
KubeCon
CloudNativeCon
Europe 2025
LONDON BUS

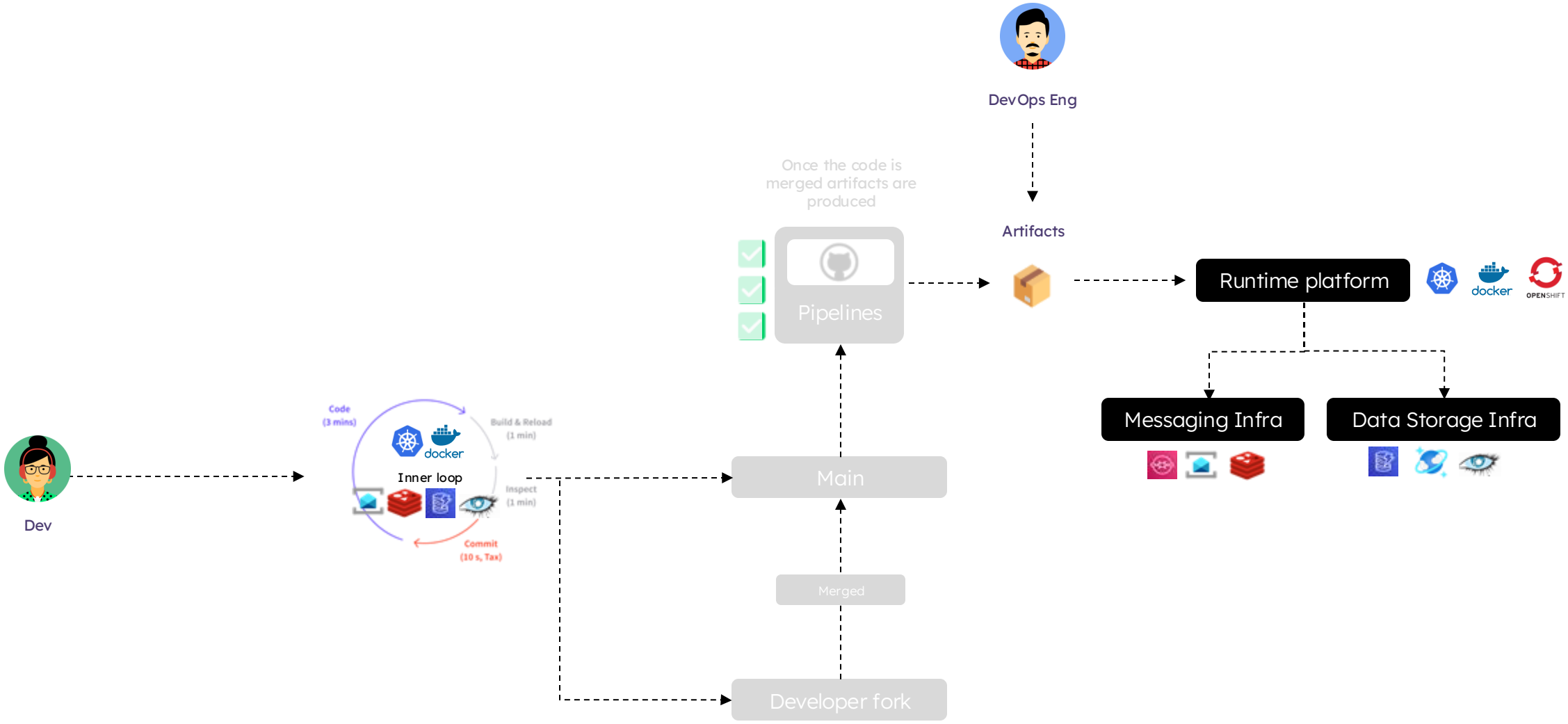
KubeCon | CloudNativeCon
Europe 2025

Dapr + Score

Mixing the Perfect Cocktail for an Enhanced Developer Experience

Kendall Roden - PM @ Diagrid
Mathieu Benoit - CSE @ Humanitec

Platform and infrastructure concerns are **increasing cognitive load** and **decreasing productivity** for developers



Setting the stage

Kendall
Application Engineer

Full-stack developer focused on business logic and **delivering features**

Wants to spend **less time on infrastructure** concerns and boilerplate code

Values consistency, simplicity, and **productivity**

Setting the stage

Kendall
Application Engineer

Full-stack developer focused on business logic and **delivering features**

Wants to spend **less time on infrastructure** concerns and boilerplate code

Values consistency, simplicity, and **productivity**

Mathieu
Platform/DevEx Team

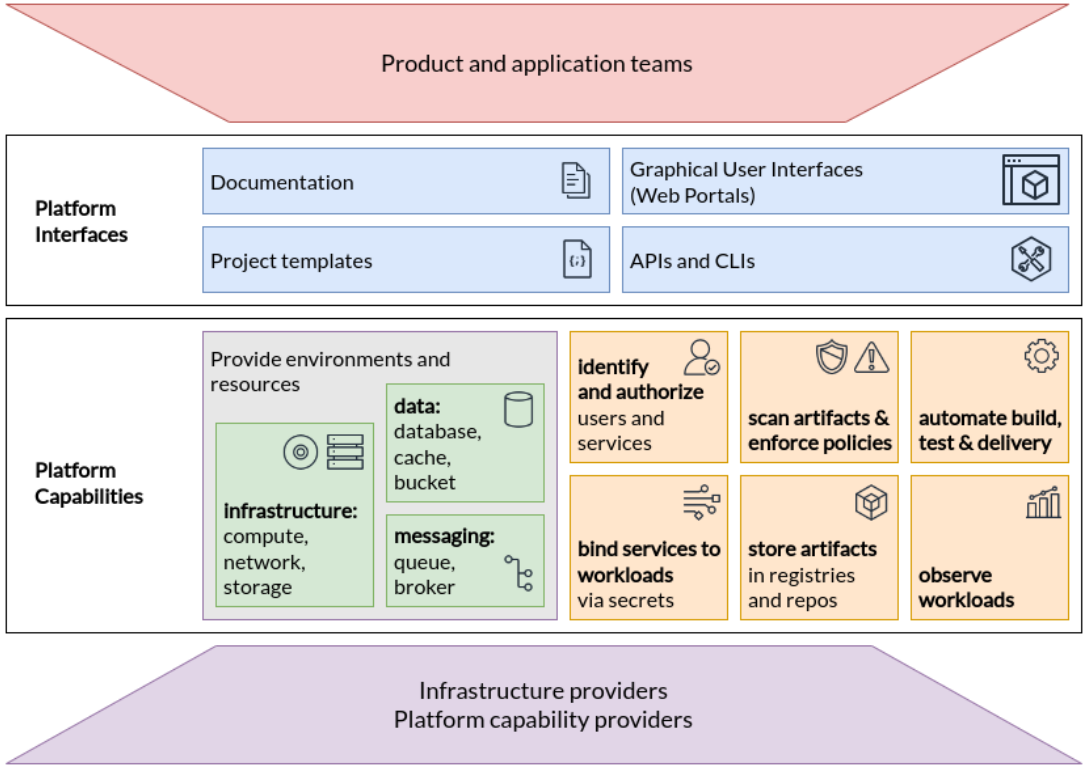
Responsible for building and maintaining the standardized interfaces that form his company's **developer platform**

Focuses on **standardization**, security, and operational excellence

Wants to **enable developers** to work quickly and efficiently while maintaining governance

Building Internal Platforms for Enablement

Capabilites of Platforms



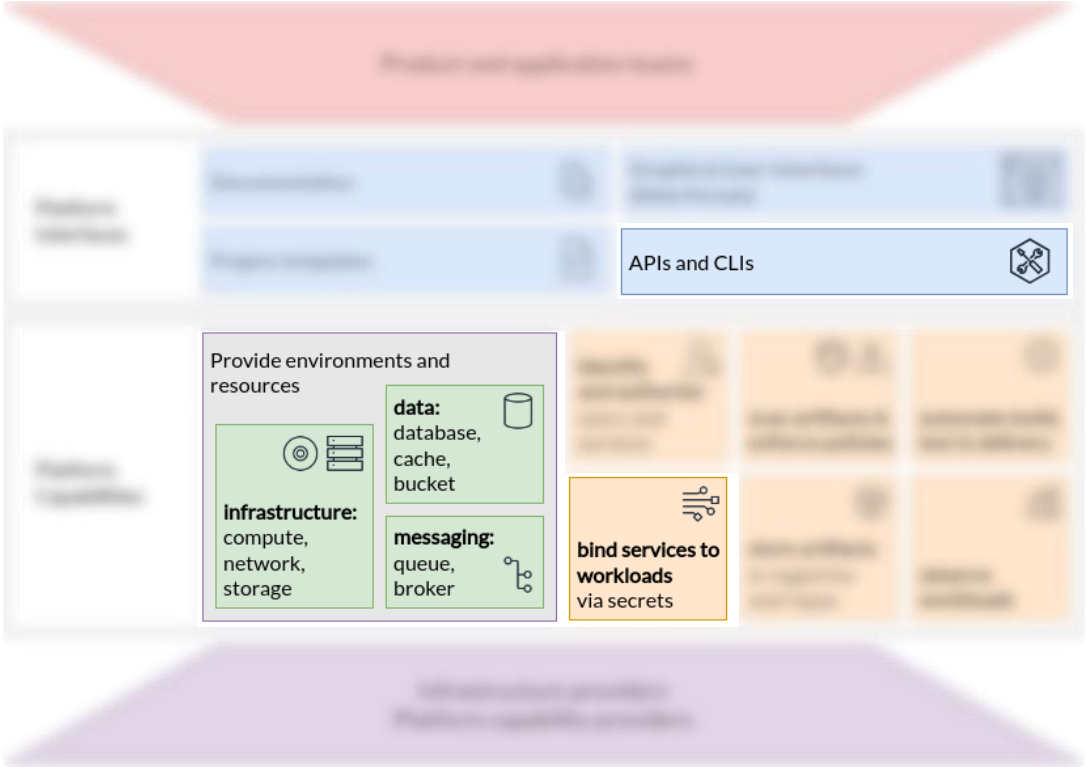
Platform Engineering Maturity Model

ASPECT		PROVISIONAL	OPERATIONAL	SCALABLE	OPTIMIZING
Investment	*How are staff and funds allocated to platform capabilities?*	Voluntary or temporary	Dedicated team	As product	Enabled ecosystem
Adoption	*Why and how do users discover and use internal platforms and platform capabilities?*	Erratic	Extrinsic push	Intrinsic pull	Participatory
Interfaces	*How do users interact with and consume platform capabilities?*	Custom processes	Standard tooling	Self-service solutions	Integrated services
Operations	*How are platforms and their capabilities planned, prioritized, developed and maintained?*	By request	Centrally tracked	Centrally enabled	Managed services
Measurement	*What is the process for gathering and incorporating feedback and learning?*	Ad hoc	Consistent collection	Insights	Quantitative and qualitative

Source: [CNCF Platforms Working Group White Paper]

CNCF Platform Interfaces

Capabilites of Platforms



Platform Engineering Maturity Model

ASPECT		PROVISIONAL	OPERATIONAL	SCALABLE	OPTIMIZING
Investment	How are staff and funds allocated to platform capabilities?	Voluntary or temporary	Dedicated team	As product	Enabled ecosystem
Adoption	Why and how do users discover and use internal platforms and platform capabilities?	Erratic	Extrinsic push	Intrinsic pull	Participatory
Interfaces	*How do users interact with and consume platform capabilities?*	Custom processes	Standard tooling	Self-service solutions	Integrated services
Operations	How are platforms and their capabilities planned, prioritized, developed and maintained?	By request	Centrally tracked	Centrally enabled	Managed services
Measurement	What is the process for gathering and incorporating feedback and learning?	Ad hoc	Consistent collection	Insights	Quantitative and qualitative

Source: [CNCF Platforms Working Group White Paper]

Enabling workload-centric development

Platform Interfaces

Product/Application Team

Product/Application Team

Product/Application Team

dapr Dapr APIs & SDKs

Score Specification

Enabling workload-centric development

Platform Interfaces

Platform Capabilities

Product/Application Team

Product/Application Team

Product/Application Team

dapr Dapr APIs & SDKs

Score Specification

Platform Engineering Team

- **Self-service tools** & frameworks
- **Automate processes** to streamline development

dapr Dapr Components

Score Implementations

Infrastructure

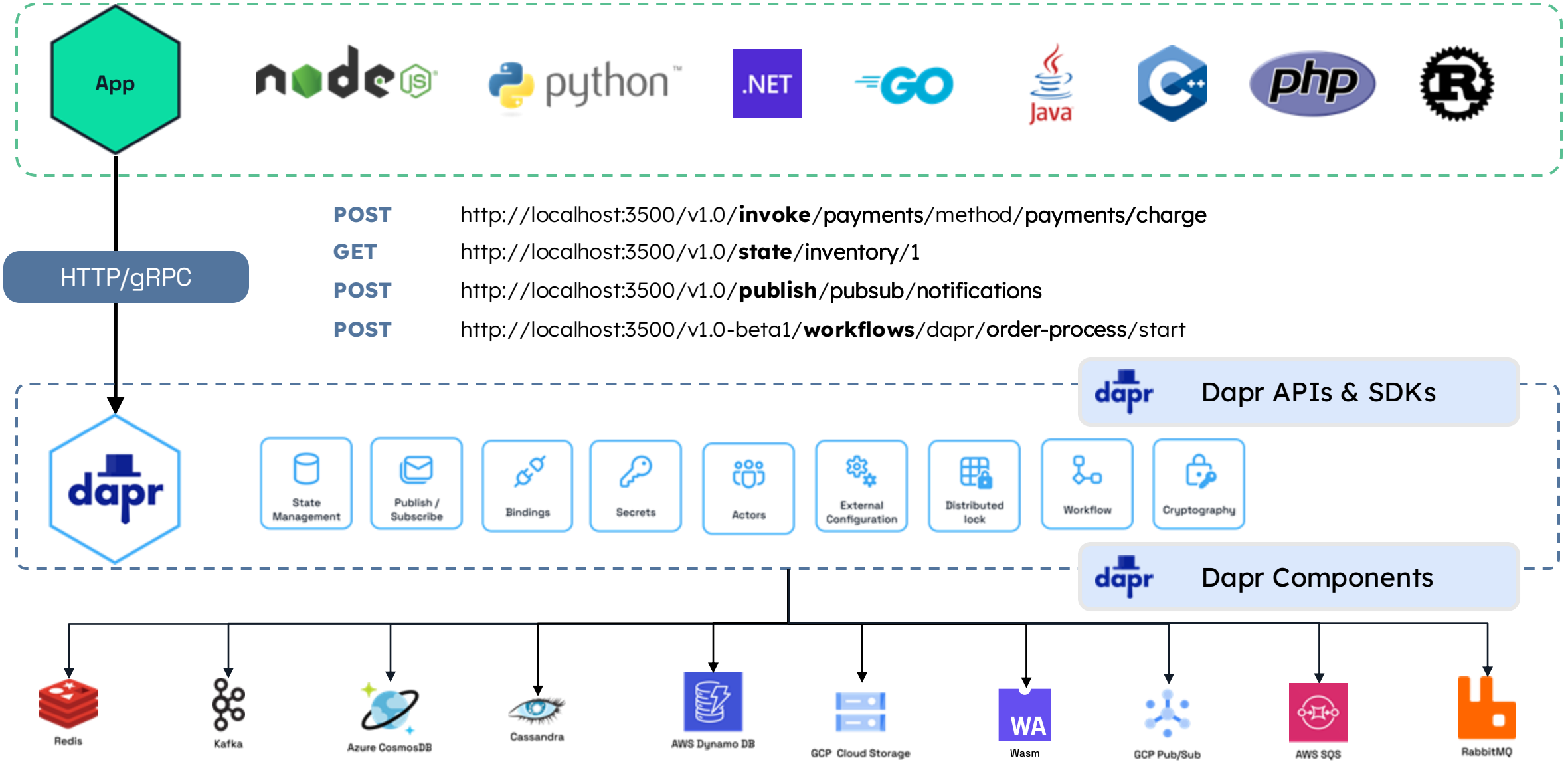
Azure aws Google Cloud

KubeCon | CloudNativeCon
Europe 2025

Dapr + Score

A practical walk-through

Dapr overview



Standardizing on Dapr

1

Author apps using Dapr locally

Use the Dapr SDKs or the HTTP/GRPC APIs

2

Configure resources for your Dapr sidecars

Connect your applications to infrastructure

3

Run with local Dapr tooling and CLI for inner loop development

Get up and running quickly with your apps and dependencies

```python
# 1. Import Redis package
import redis
import json
from datetime import datetime

# 2. Function to save message
def save_message(key, message):
    try:
        # 3. Connect to Redis with detailed configuration
        r = redis.Redis(
            host='redis-server',
            port=6379,
            password='your-password',
            decode_responses=True,
            socket_timeout=5,
            retry_on_timeout=True
        )

        # 4. Prepare data
        payload = {
            'content': message,
            'timestamp': datetime.now().isoformat()
        }

        # 5. Save to Redis
        r.set(key, json.dumps(payload))

        # 6. Must close connection
        r.close()

        return {"status": "success"}

    except redis.RedisError as e:
        print(f"Redis error: {e}")
        raise Exception("Failed to save message")
```

Standardizing on Dapr

1

Author apps using Dapr locally

Use the Dapr SDKs or the HTTP/GRPC APIs

2

Configure resources for your Dapr sidecars

Connect your applications to infrastructure

3

Run with local Dapr tooling and CLI for inner loop development

Get up and running quickly with your apps and dependencies

```python
# 1. Import Dapr client
from dapr.clients import DaprClient
from datetime import datetime

# 2. Function to save message
def save_message(key, message):
    try:
        # 3. Create Dapr client - no connection details!
        with DaprClient() as client:
            # 4. Prepare data
            payload = {
                'content': message,
                'timestamp': datetime.now().isoformat()
            }

            # 5. Save to state store
            client.save_state(
                store_name="statestore",
                key=key,
                value=payload
            )

            # No connection management needed!

            return {"status": "success"}

    except Exception as e:
        print(f"Dapr error: {e}")
        raise Exception("Failed to save message")
```

Standardizing on Dapr

1

Author apps using Dapr locally

Use the Dapr SDKs or the HTTP/GRPC APIs

2

Configure resources for your Dapr sidecars

Connect your applications to infrastructure

3

Run with local Dapr tooling and CLI for inner loop development

Get up and running quickly with your apps and dependencies

```yaml
apiVersion: dapr.io/v1alpha1
kind: Component
metadata:
  name: statestore
  namespace: default
spec:
  type: state.redis
  version: v1
  metadata:
    - name: redisHost
      secretKeyRef:
        name: redis-secrets
        key: host
    - name: redisPassword
      secretKeyRef:
        name: redis-secrets
        key: password
    - name: enableTLS
      value: "true"
    - name: redisDB
      value: "0"
    - name: redisMaxRetries
      value: "5"
    - name: redisMaxConnections
      value: "30"
    - name: keyPrefix
      value: "app:"
auth:
  secretStore: kubernetes
```

```python
# 5. Save to state store
client.save_state(
    store_name="statestore",
    key=key,
    value=payload
)
```

Standardizing on Dapr

1

Author apps using Dapr locally

Use the Dapr SDKs or the HTTP/GRPC APIs

2

Configure resources for your Dapr sidecars

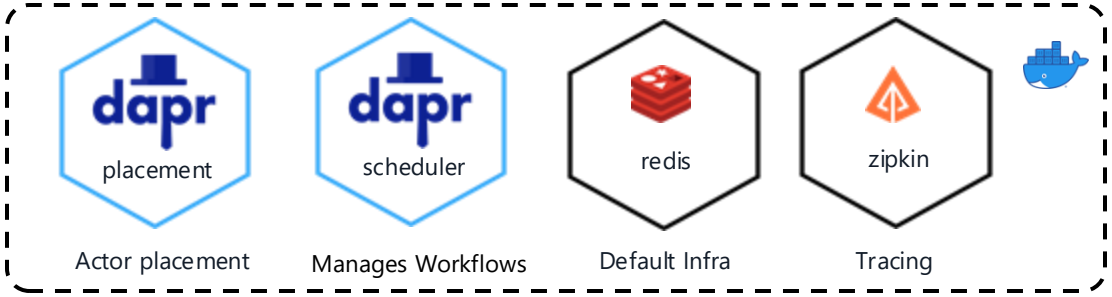
Connect your applications to infrastructure

3

Run with local Dapr tooling and CLI for inner loop development

Get up and running quickly with your apps and dependencies

`dapr init`

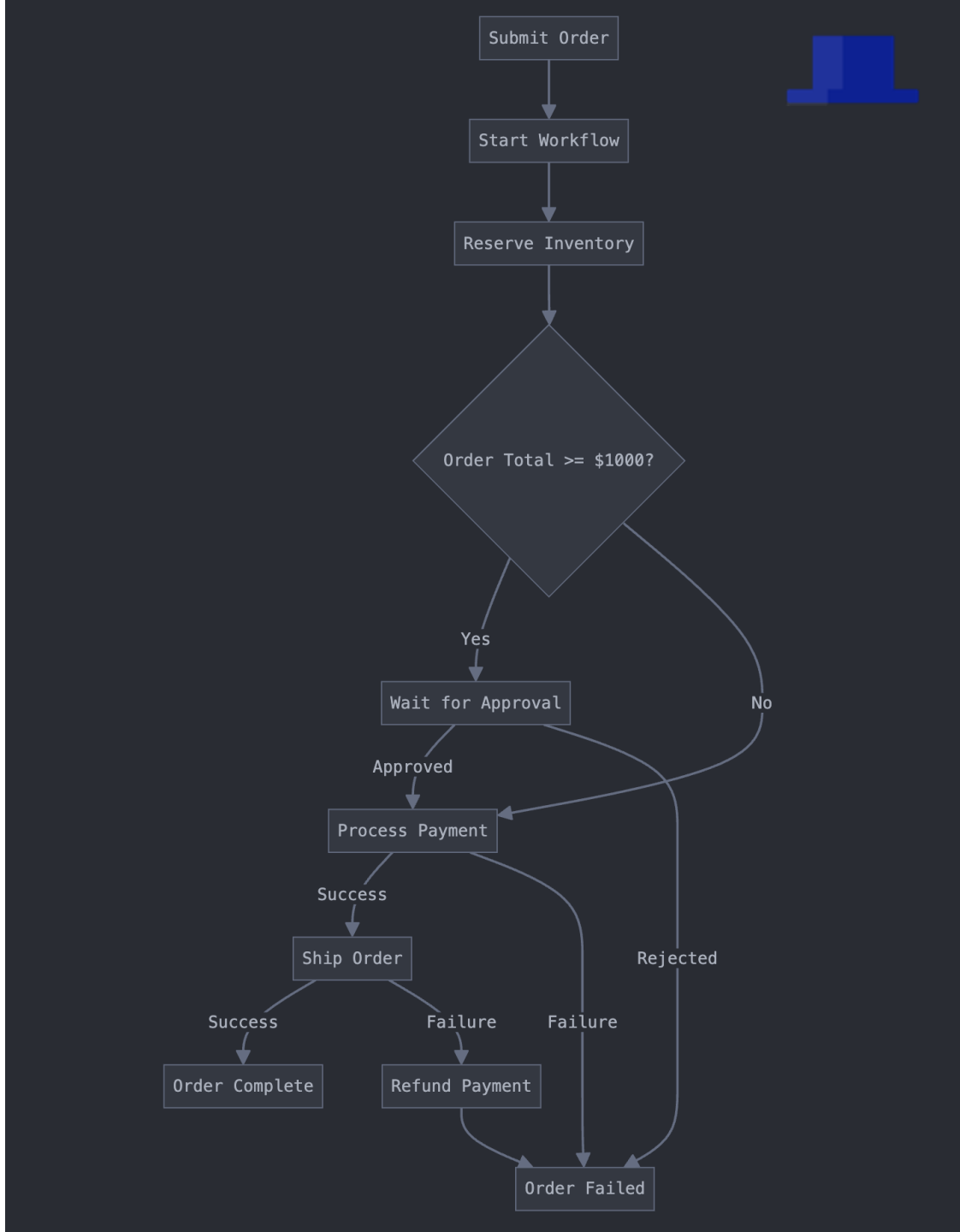


`dapr run -f .`

dapr.yaml > { } common

```yaml
version: 1
common:
  logLevel: info
  appLogDestination: "console"
  daprdLogDestination: "console"
  resourcesPaths: ["./components"]
apps:
  - appId: inventory…
  - appId: notifications
    appPort: 3001
    appDirPath: ./services/notifications
    command: ["python3", "app.py"]
    env:
      PYTHON_DEBUG: "true"
      DEBUG_PORT: "5679"
      SERVICE_NAME: "notifications"
  - appId: order-processor
    appPort: 3000
    appDirPath: ./services/order-processor
    command: ["python3", "app.py"]
    env:…
  - appId: payments…
  - appId: shipping…
```

Demo: Local Dapr Application



Emerging challenges

Kendall
Application Engineer

How can I make sure that my workloads once **containerized** will still successfully work/run?

How to promote in Dev, Staging and **Prod environments**?

Do I need to learn and write **Docker Compose** files or **Kubernetes manifests**?

Emerging challenges

Kendall
Application Engineer

How can I make sure that my workloads once **containerized** will still successfully work/run?

How to promote in Dev, Staging and **Prod environments**?

Do I need to learn and write **Docker Compose** files or **Kubernetes manifests**?

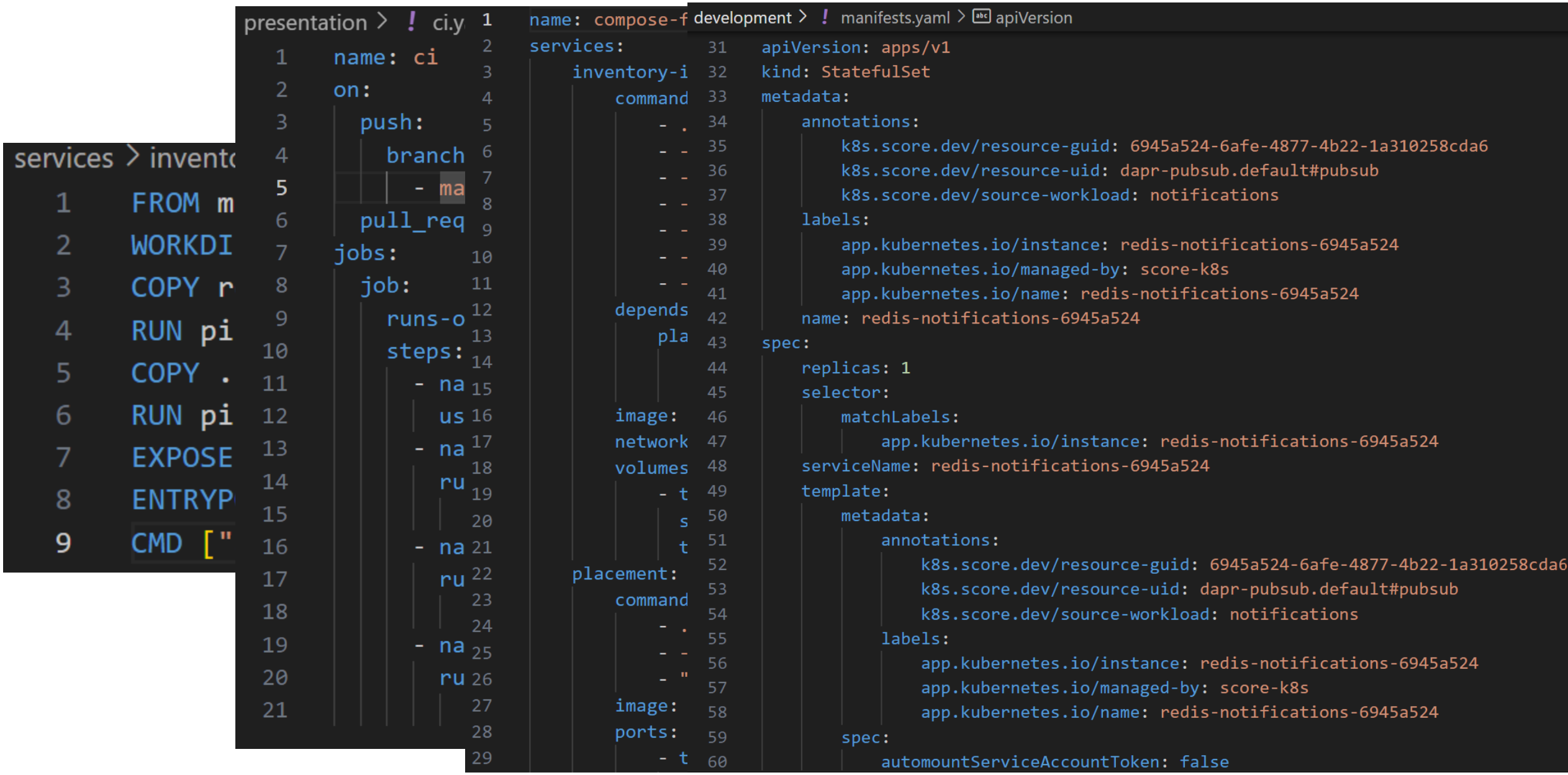
Mathieu
Platform/DevEx Team

How can I avoid the "**It works on my machine**" effect? Can I define integration tests as early as possible in their Continuous Integration pipelines and fail fast if needed?

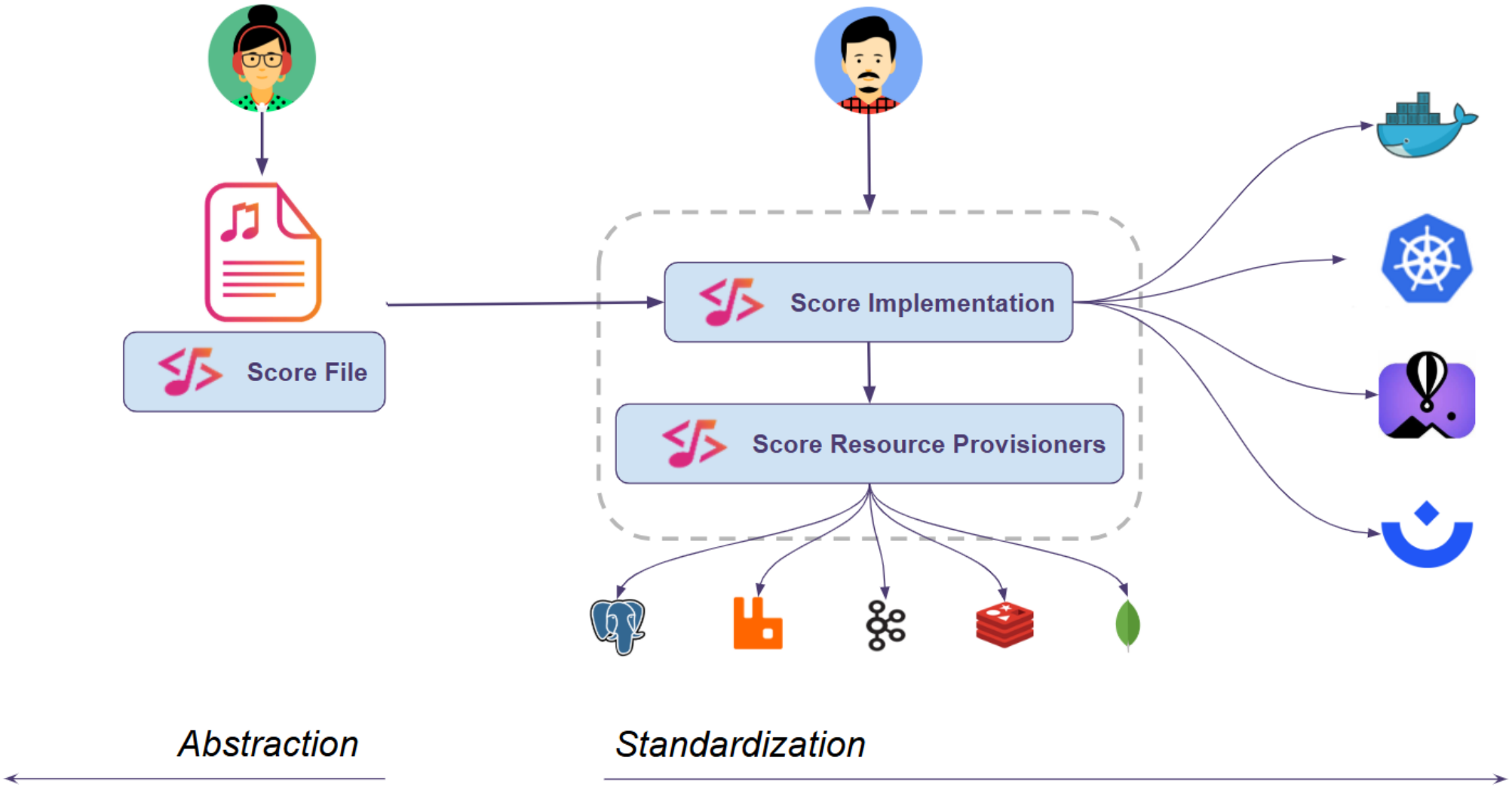
How can I support the Devs with their definition of Components? How can I make sure I will be able to **support them in Production**?

How can I standardize Components recipes **with well supported recipes** to use Enterprise-ready Redis, RabbitMQ, etc.?

Hosting your app on Kubernetes in Production?



Score overview



Standardizing with Score

1

Author Score file

*Describe your **Workload and its dependencies**.*

Agnostic to the Platform/Runtime and the Environment.

score.yaml U

score.yaml > { } resources > { } state-store > abc type

Score schema - Score workload specification (score-v1b1.json)

```yaml
apiVersion: score.dev/v1b1
metadata:
  name: my-app
  annotations:
    dapr.io/enabled: "true"
containers:
  main:
    image: .
    variables:
      APP_PORT: "3002"
      STATESTORE_NAME: "${resources.state-store.name}"
resources:
  state-store:
    type: dapr-state-store
```

Standardizing with Score

1

Author Score file

*Describe your **Workload and its dependencies**.*

Agnostic to the Platform/Runtime and the Environment.

2

Configure resources provisioning templates and Dapr Components for target platform

*Use **Score implementation** and author resource provisioners.*

Score Implementations

```
- op: set
  path: services.placement
  value:
    image: ghcr.io/dapr/placement:latest
    command: ["./placement", "--port", "50006"]
    ports:
    - target: 50006
      published: "50006"
- op: set
  path: services.scheduler
  value:
    image: ghcr.io/dapr/scheduler:latest
    command: ["./scheduler", "--port", "50007", "--etcd-data-dir", "/data"]
    ports:
    - target: 50007
      published: "50007"
    user: root
    volumes:
    - type: bind
      source: ./dapr-etcd-data/
      target: /data
{{ range $name, $cfg := .Compose.services }}
{{ if dig "annotations" "dapr.io/enabled" false $cfg }}
- op: set
  path: services.{{ $name }}-sidecar
  value:
    image: ghcr.io/dapr/daprd:latest
    command: ["./daprd", "--app-id={{ dig "annotations" "dapr.io/app-id" "" $cfg }}", "--app-port=
    network_mode: service:{{ $name }}
    volumes:
    - type: bind
      source: .score-compose/mounts/components/
      target: /components
    depends_on:
      placement:
        condition: service_started
        required: true
{{ end }}
{{ end }}
```

Standardizing with Score

1

Author Score file

*Describe your **Workload and its dependencies**.*

Agnostic to the Platform/Runtime and the Environment.

2

Configure resources provisioning templates and Dapr Components for target platform

*Use Score implementation and author **resource provisioners**.*

Score Resource provisioners

redis-dapr-state-store.yaml U

redis-dapr-state-store.yaml > {} 0 > abc files

```yaml
# Downloaded from https://raw.githubusercontent.com/score-spec/community-
- uri: template://community-provisioners/redis-dapr-state-store
  type: dapr-state-store
  description: Generates a Redis Service and a Dapr StateStore Component
  init: |
    port: 6379
    randomServiceName: redis-{{ randAlphaNum 6 }}
    randomPassword: {{ randAlphaNum 16 | quote }}
  state: |
    serviceName: {{ dig "serviceName" .Init.randomServiceName .State | qu
    password: {{ dig "password" .Init.randomPassword .State | quote }}
  outputs: |
    name: {{ .State.serviceName }}
  files: |
    components/{{ .State.serviceName }}.yaml: |
      apiVersion: dapr.io/v1alpha1
      kind: Component
      metadata:
        name: {{ .State.serviceName }}
      spec:
        type: state.redis
        version: v1
        metadata:
        - name: redisHost
          value: {{ .State.serviceName }}:{{ .Init.port }}
        - name: redisPassword
          value: {{ .State.password }}
  services: |
    {{ .State.serviceName }}:
      labels:
        dev.score.compose.res.uid: {{ .Uid }}
      image: redis:7-alpine
      restart: always
      entrypoint: ["redis-server"]
```

Standardizing with Score

1

Author Score file

Describe your Workload and its dependencies.

2

Configure resources provisioning templates and Dapr Components bindings

Use Score implementation and author resource provisioners.

3

Deploy to a specific Environment and Platform

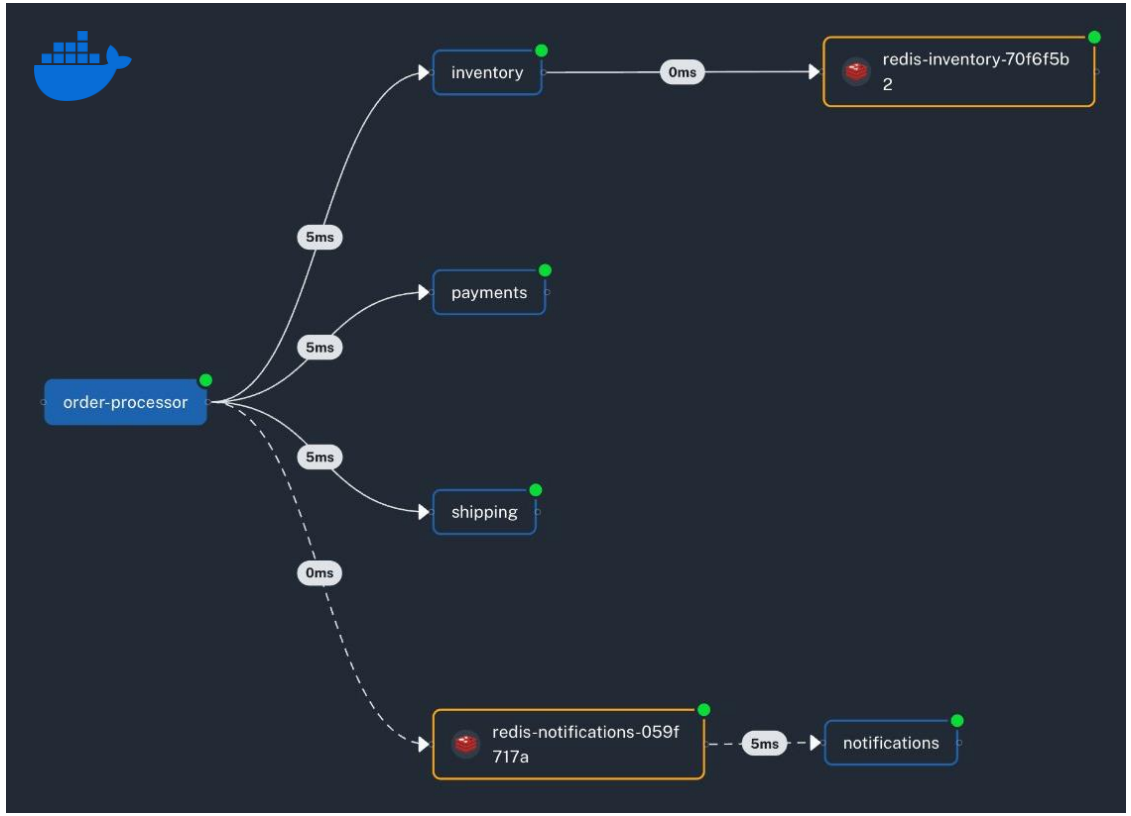
Generate manifests for the Score files based on the Score implementation and provisioners. And trigger the deployment.

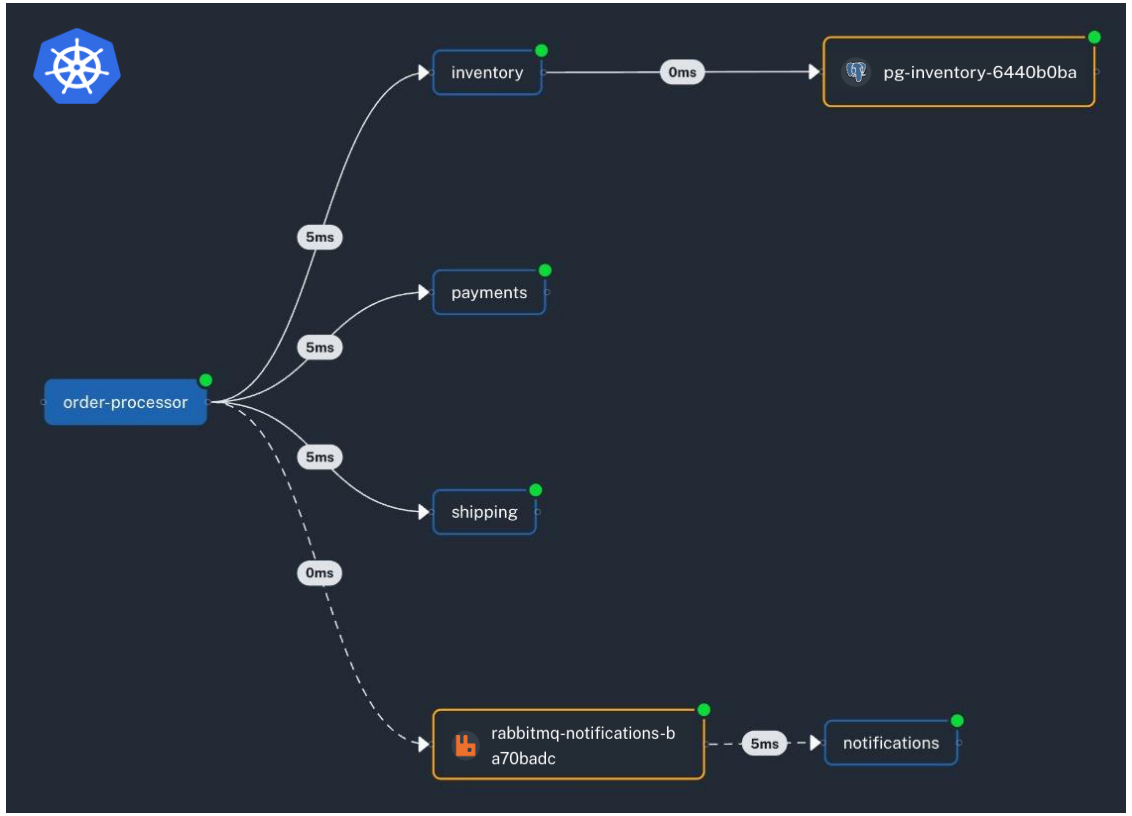
$ deploy-compose.sh U

$ deploy-compose.sh

```
score-compose init \
    --provisioners dapr.yaml \
    --patch-templates dapr-compose.tpl

score-compose generate \
    services/inventory/score.yaml \
    services/notifications/score.yaml \
    services/order-processor/score.yaml \
    services/payments/score.yaml \
    services/shipping/score.yaml

docker compose up \
    --build \
    -d

CONTAINER ID   IMAGE            COMMAND
160e6bc87981   shipping         "python app.py"
1fd80bfb202b   payments         "python app.py"
f857d0eac3fd   inventory        "python app.py"
3f109e8075f7   notifications    "python app.py"
118843c88755   processor        "python app.py"
fcd3b30d7f3d   redis:7-alpine   "redis-server /usr/l…"
81626e9493a4   redis:7-alpine   "redis-server /usr/l…"
ce48349b4615   dapr/placement   "./placement --port …"
837d3ab25629   dapr/daprd       "./daprd --app-id=sh…"
837d3ab25629   dapr/daprd       "./daprd --app-id=pa…"
837d3ab25629   dapr/daprd       "./daprd --app-id=in…"
837d3ab25629   dapr/daprd       "./daprd --app-id=no…"
837d3ab25629   dapr/daprd       "./daprd --app-id=pr…"
```

Demo: Deploying Dapr App using Score

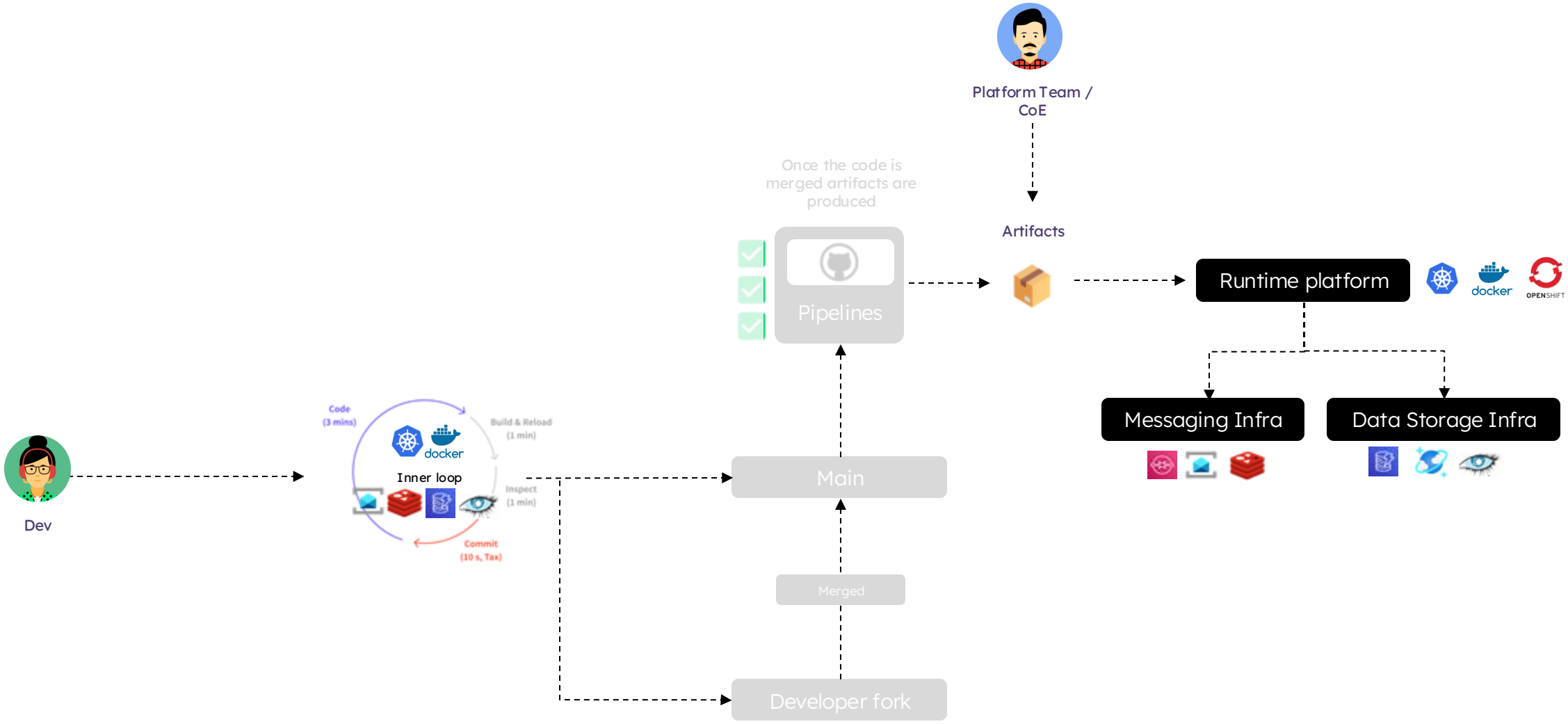




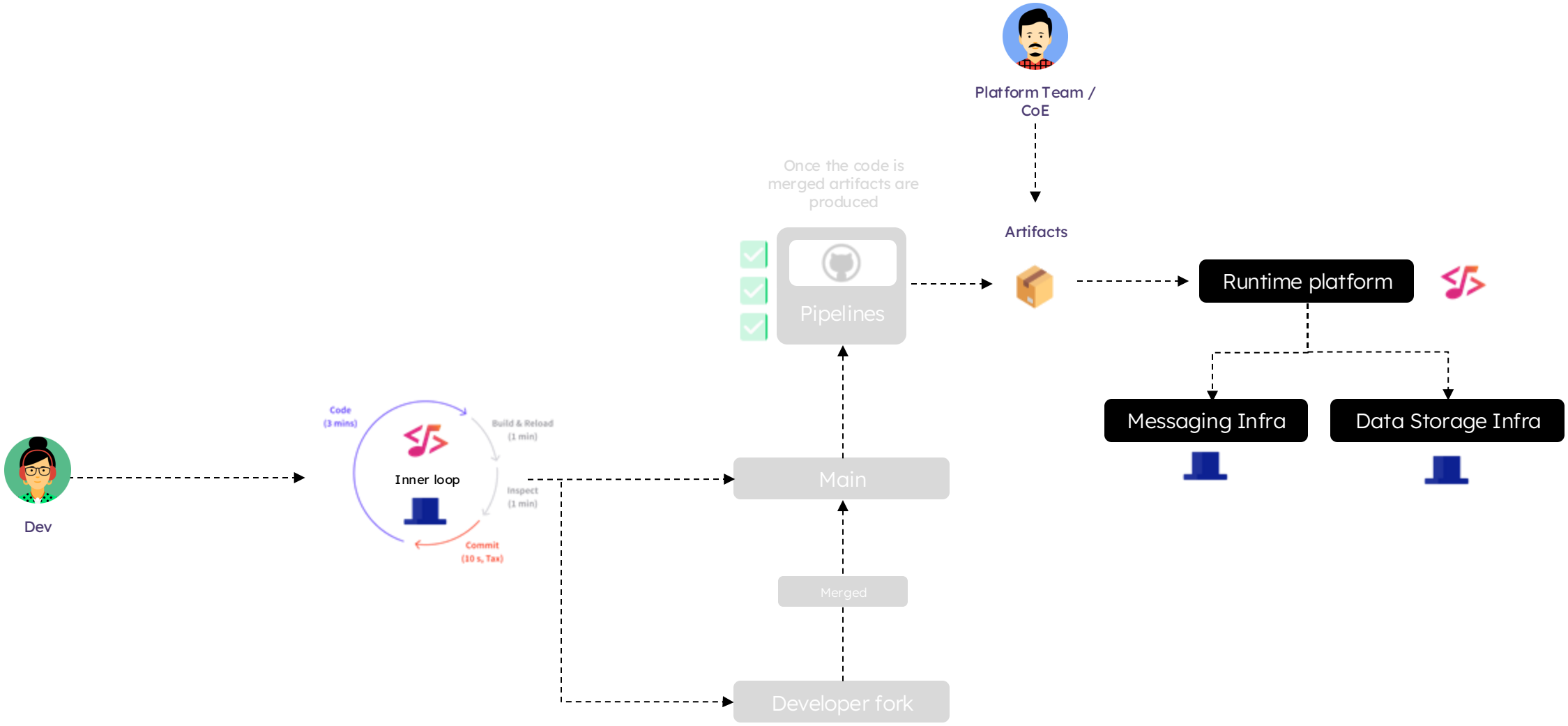
That’s a wrap!

KubeCon | CloudNativeCon
Europe 2025

Platform and infrastructure concerns are **increasing cognitive load** and **decreasing productivity** for developers

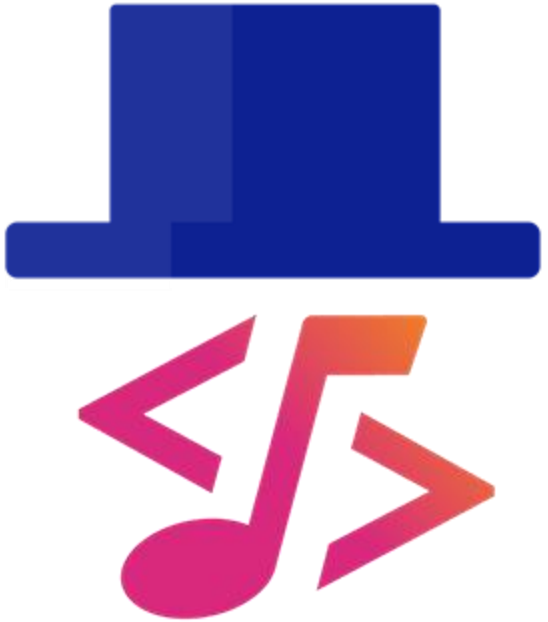


Solving complexity with consistent abstraction interfaces



Next steps

- Demo's source code: kendallroden/kubecton-dapr-score-demo
- **Dapr Kiosk: 8A**
  - *More about Dapr at KubeCon.*
- **Score Kiosk: 2A** (Thu April 3rd - 10:30-13:30)
  - *More about Score at KubeCon.*



Evaluate the session, please!

KubeCon | CloudNativeCon
Europe 2025


KubeCon
CloudNativeCon
Europe 2025
LONDON BUS