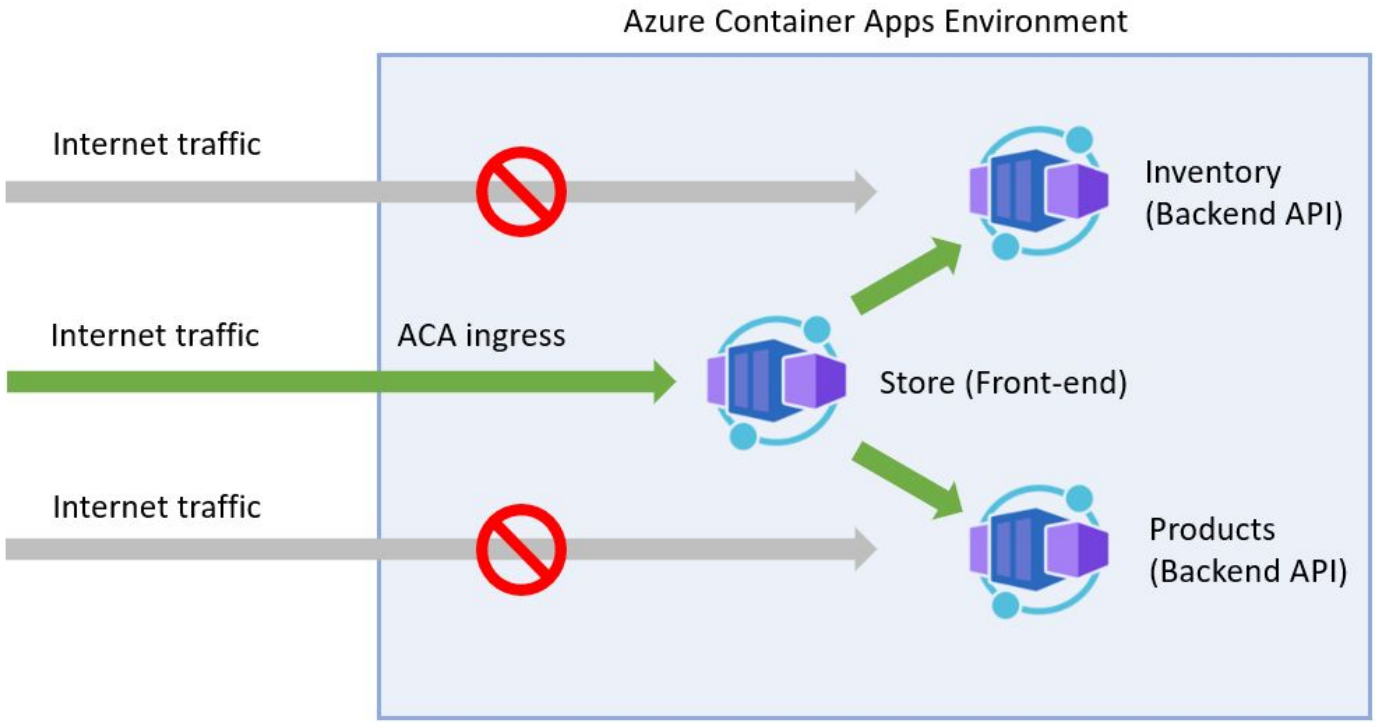



CONTAINER APP
REVISION 1
REPLICA
CONTAINER(S)
HTTP requests
80%
20%
REVISION 2
REPLICA
CONTAINER(S)

CONTAINER APP ADDRESS

`https://myapp.happyhill-70162bb9.eastus2.azurecontainerapps.io`

1. Container app name
2. Environment unique identifier
3. Region name


Azure Container Apps Environment
Internet traffic
Internet traffic
ACA ingress
Internet traffic
Inventory
(Backend API)
Store (Front-end)
Products
(Backend API)

Supported scale triggers

HTTP traffic: Scaling based on the number of concurrent HTTP requests to your revision.

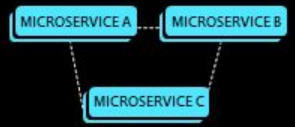
TCP traffic: Scaling based on the number of concurrent TCP requests to your revision.

Event-driven: Event-based triggers such as messages in an Azure Service Bus.

CPU or Memory usage: Scaling based on the amount of CPU or memory consumed by a replica.

What can you build with Serverless in Cloud-native?

Microservices



Microservices architecture with the option to integrate with Dapr

Public API endpoints

E.g., API app with HTTP requests split between two revisions of the app

Web Apps

E.g., Web app with custom domain, TLS certificates, and integrated authentication

Event-driven processing

E.g., Queue reader app that processes messages as they arrive in a queue

Background processing

E.g., Continuously running background process transforms data in a database

AUTO-SCALE CRITERIA

Individual microservices can scale independently using any KEDA scale triggers

Scaling is determined by the number of concurrent HTTP requests

Scaling is determined by the number of concurrent HTTP requests

Scaling is determined by the number of messages in the queue

Scaling is determined by the level of CPU or memory load

What about security?

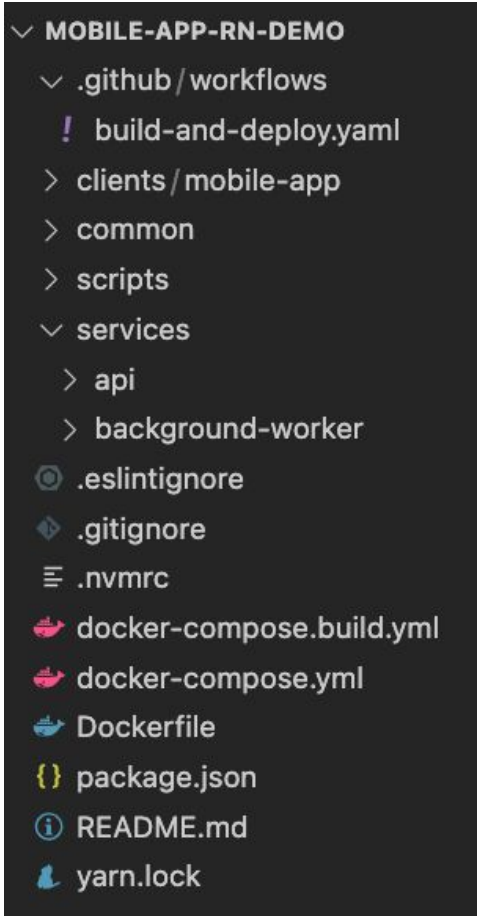
Application layer

Docker images

Azure networking (optional)

Identities

DEMO



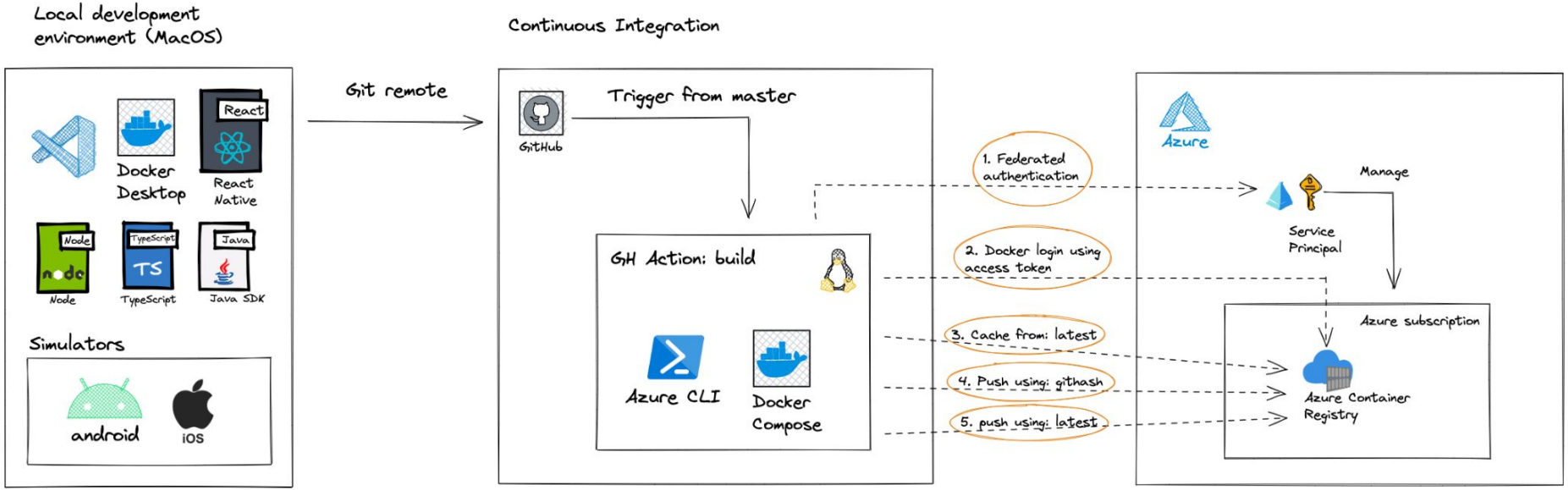
Github Action

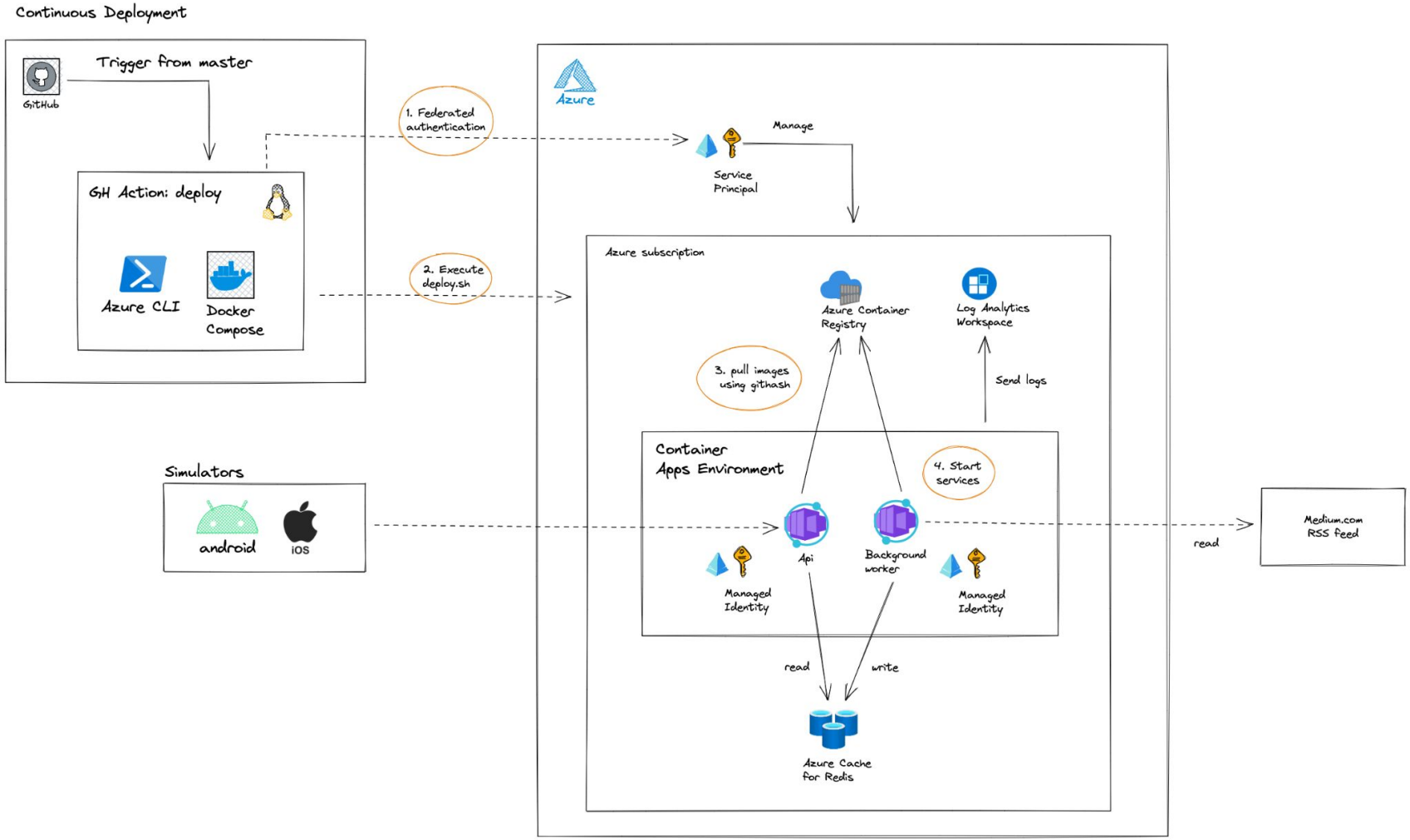
Mobile app

Bash scripts

Common library and two services

Docker & Docker Compose files


Local development environment (MacOS)
Docker Desktop
React
React Native
Node
Node
TypeScript
TS
TypeScript
Java
Java SDK
Simulators
android
iOS
Git remote
Continuous Integration
GitHub
Trigger from master
GH Action: build
Azure CLI
Docker Compose
1. Federated authentication
2. Docker login using access token
3. Cache from: latest
4. Push using: githash
5. push using: latest
Azure
Manage
Service Principal
Azure subscription
Azure Container Registry


Continuous Deployment
Trigger from master
GitHub
GH Action: deploy
Azure CLI
Docker Compose
1. Federated authentication
2. Execute deploy.sh
Azure
Manage
Service Principal
Azure subscription
Azure Container Registry
Log Analytics Workspace
Send logs
3. pull images using githash
Container Apps Environment
4. Start services
Api
Background worker
Managed Identity
Managed Identity
read
write
Azure Cache for Redis
Simulators
android
iOS
read
Medium.com RSS feed

api

```
az containerapp create \
--name $CONTAINER_APP_EXTERNAL_NAME \
--resource-group $RESOURCE_GROUP \
--environment $ENVIRONMENT \
--image $ACR_NAME.azurecr.io/$EXTERNAL_SERVICE_NAME:$IMAGE_TAG \
--target-port 3000 \
--ingress 'external' \
--cpu 0.5 \
--memory 1.0Gi \
--min-replicas 1 \
--max-replicas 6 \
--scale-rule-name my-http-rule \
--scale-rule-http-concurrency 10 \
--user-assigned $API_IDENTITY_ID \
--registry-identity $API_IDENTITY_ID \
--secrets redispwd=$REDIS_PWD \
--env-vars REDIS_HOST=$REDIS_HOST REDIS_PORT=6379 REDIS_PW=secretref:redispwd \
--registry-server $ACR_NAME.azurecr.io
```

background-worker

```
az containerapp create \
--name $CONTAINER_APP_INTERNAL_NAME \
--resource-group $RESOURCE_GROUP \
--environment $ENVIRONMENT \
--image $ACR_NAME.azurecr.io/$INTERNAL_SERVICE_NAME:$IMAGE_TAG \
--target-port 3002 \
--ingress 'internal' \
--cpu 0.25 \
--memory 0.5Gi \
--min-replicas 1 \
--max-replicas 1 \
--user-assigned $WORKER_IDENTITY_ID \
--registry-identity $WORKER_IDENTITY_ID \
--secrets redispwd=$REDIS_PWD \
--env-vars REDIS_HOST=$REDIS_HOST REDIS_PORT=6379 REDIS_PW=secretref:redispwd CRON_JOB_SCHEDULE="*/5 * * * *" \
--registry-server $ACR_NAME.azurecr.io
```

RECAP

New serverless service in Azure for cloud-native, container based workloads

Powered by Kubernetes and fully-managed

Best suited for small teams and/or application workloads which do not require all the bells and whistles of Kubernetes

App centric. Deliver faster.


ANY QUESTIONS?

Come say hi!


WUN
DER
DOG
We're Hiring!
Check open positions at
wunderdog.fi/hola
Future-proofing your
digital transformation
polar squad
The Best DevOps Company*
Join the Squad
—
polarsquad.com/careers-malaga
*According to us

Verisk™

kamstrup

WUNDERDOG

TOKIOTA

LEMONCODE

METADEV

plain concepts

Ortus Solutions
A Software Revolution