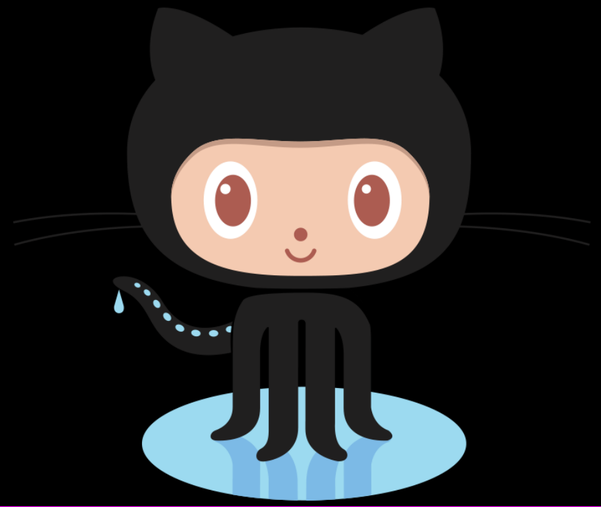
https://github.com/ewolff/microservice-dapr



# Example

Each box is a container

Invoicing

Order Process

Shipping

# Feeds

- You can subscribe to feeds, blogs, podcasts etc
- Access through HTTP

- Idea: Provide a feeds of events / orders

```
{
  "updated": "2022-05-28T18:07:03.931+0000",
  "orders": [
    {
      "id": 1,
      "link": "http://swaglab.rocks:8080/order/1",
      "updated": "2022-05-28T18:07:03.931+0000"
    }
  ]
}
```


Can use content negotiation for specific data for e.g. invoice / delivery (Accept header)

# Subscribing to Feed

- Poll the feed (HTTP GET)
- Client decides when to process new events
- …but very inefficient
- Lots of unchanged data

# HTTP Caching

- Client GETs feed
- Server send data

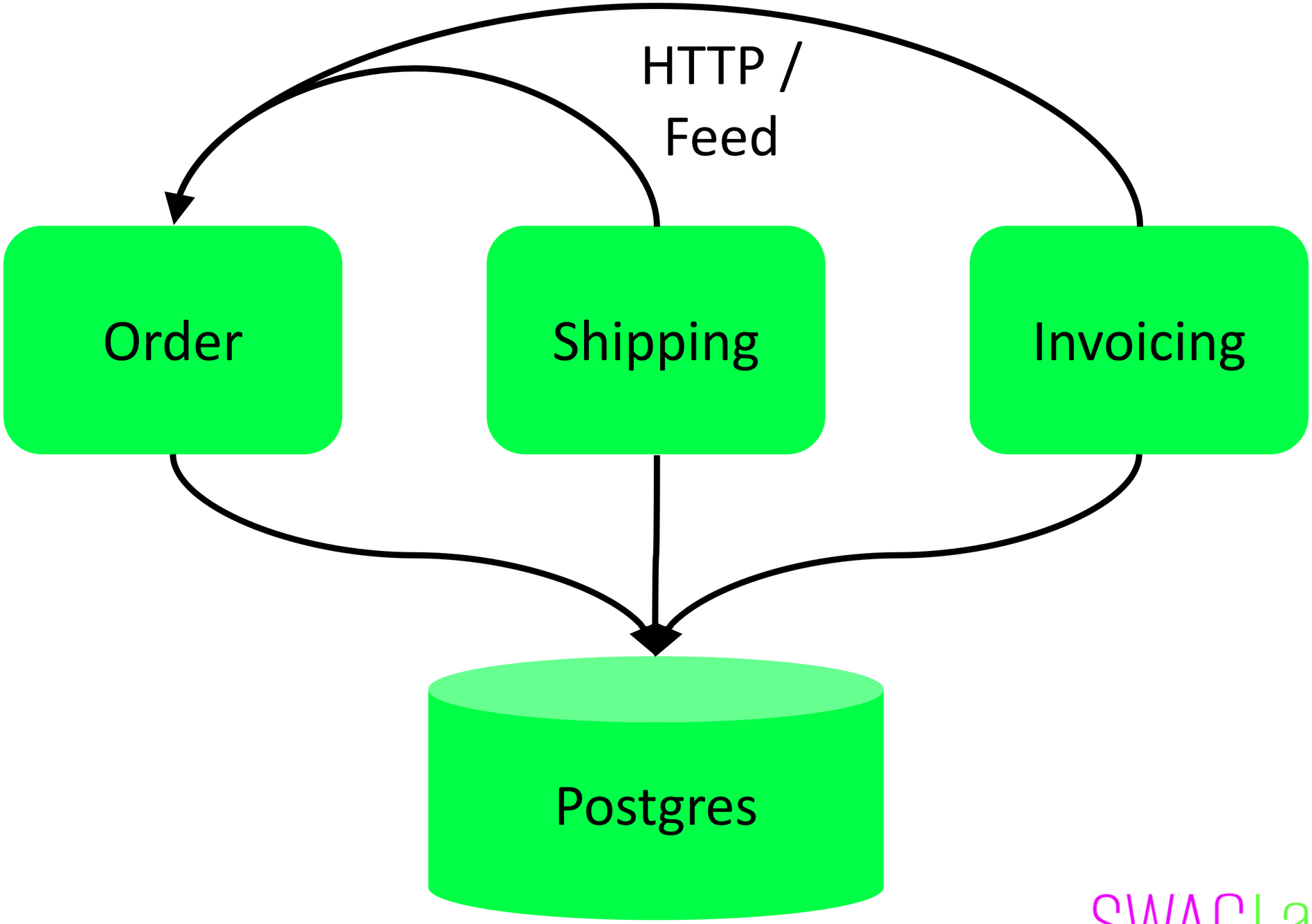
+ Last-Modified header

- Client sends GET

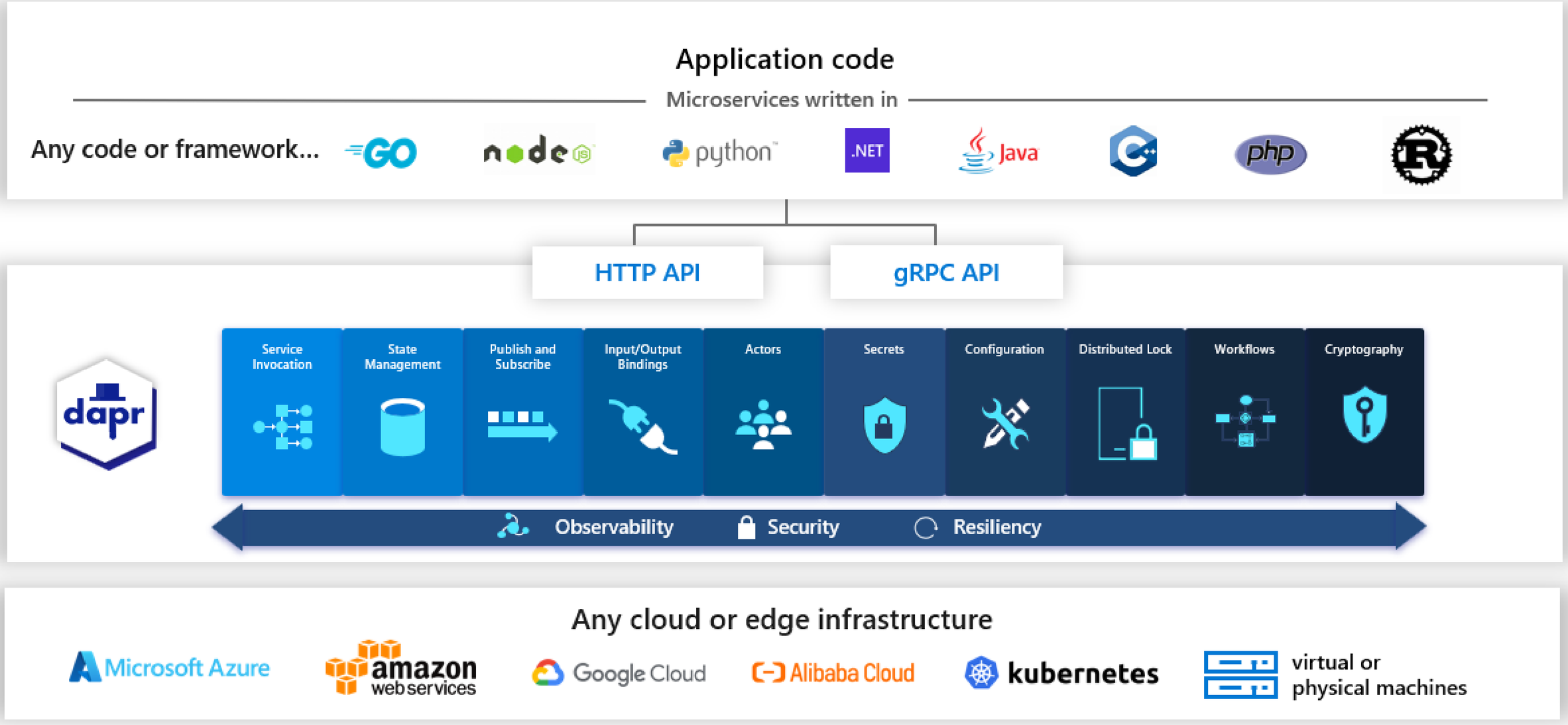
+ If-Modified-Since header

- Server: 304 (Not modified)

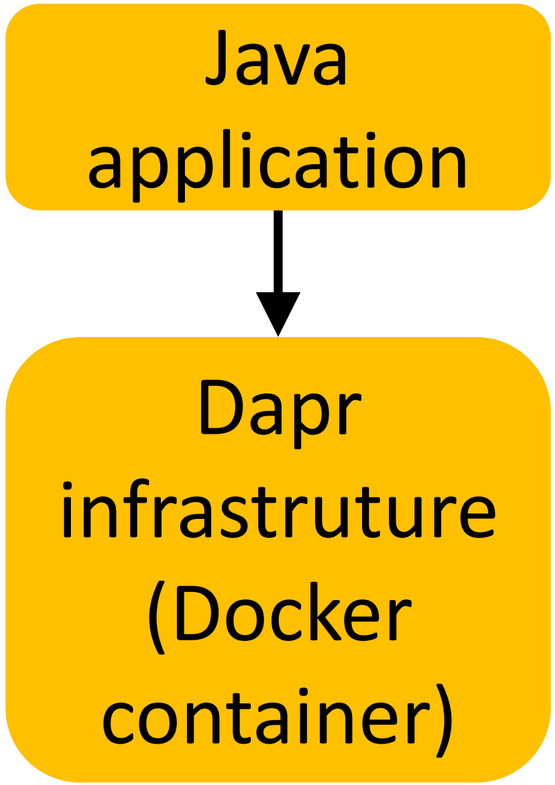
…or new data


HTTP /
Feed
Order
Shipping
Invoicing
Postgres

# Dapr


Application code
Microservices written in
Any code or framework...
GO
node
python
.NET
Java
C++
php
R
HTTP API
gRPC API
dapr
Service Invocation
State Management
Publish and Subscribe
Input/Output Bindings
Actors
Secrets
Configuration
Distributed Lock
Workflows
Cryptography
Observability
Security
Resiliency
Any cloud or edge infrastructure
Microsoft Azure
amazon web services
Google Cloud
Alibaba Cloud
kubernetes
virtual or physical machines

# Dapr Local for Development




dapr init

Dapr listens on e.g. port 3500 -> http://localhost:3500

# Dapr: Alternative to Feed

- Might use pubsub abstraction instead of feeds
- Many implementations (Kafka, SQS, … >40) of  varying maturity
- https://docs.dapr.io/reference/components-reference/supported-pubsub/

# Dapr: Alternative to Feed

- Access via HTTP / sidecar or declarative
- Unified API
- Content-based routing
- Dead letter topics
- Outbox pattern
- At least once
- Consumer groups (for some implementations)…

# Dapr: Alternative to Feed

- E.g. RabbitMQ and Kafka are quite different (e.g. backpressure, persistence)
- I'd rather use the specific strength of a specific technology…
- …than seemingly be able to exchange it.
- You can argue that I did not use Dapr as intended.

# Dapr: Alternative to Postgres

- Might use state store instead of Postgres
- Many implementations (Postgres, MySQL, Cassandra … >25) of  varying maturity i.e. least common denominator
- https://docs.dapr.io/reference/components-reference/supported-state-stores/

# Dapr: Alternative to Postgres

- Access via HTTP / sidecar
- Unified key / value API for JSON data
- Query in alpha
- Transactional outbox with pub / sub
- https://docs.dapr.io/reference/components-reference/supported-state-stores/

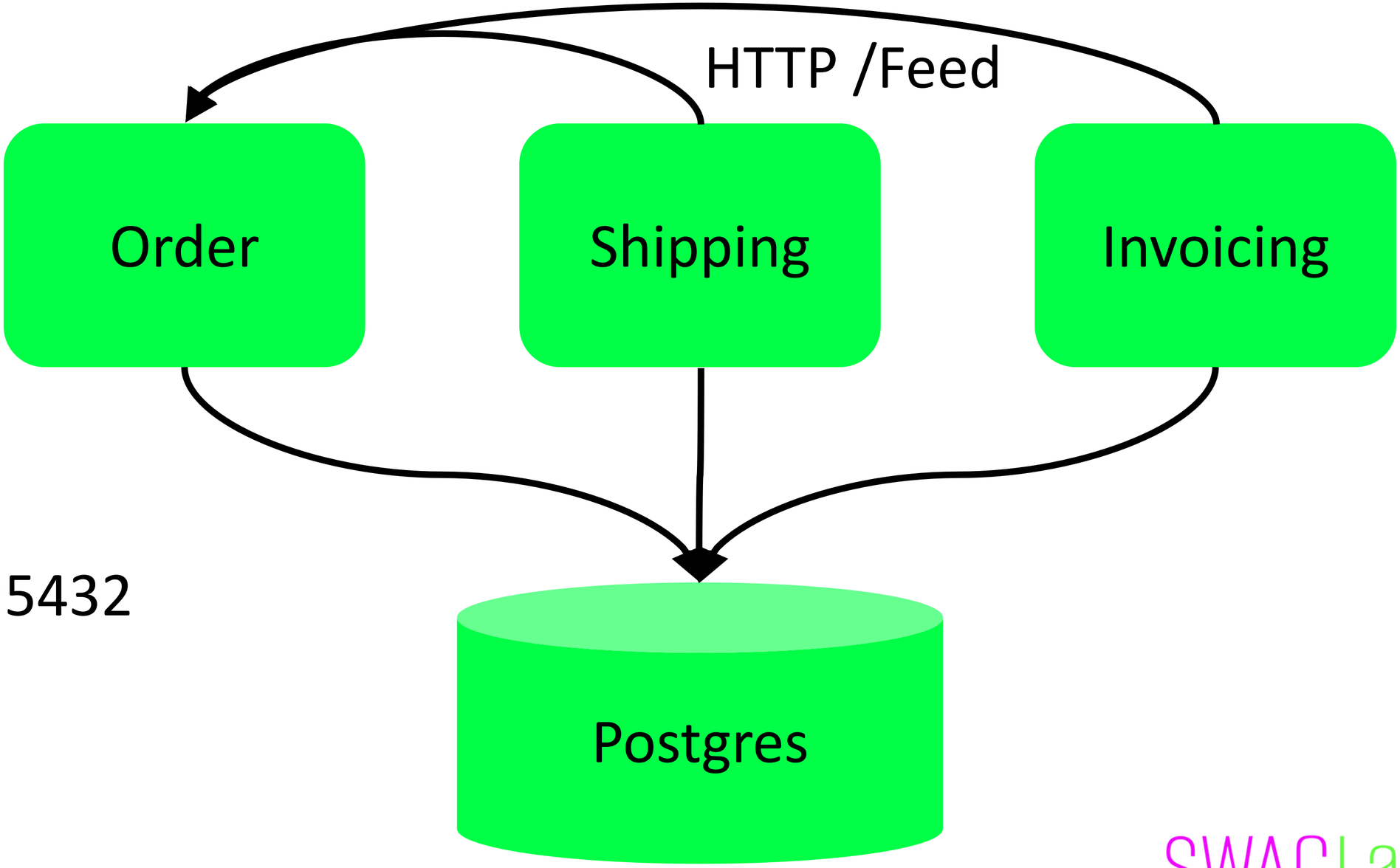
# Dapr: Alternative to Postgres

- Postgres (relational) vs. Redis (key / value in memory) are very different wrt speed etc
- I'd rather use specific strength of a technology…

…than seemingly be able to exchange it.

- You can argue that I did not use Dapr as intended.
- Should persistence be internal to the microservice?
- CORBA services flash back

Sidecar: http://localhost:3500/v1.0/invoke/order/method/feed



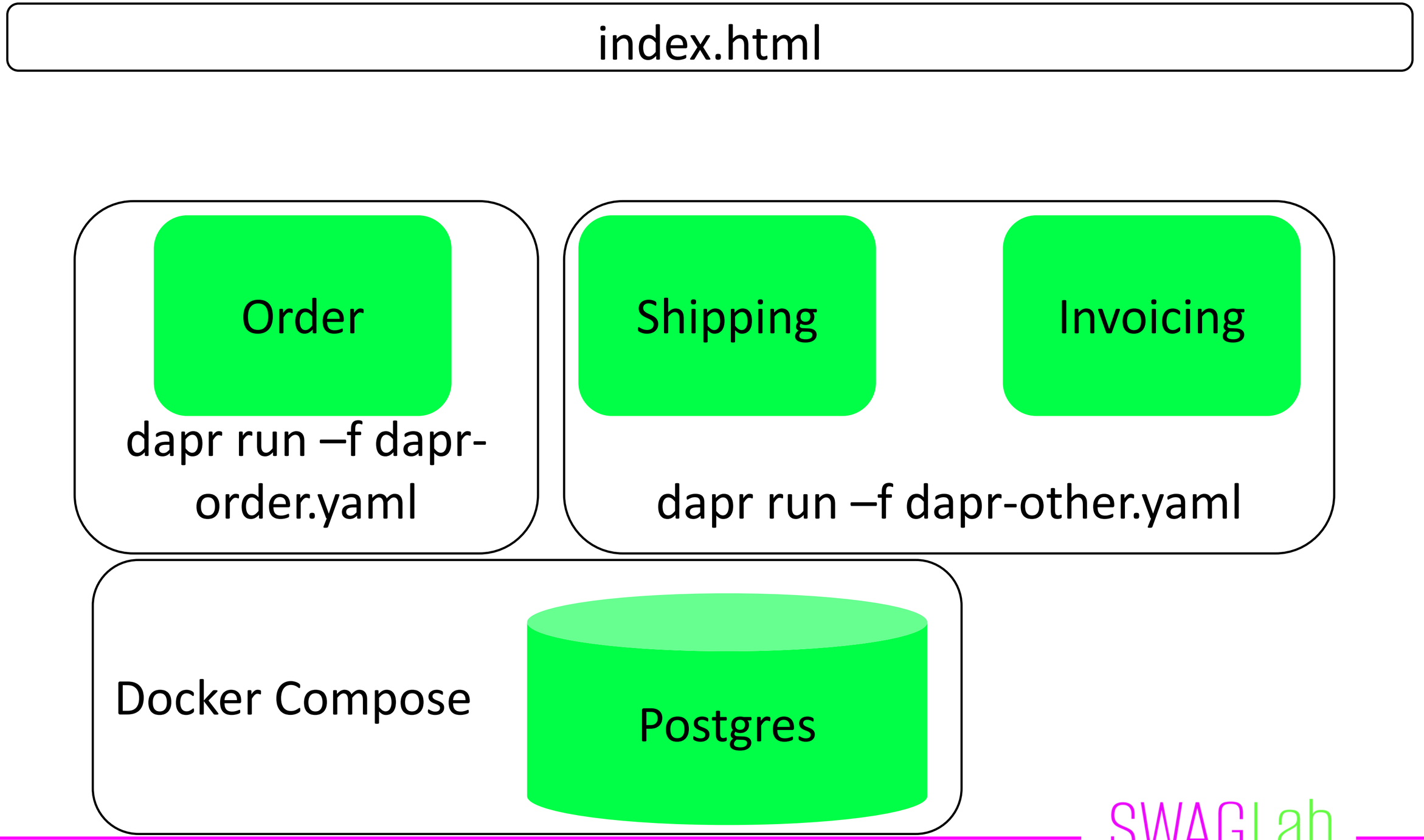

localhost:5432

# Dapr: Service Discovery

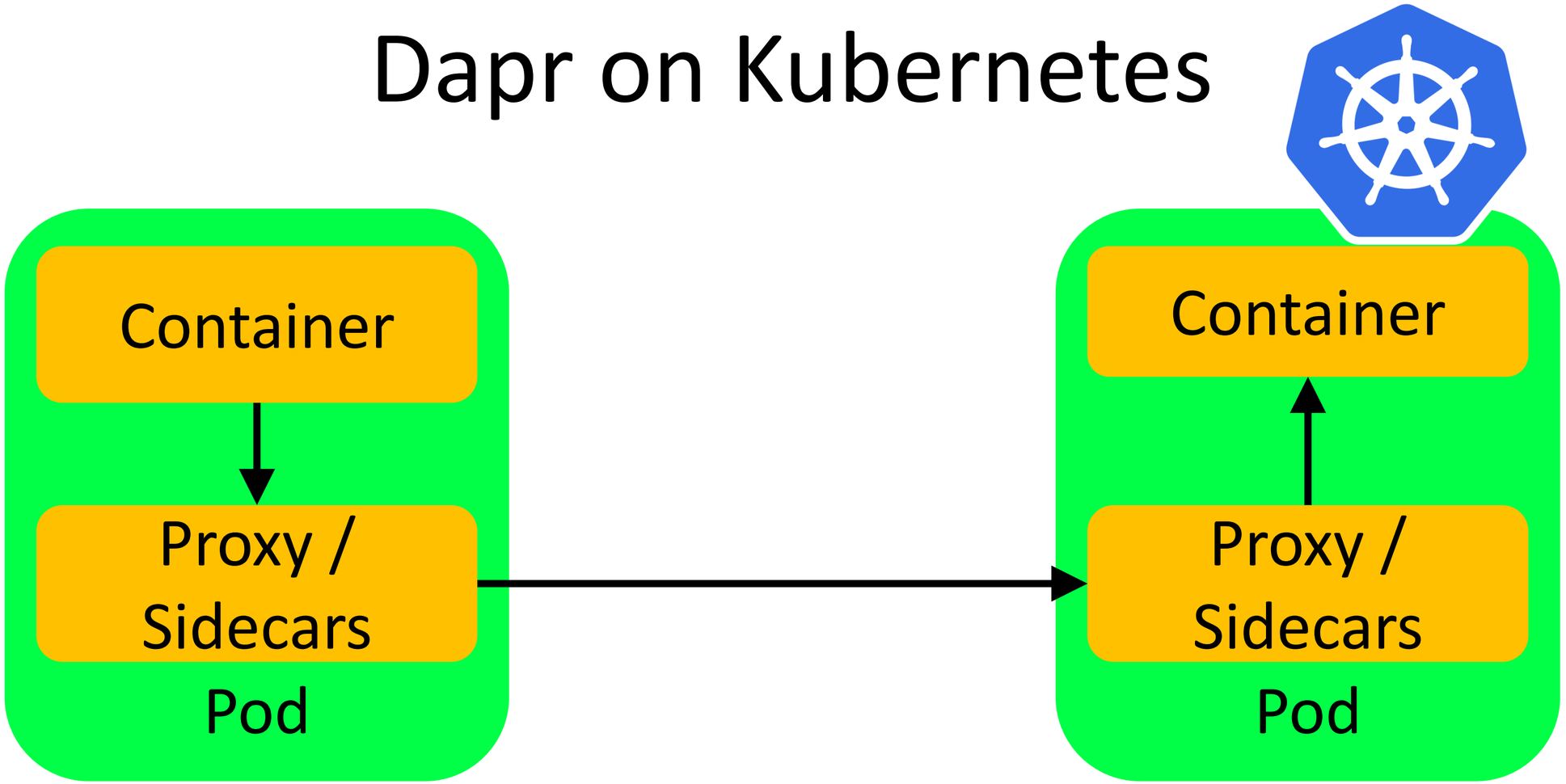
- Service discovery via sidecar
- Could have been an HTTP proxy.

- Works only for Dapr services

# Demo: Local Dapr

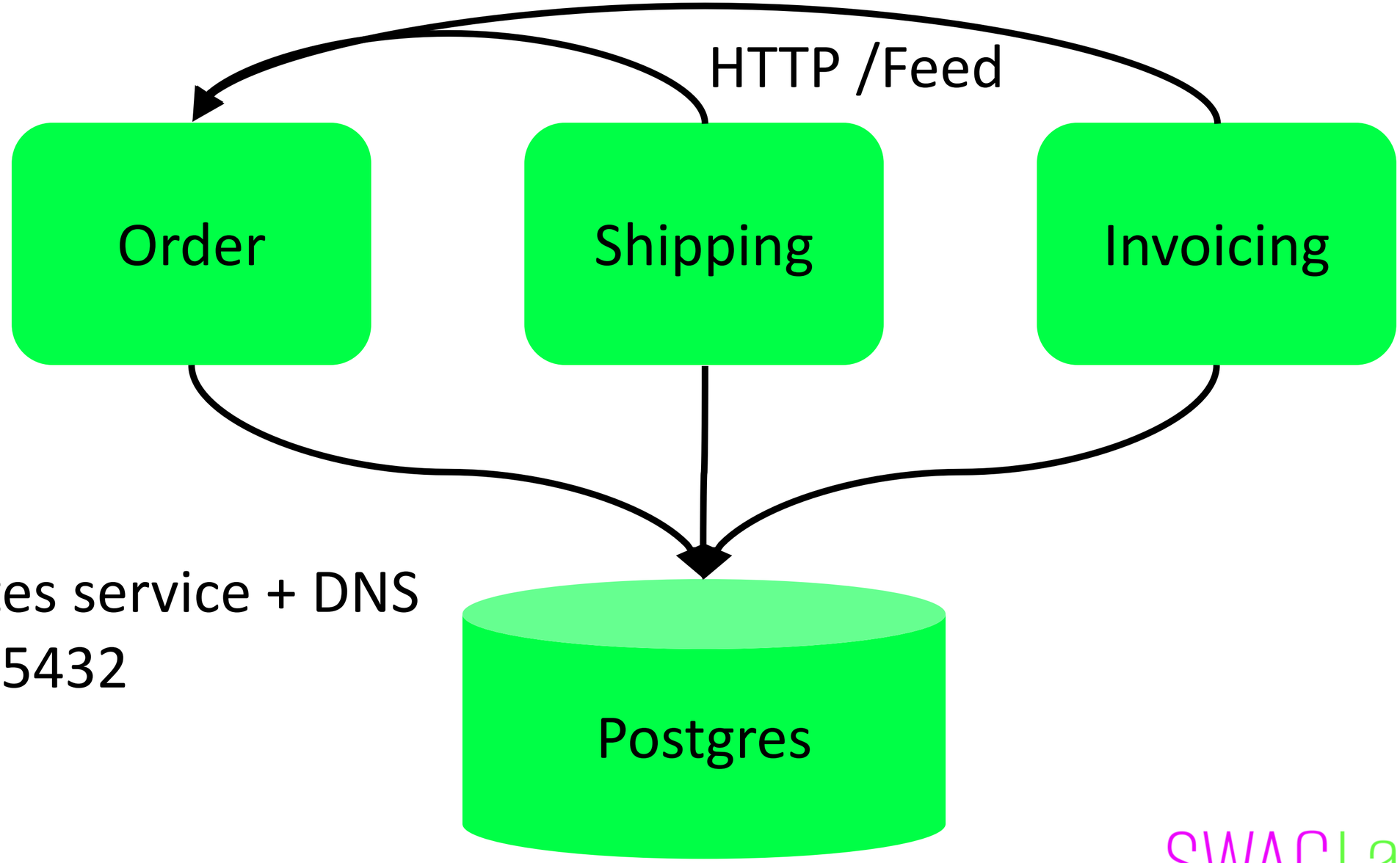

index.html
Order
dapr run –f dapr-order.yaml
Shipping
Invoicing
dapr run –f dapr-other.yaml
Docker Compose
Postgres

# Dapr on Kubernetes

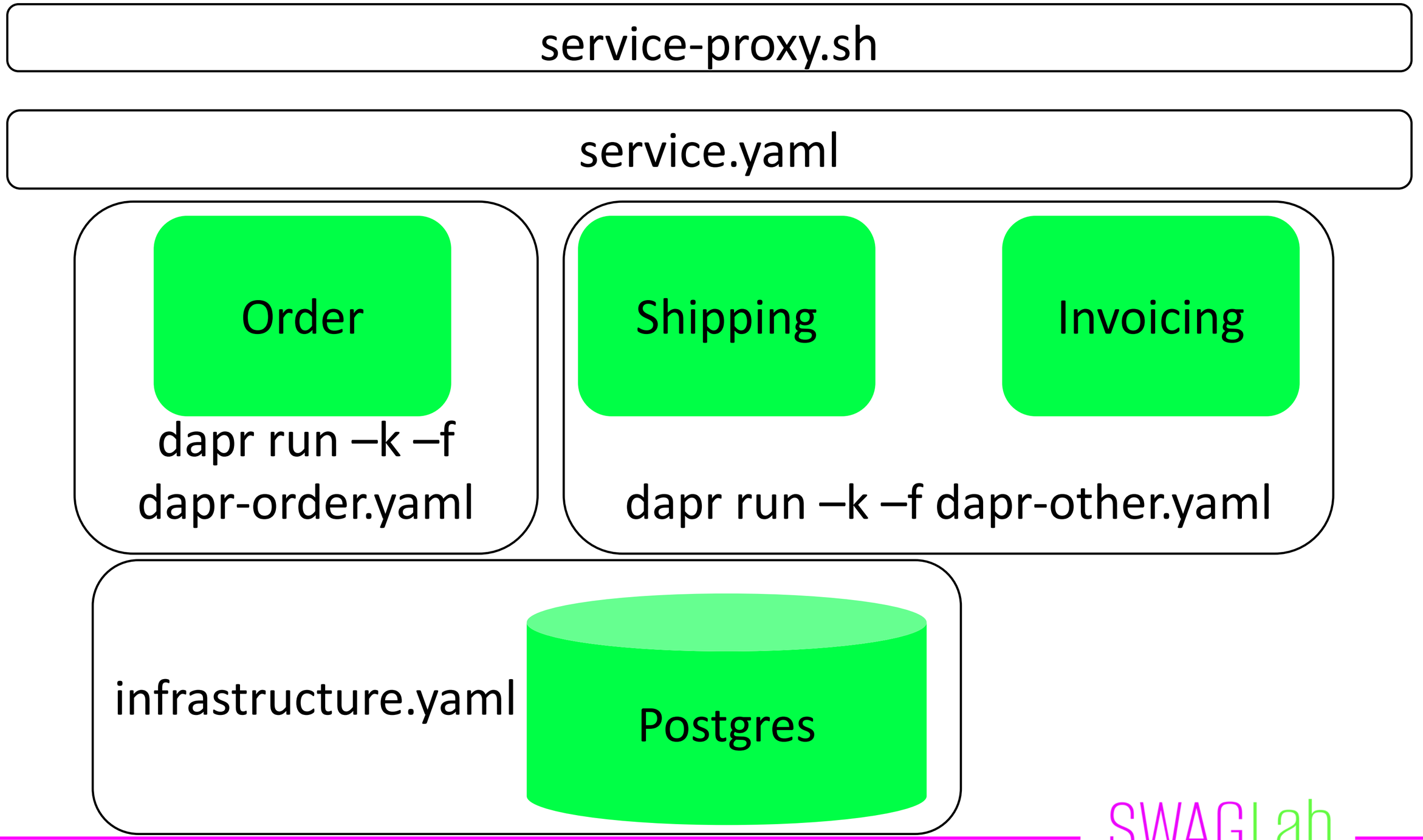



Sidecar listens on e.g. port 3500 -> http://localhost:3500

Sidecar: http://localhost:3500/v1.0/invoke/order/method/feed

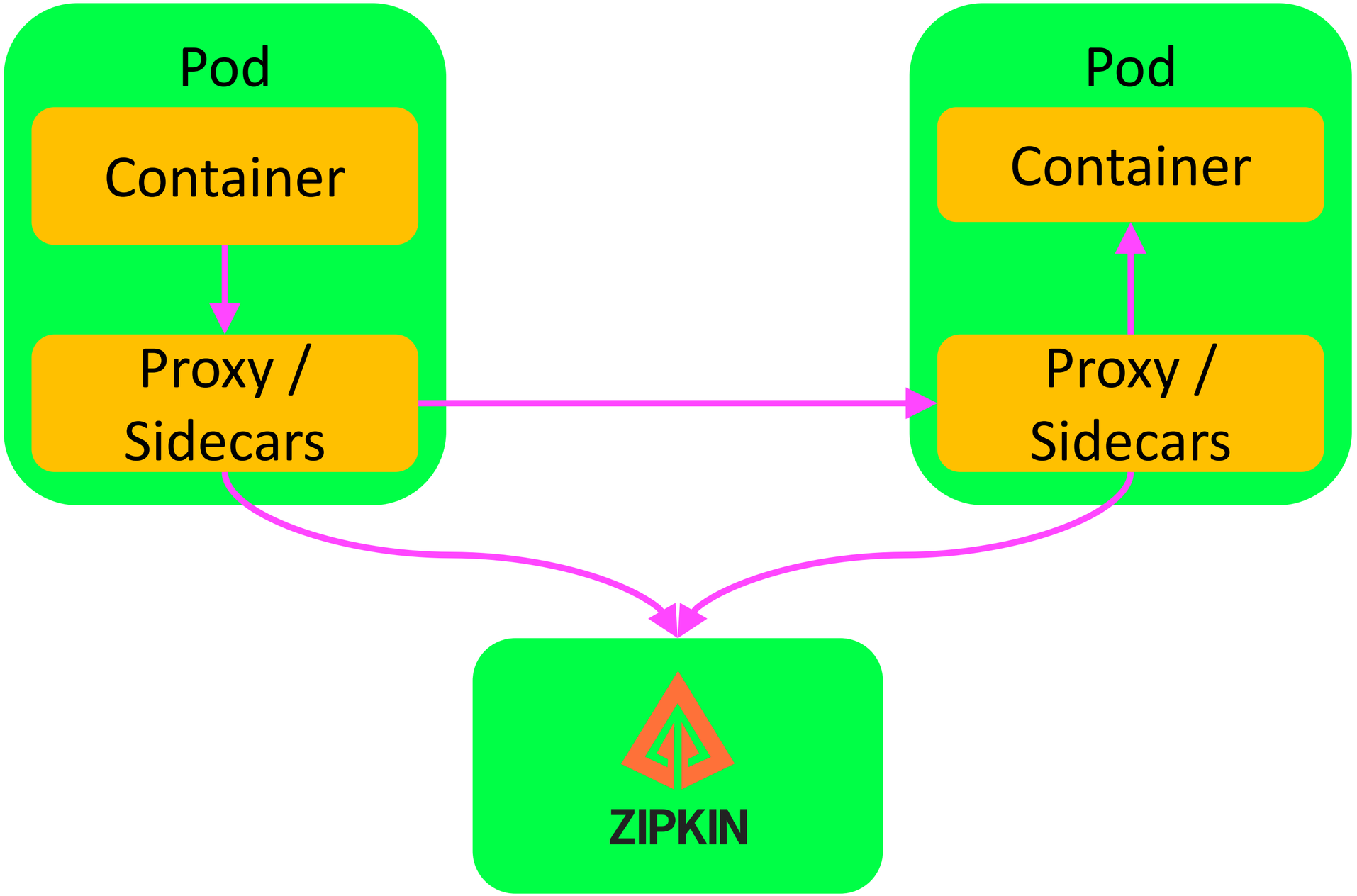



Kubernetes service + DNS
postgres:5432


service-proxy.sh
service.yaml
Order
Shipping
Invoicing
dapr run –k –f
dapr-order.yaml
dapr run –k –f dapr-other.yaml
infrastructure.yaml
Postgres

# Demo: Dapr Deployment

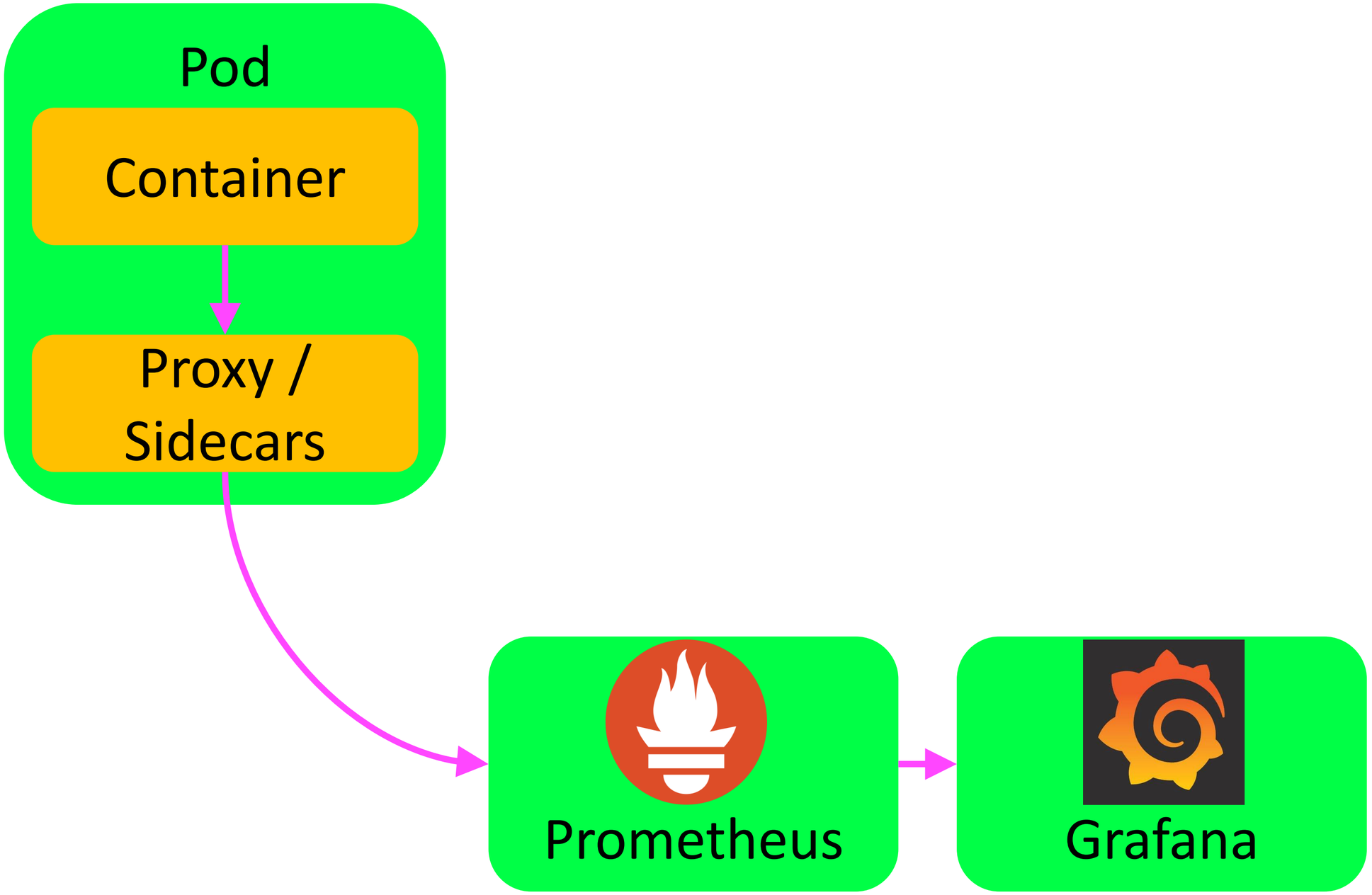
# Dapr: Tracing


Pod
Container
Proxy /
Sidecars
Pod
Container
Proxy /
Sidecars
ZIPKIN

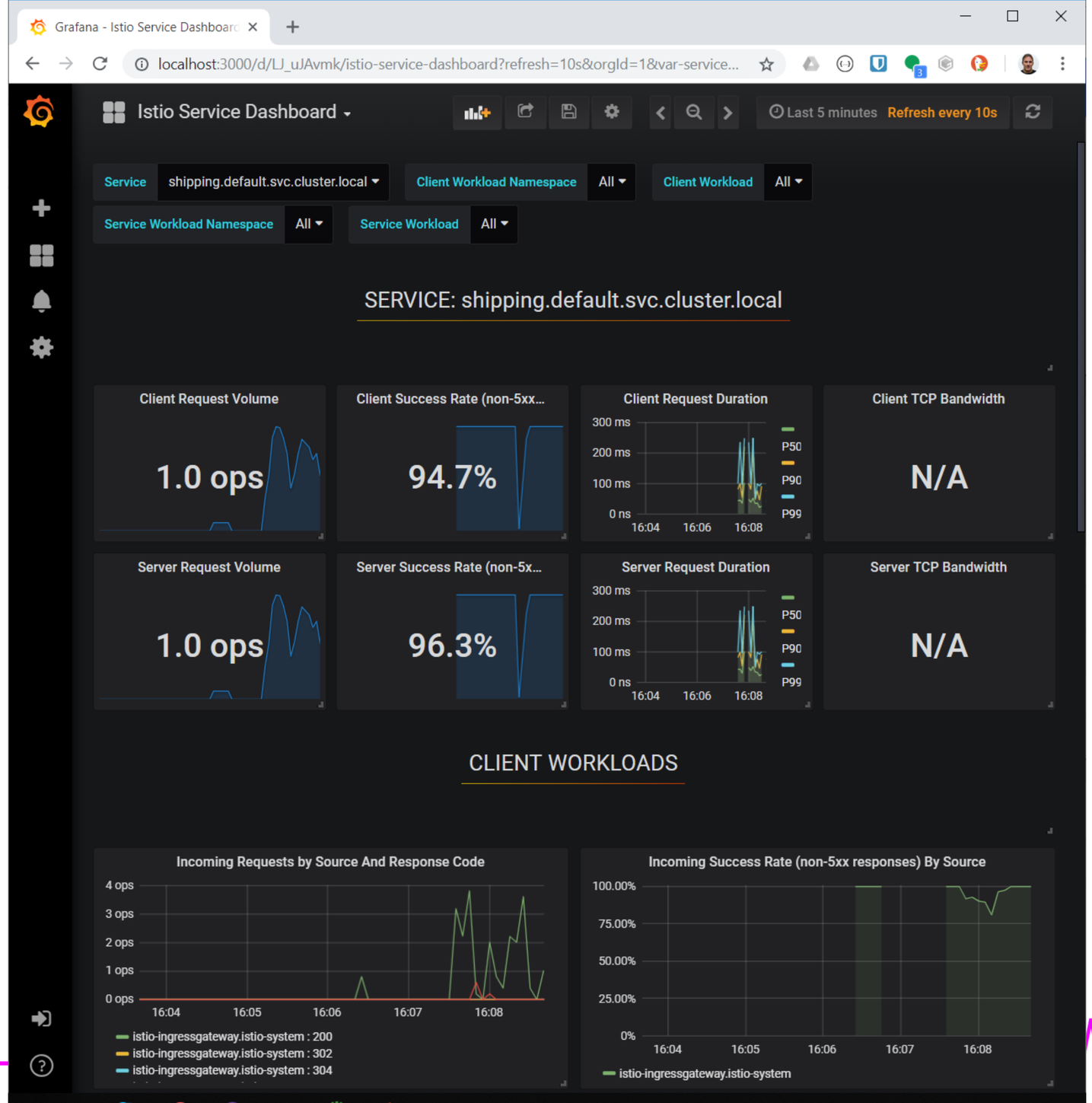
# Dapr Monitoring

# Monitoring in the Dapr Example

- Proxy / sidecar reports metrics
- Pre-defined Grafana dashboards


Pod
Container
Proxy /
Sidecars
Prometheus
Grafana

# Demo: Dapr Monitoring


Grafana - Istio Service Dashboard
localhost:3000/d/LJ_uJAvmk/istio-service-dashboard?refresh=10s&orgId=1&var-service...
Istio Service Dashboard
Last 5 minutes
Refresh every 10s
Service
shipping.default.svc.cluster.local
Client Workload Namespace
All
Client Workload
All
Service Workload Namespace
All
Service Workload
All
SERVICE: shipping.default.svc.cluster.local
Client Request Volume
1.0 ops
Client Success Rate (non-5xx...
94.7%
Client Request Duration
300 ms
200 ms
100 ms
0 ns
16:04
16:06
16:08
P50
P90
P99
Client TCP Bandwidth
N/A
Server Request Volume
1.0 ops
Server Success Rate (non-5x...
96.3%
Server Request Duration
300 ms
200 ms
100 ms
0 ns
16:04
16:06
16:08
P50
P90
P99
Server TCP Bandwidth
N/A
CLIENT WORKLOADS
Incoming Requests by Source And Response Code
4 ops
3 ops
2 ops
1 ops
0 ops
16:04
16:05
16:06
16:07
16:08
istio-ingressgateway.istio-system : 200
istio-ingressgateway.istio-system : 302
istio-ingressgateway.istio-system : 304
Incoming Success Rate (non-5xx responses) By Source
100.00%
75.00%
50.00%
25.00%
0%
16:04
16:05
16:06
16:07
16:08
istio-ingressgateway.istio-system

# Example:
# Resilience

# Resilience

- Example: data becomes outdated
- Synchronous call: What if the other system fails?
- Must be part of the domain logic
- I.e. accept order if stock unknown?

# Resilience: Timeout

- If a call takes too long
- …the thread will wait and block
- If the call takes really long
- …all thread will end up blocked

- Solution: Add a timeout to operations

# Resilience: Retry

- If a call results in 5xx or a connect failure
- Retry!

- Solution: Add retries to operations

# Resilience: Circuit Breaker

- Circuit breaker: cut circuit if there's a short circuit
- Idea: Avoid overload by cutting circuit

- Software Circuit Breaker:
- Limit # of waiting requests
- Exclude failed instances

# Resilience: Dapr

- Configuration for the sidecar
- Timeout
- Retry
- Circuit Breaker

# Resilience: Domain Logic

- Example: data becomes outdates
- Synchronous call: What if the other system fails?
- Must be part of the domain logic
- I.e. accept order if stock unknown?

# More Dapr Building Blocks

- Service invocation (as seen)
- State management
- Publish and subscribe
- Bindings to external sources
- Actors (stateful, long-running objects)
- Secrets management
- Configuration
- Distributed locks
- Workflow
- Cryptography

# How to Dig Deeper…

- Try the demo for yourself:
  https://github.com/ewolff/microservice-dapr
  Learn more about Dapr
  https://docs.dapr.io/#learn-more-about-dapr
- Read Microservices Recipes / Practical Guide to Microservices to understand technology alternatives
- Read Microservices Primer / Microservice to understand microservices architecture