May 5th – 7th, 2022

#GlobalAzure

May 7th, 2022

organized by

DEV CAFE

Connect and share

#Vietnam

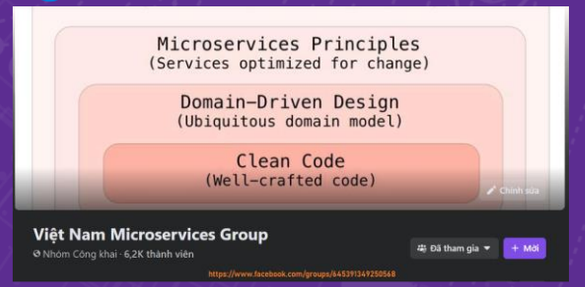
Building Applications Fast and Scalable with Azure Container Apps

Thang Chung
May 2022

Thang Chung

Sr Solution Architect

- Work at NashTech Vietnam
- Microsoft MVP
- Experience: ~15 years in the software development area, in both outsourcing and product companies
- Creator of Vietnam Microservices Group on Facebook
- Hobbies:
  - Research and deep dive into new things in the software development world
  - Read books, drink beer, and contemplate the nature of my life
  - Spend many hours per day at https://github.com/thangchung and https://www.facebook.com/groups/645391349250568



Agenda

- Cloud-native Maturity Model
- The Fifteen Factor App and Microservices Architectural Patterns
- Multiple Runtime Microservices Architecture problems
- Azure Container Apps: what, why, where, when?
- Demo: CoolStore Application on ACA

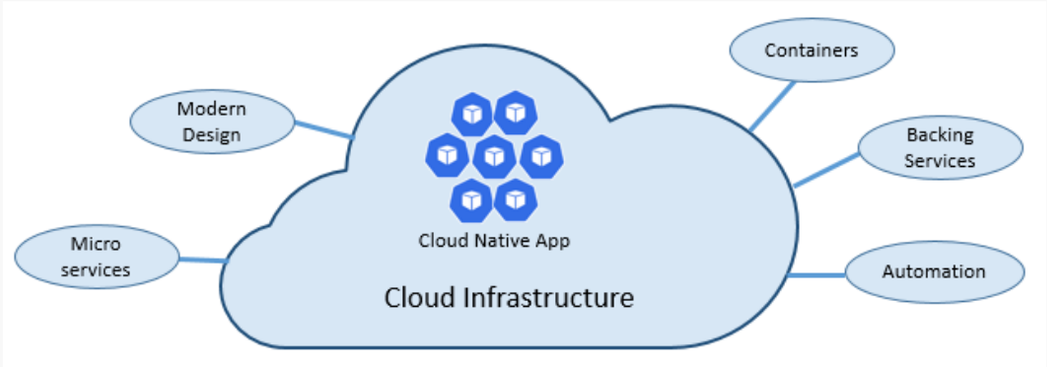
What is Cloud-native? And why do we need it?

“By cloud, we mean any computing environment in which **computing**, **networking**, and **storage** resources can be provisioned and released **elastically** in an **on-demand**, **self-service** manner.”

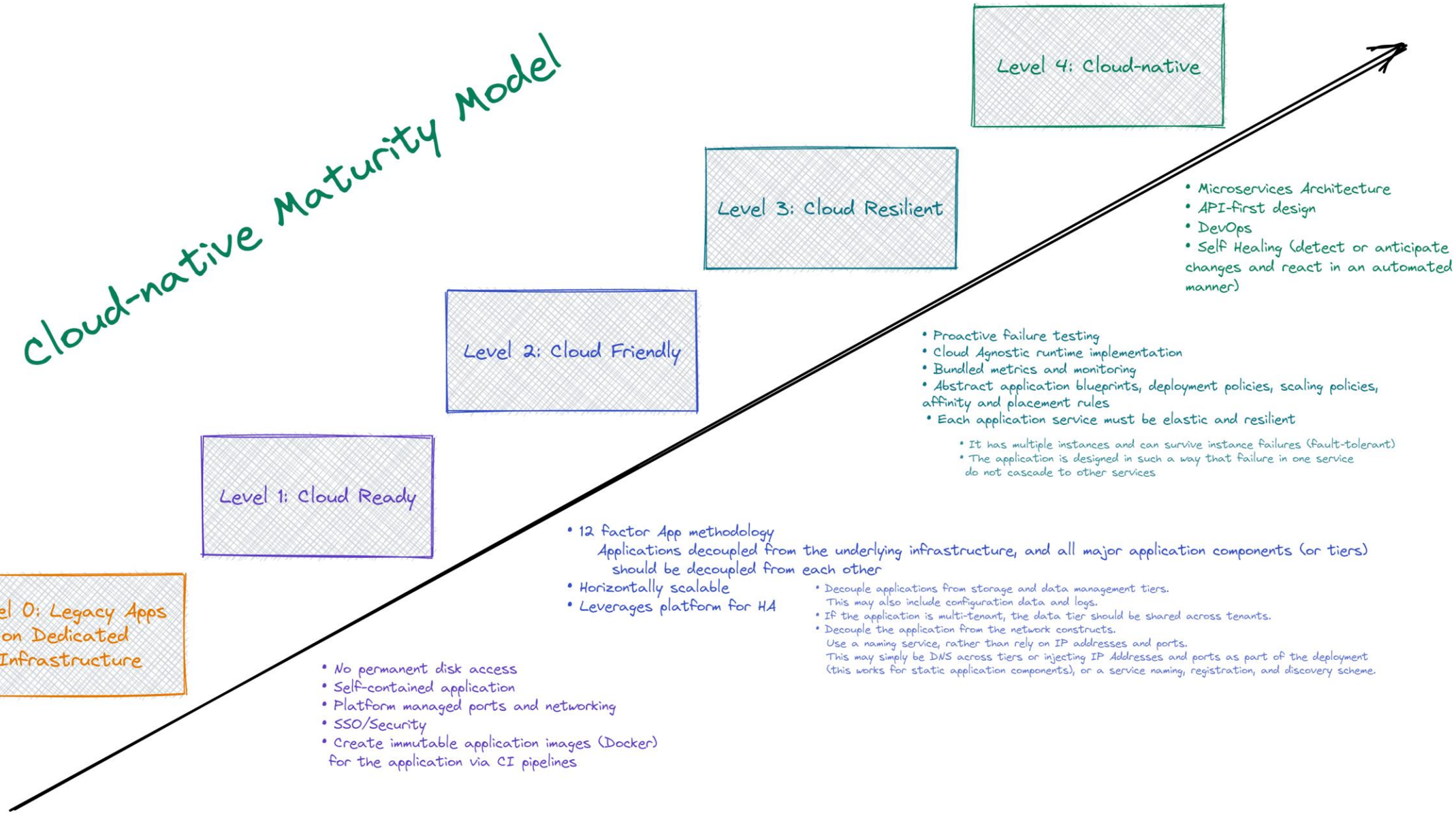
Migrating to Cloud-Native Application Architectures, Matt Stine

And by Cloud-native, we mean:

- An application that has been designed to run in such an environment and allow such an environment to be **exploited** to the maximum
- Cloud-native technologies empower organizations to build and run **scalable** applications in **modern**, **dynamic** environments such as **public**, **private**, and **hybrid** clouds. **Containers**, **service meshes**, **microservices**, **immutable** infrastructure, and **declarative** APIs exemplify this approach



Cloud-native Maturity Model



Credits: HCL Tech and Ian Cooper

@thangchung

The Fifteen Factor App

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

IV. Backing services

Treat backing services as attached resources

V. Build, Release, Run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin Processes

Run admin/management tasks as one-off processes

XIII. API First

Strong API-first contracts follow DDD to provide consistent and reusable components in collaboration teams

XIV. Telemetry

Provide standards for APM, domain specific metrics, health + system logs

XV. Security

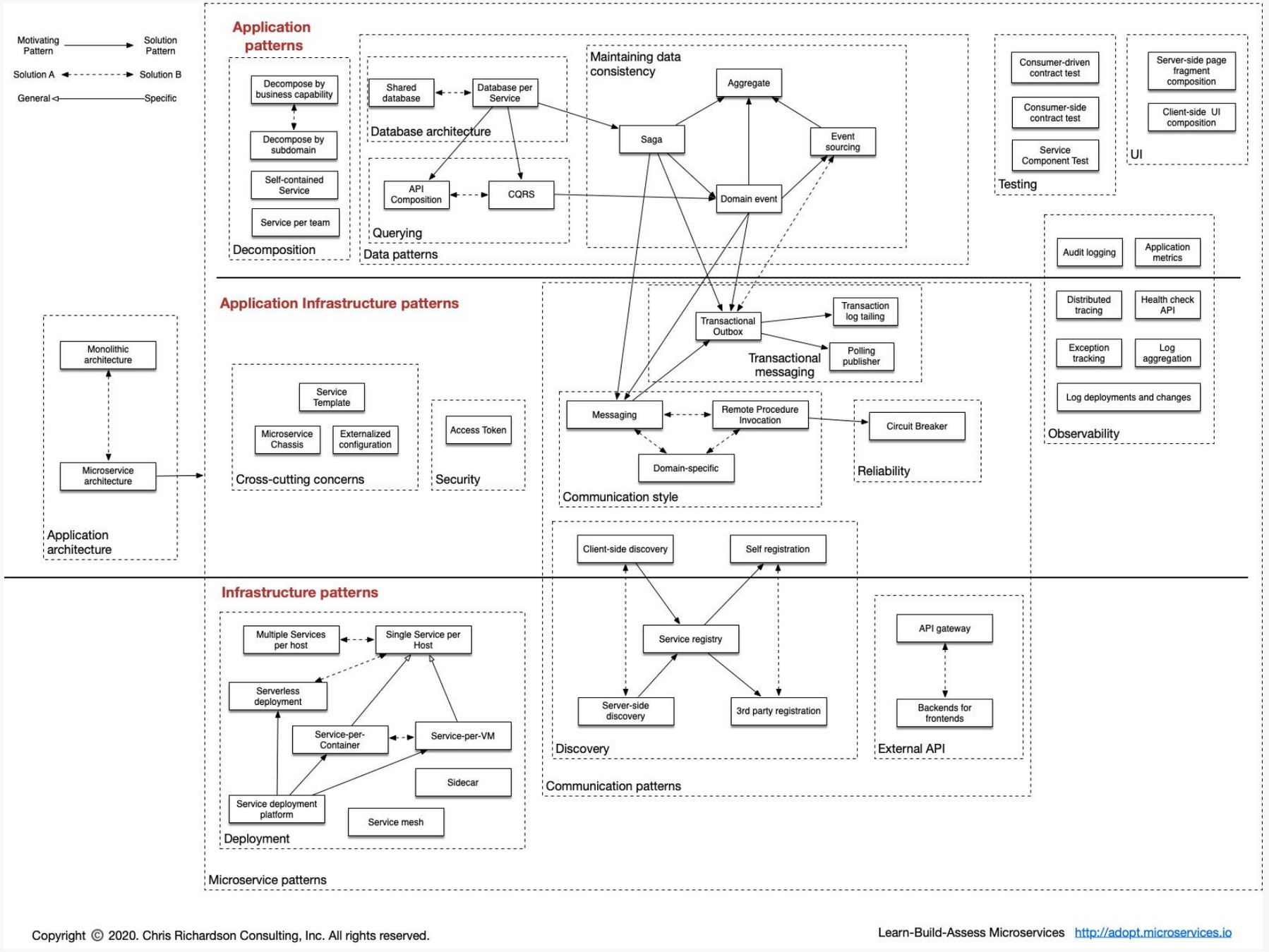
Provide Authn and Authz aspects with (m)TLS, OAuth, OIDC, JWTs, Identity Providers, API Gateways, and container patching

https://12factor.net

https://tinyurl.com/4mhyzcfn

@thangchung

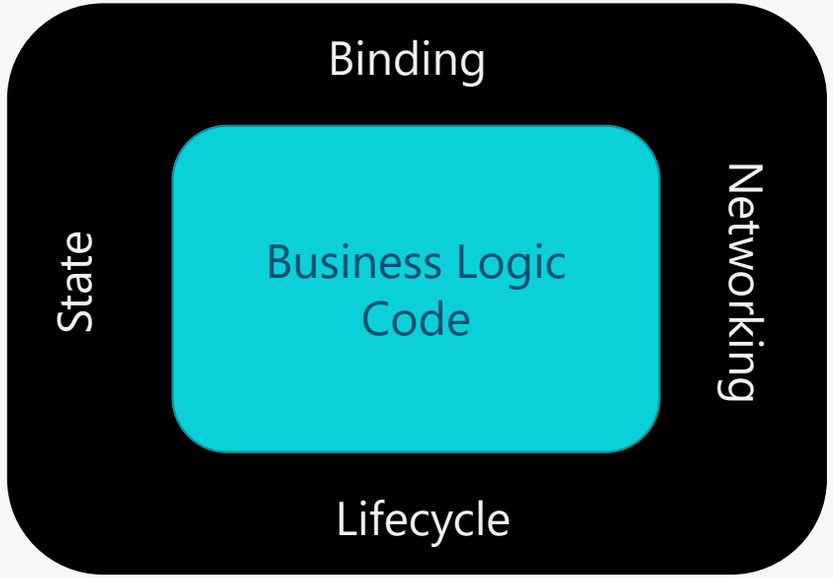
Apply Microservice Architectural Patterns



Learn-Build-Assess Microservices http://adopt.microservices.io

Multiple Runtime Microservices Architecture

- Connectors
- Protocol conversion
- Message transformation
- Message routing
- Transactionality



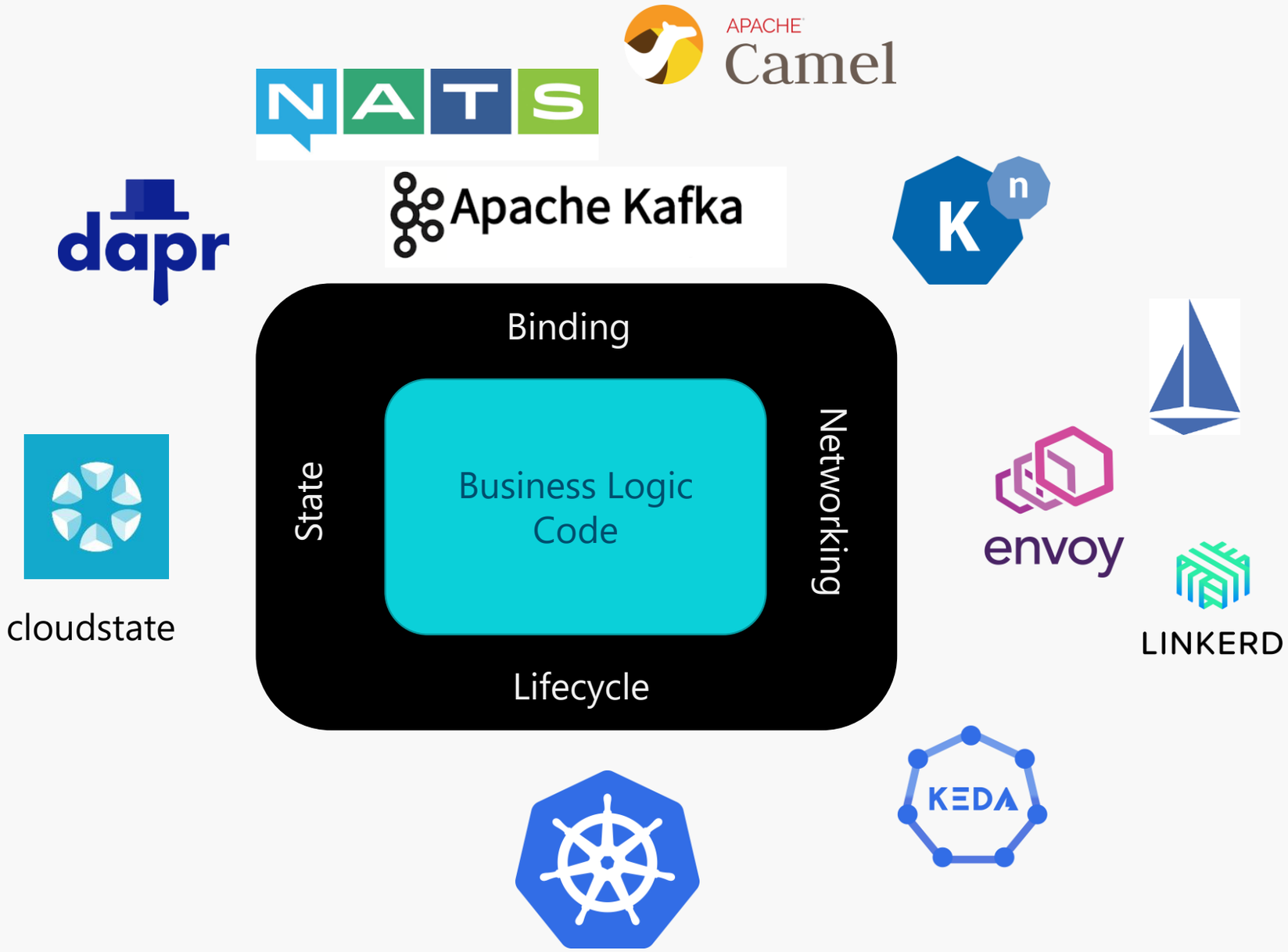
- Workflow management
- Idempotency
- Temporary scheduling
- Caching
- Application state

- Service Discovery
- A/B testing, canary rollouts
- Retry, timeout, circuit breaker
- Point-to-point, pub/sub
- Security
- Observability

- Packaging
- Healthcheck
- Deployment
- Scaling
- Configuration, secrets

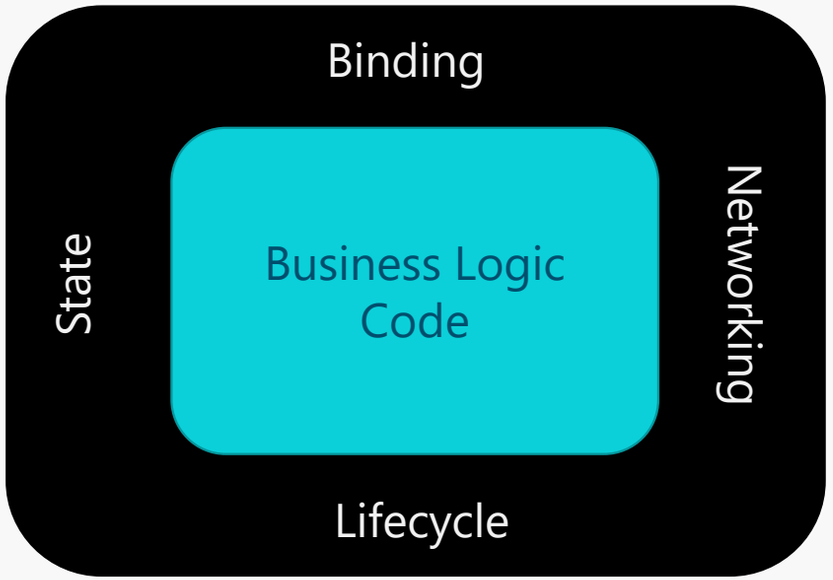
Credit to Bilgin Ibryam

Multiple Tooling



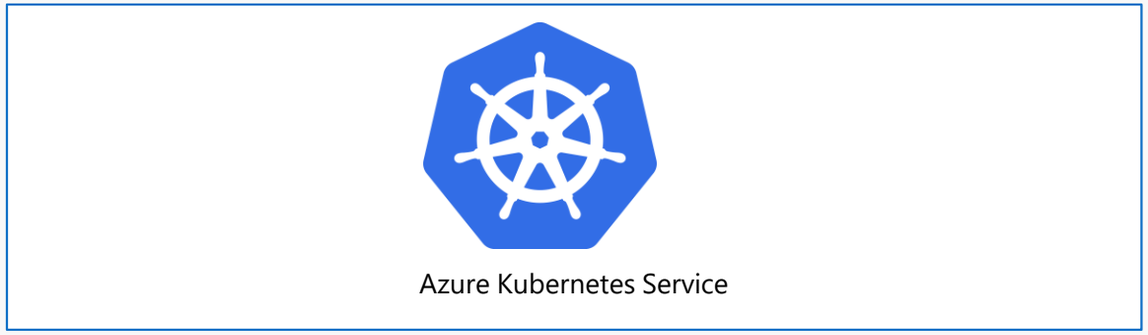
Credit to Bilgin Ibryam

Multiple Tooling

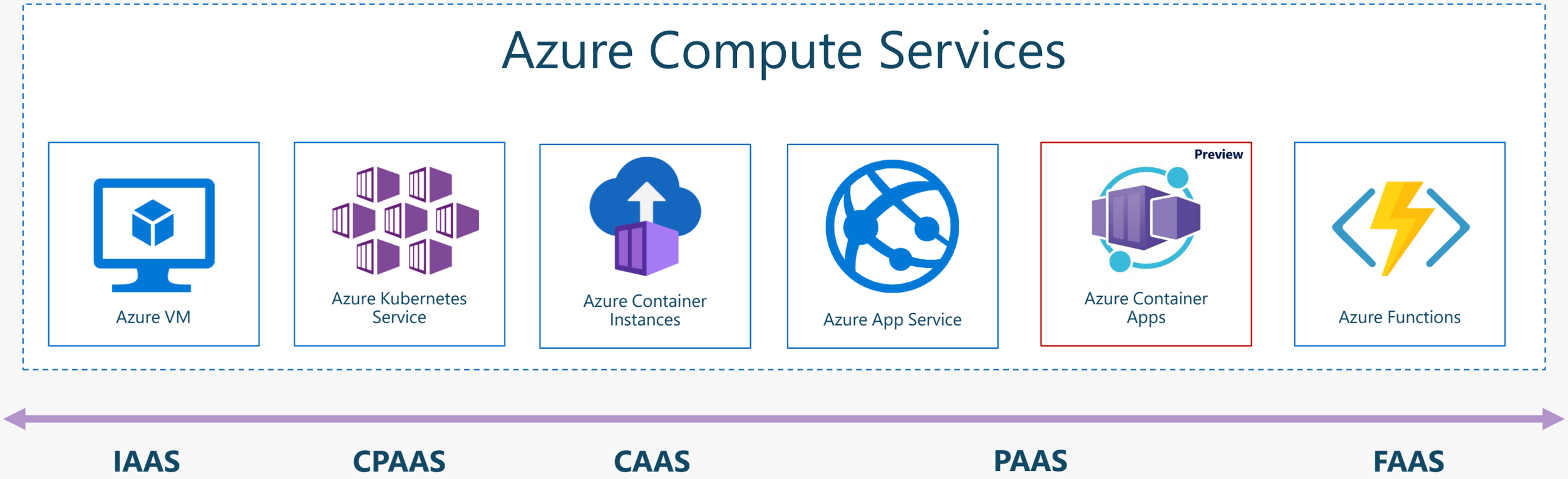


Credit to Bilgin Ibryam

Azure Container Apps



Container is everywhere? And where's ACA?



ACA: Use cases

Azure Container Apps: Example scenarios

PUBLIC API ENDPOINTS

HTTP requests are split between two versions of the container app where the first revision gets 80% of the traffic, while a new revision receives the remaining 20%.

AUTO-SCALE CRITERIA

Scaling is determined by the number of concurrent HTTP requests.

BACKGROUND PROCESSING

A continuously-running background process that transforms data in a database.

AUTO-SCALE CRITERIA

Scaling is determined by the level of CPU or memory load.

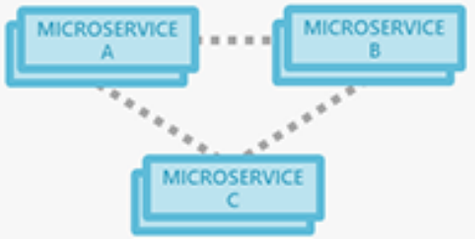
EVENT-DRIVEN PROCESSING

A queue reader application that processes messages as they arrive in a queue.

AUTO-SCALE CRITERIA

Scaling is determined by the number of messages in the queue.

MICROSERVICES



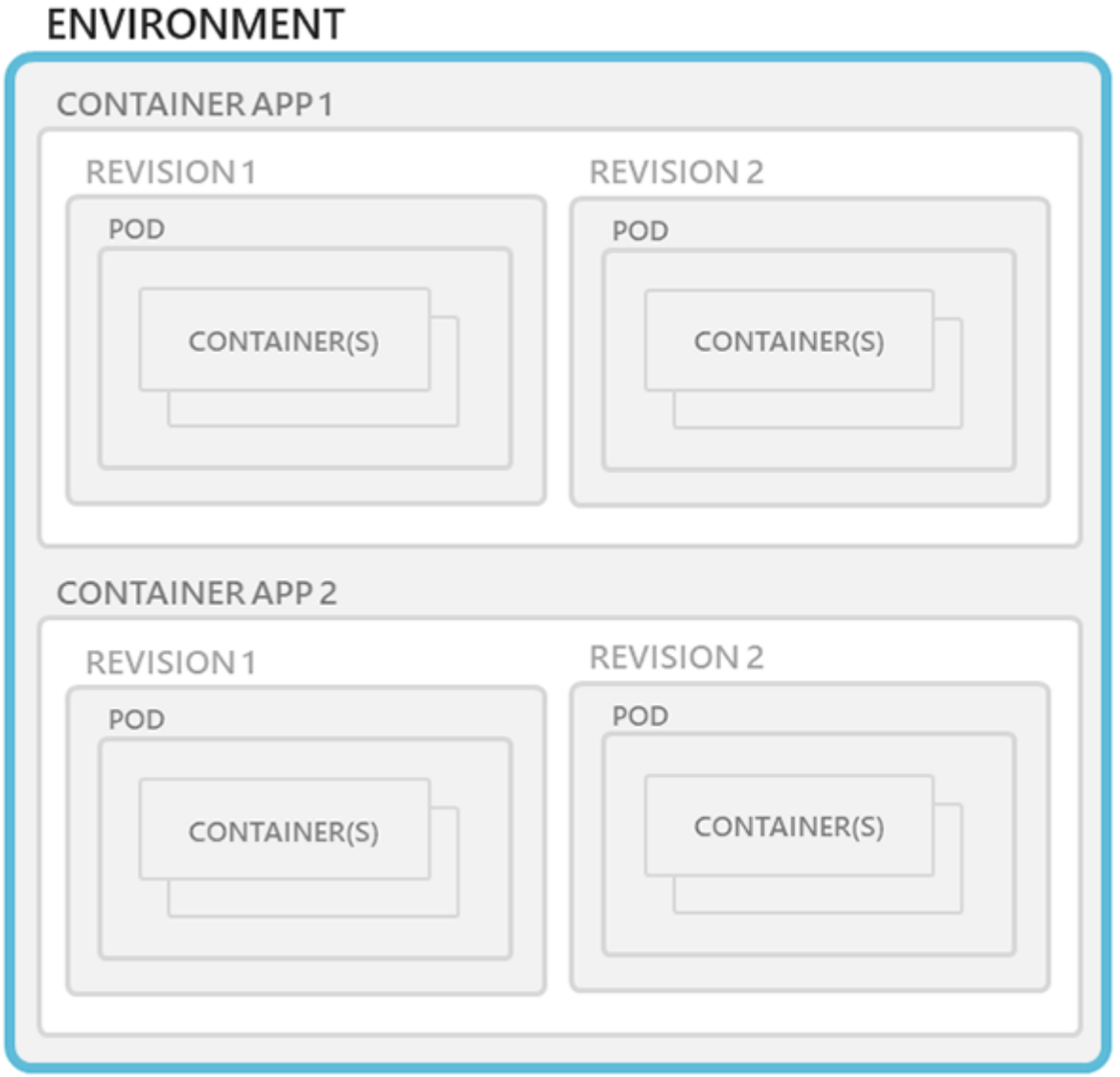
Deploy and manage a microservices architecture with the option to integrate with Dapr.

AUTO-SCALE CRITERIA

Individual microservices can scale according to any KEDA scale triggers.

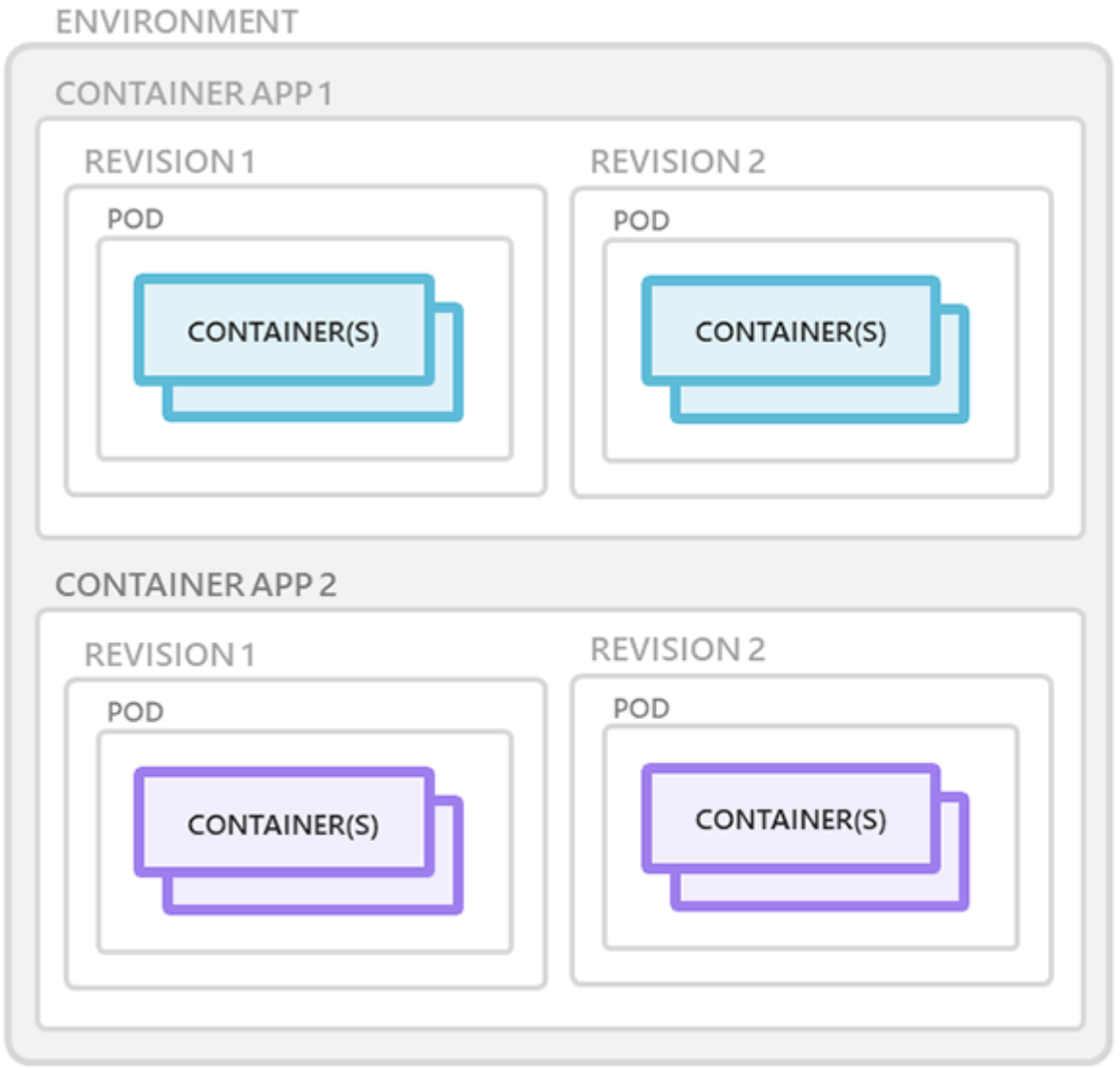
ACA- Environments

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



ACA- Containers

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

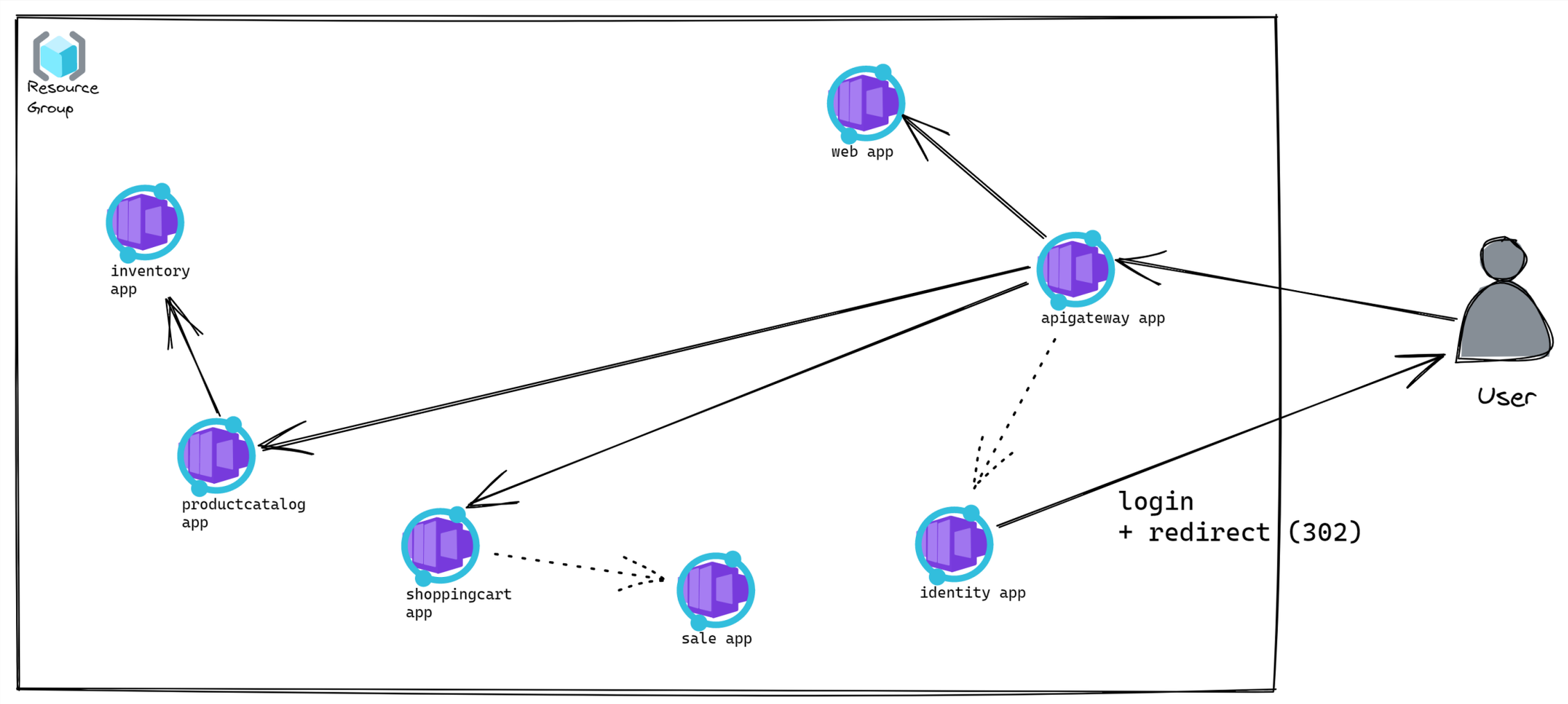


#Vietnam

DEMO: CoolStore Application on Azure Container Apps

https://github.com/vietnam-devs/coolstore-microservices

DEMO: High Level Architecture



https://github.com/vietnam-devs/coolstore-microservices

ACA: Pricing

Region: Southeast Asia

Currency: United States – Dollar ($) USD

Resource consumption

Container Apps are billed based on resource allocation measured in vCPU seconds (vCPU-s) and gibibyte seconds (GiB-s). The first 180,000 vCPU-s and 360,000 GiB-s each month are free.

Applications scale on-demand based on requests and events. Container Apps replicas are billed for active usage when they are running. An application can be configured to scale to zero replicas when there are no requests or events to process. No usage charges apply when an application is scaled to zero.

You can optionally configure Container Apps with a minimum number of replicas to be always running in idle mode. When an application scales down to its minimum number of replicas, usage is charged at a reduced idle rate when a replica is inactive. A replica enters active mode and is charged at the active rate when it is starting up, when it is processing requests, or when its vCPU or bandwidth usage are above the active billing thresholds[1].

[1] A replica is active when vCPU usage is above 0.01 cores or when data received is above 1000 bytes per second.

Metre	Active Usage Price	Idle Usage Price	Free Grant (Per Month)
vCPU (seconds)	**$0.000034** per second	**$0.000004** per second	180,000 vCPU-seconds
Memory (GiB-Seconds)	**$0.000004** per second	**$0.000004** per second	360,000 GiB-seconds

Requests

Container Apps are billed based on the total number of requests processed each month. The first two million requests are included free each month.

Metre	Price	Free Grant (Per Month)
Requests	**$0.40** per million	2 Million

References

- https://docs.microsoft.com/en-us/azure/container-apps/
- https://kubernetes.io/
- https://www.envoyproxy.io/
- https://keda.sh/
- https://dapr.io/
- https://www.infoq.com/articles/multi-runtime-microservice-architecture
- https://yourazurecoach.com/2021/11/03/container-apps-the-missing-piece-in-the-azure-compute-puzzle
- https://github.com/iancooper/Presentations/blob/master/12-factor%20apps%20in%20.NET%20(Short).pptx
- https://modernad.services.hclets.com/wp-content/uploads/2018/06/ENTERPRISE-CLOUD-NATIVE-MATURITY-MODEL.pdf

Thanks for joining !

DEV CAFE
Connect and share

https://www.facebook.com/devcafevn