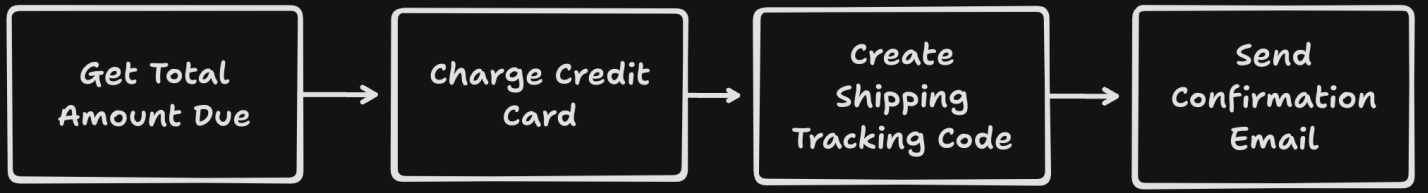
Durable Workflows

with Effect Cluster

THANK YOU ORGANIZERS!

Ego Slide

- Push buttons
- Type things
- Break stuff
- Wear sweatshirt as cape
- This picture is not stack safe


Get Total Amount Due
Charge Credit Card
Create Shipping Tracking Code
Send Confirmation Email

Failures along the way

```
function chargeCreditCard(cardNumber, totalAmount): Effect<void, InsufficientFundsError | PaymentGatew
function createShippingTrackingCode(deliveryAddress): Effect<TrackingId, NoMoreApiCallQuotaError>
function sendConfirmationEmail(email, orderId, trackingId): Effect<void, SmtpFailureError>

const processPayment: Effect<
    void,
    InsufficientFundsError | PaymentGateway503Error | NoMoreApiCallQuotaError | SmtpFailureError
  > = ...
```

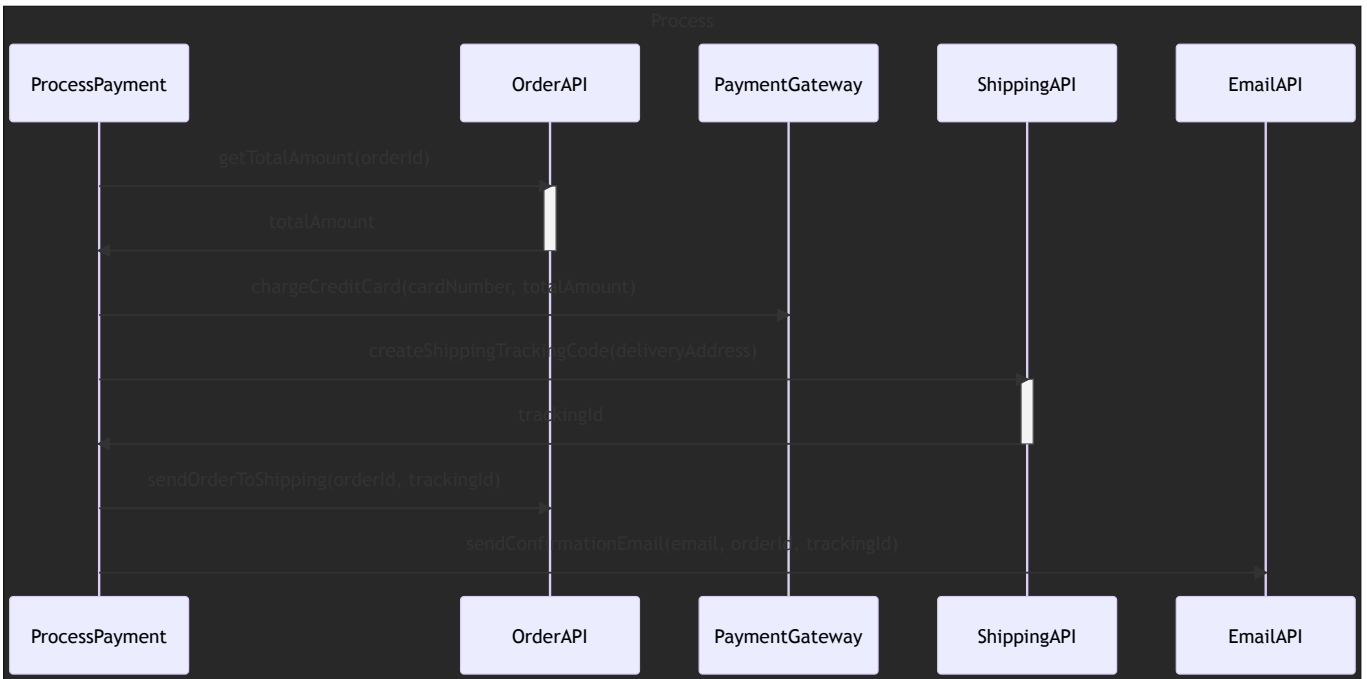
Log on failure

```
const processPayment =
  (cardNumber: CardNumber, deliveryAddress: DeliveryAddress,
  email: EmailAddress, orderId: OrderId) ⇒
    Effect.gen(function*($) {
      const totalAmount = yield* $(getTotalAmount(orderId))
      yield* $(chargeCreditCard(cardNumber, totalAmount))
      const trackingId = yield* $(createShippingTrackingCode(deliveryAddress))
      // ^- failure raised here!
      yield* $(sendOrderToShipping(orderId, trackingId)) // skipped
      yield* $(sendConfirmationEmail(email, orderId, trackingId)) // skipped
    })
```

Retrying everything

```
const processPayment =
  (cardNumber: CardNumber, deliveryAddress: DeliveryAddress,
  email: EmailAddress, orderId: OrderId) ⇒
    Effect.gen(function*($) {
      const totalAmount = yield* $(getTotalAmount(orderId))
      yield* $(chargeCreditCard(cardNumber, totalAmount))
      const trackingId = yield* $(createShippingTrackingCode(deliveryAddress))
      // ^- failure raised here!
      yield* $(sendOrderToShipping(orderId, trackingId)) // skipped
      yield* $(sendConfirmationEmail(email, orderId, trackingId)) // skipped
    }).pipe(
      Effect.retry({
        while: error ⇒ isTemporaryError(error)
      })
    )
```

Business Process



Transactions?

```
BEGIN TRANSACTION;
UPDATE card_balances SET balance = balance - 10 WHERE card_number = 42
UPDATE orders SET tracking_id = 'abc' WHERE order_id = 12
COMMIT TRANSACTION;
```

Distributed systems are everywhere

Distributed Workflows are everywhere

Welcome Effect Cluster

All the building blocks you need to deal with distributed workflows with ease!

SAGAS

Hector Garcia-Molina
Kenneth Salem

Department of Computer Science
Princeton University
Princeton, N J 08544

…long lived transactions?

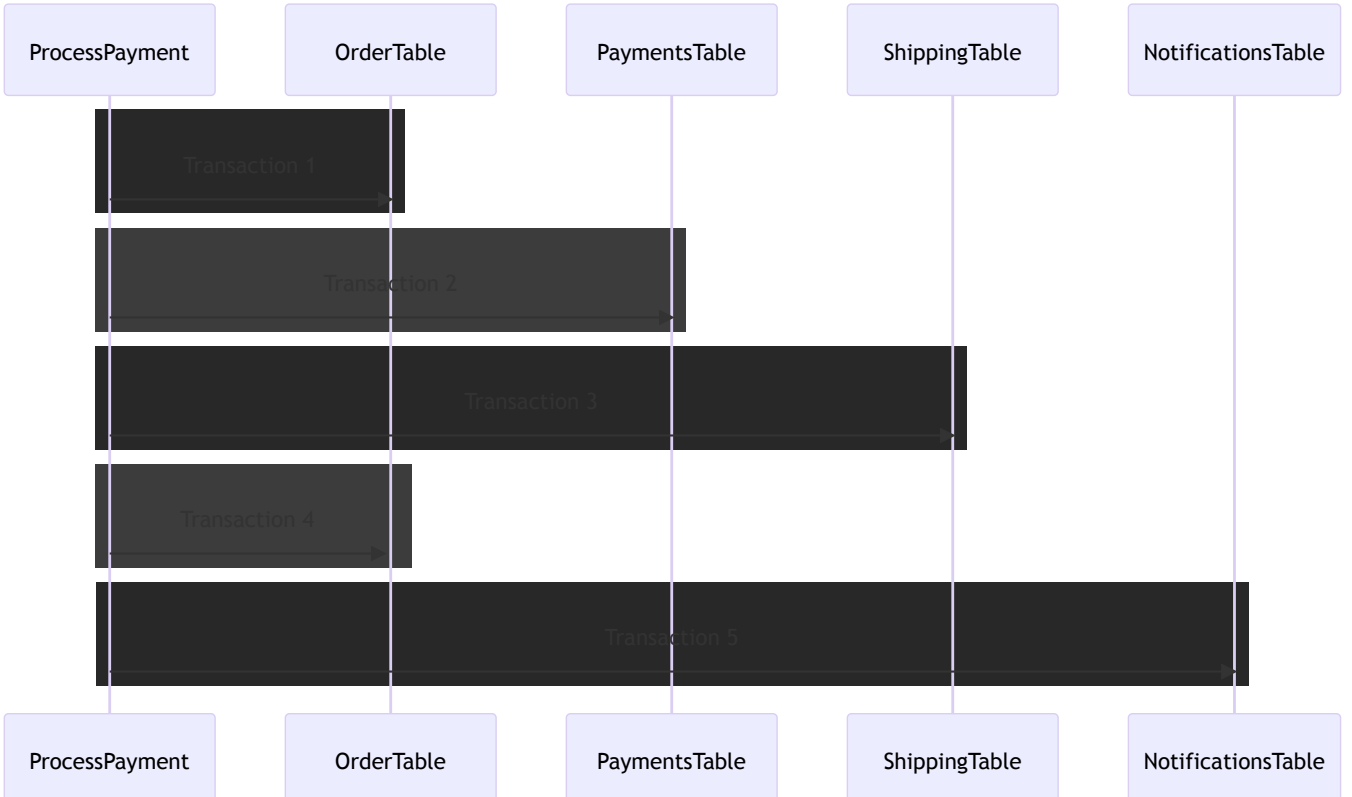
- Sagas!
- in 1987 they had computers!

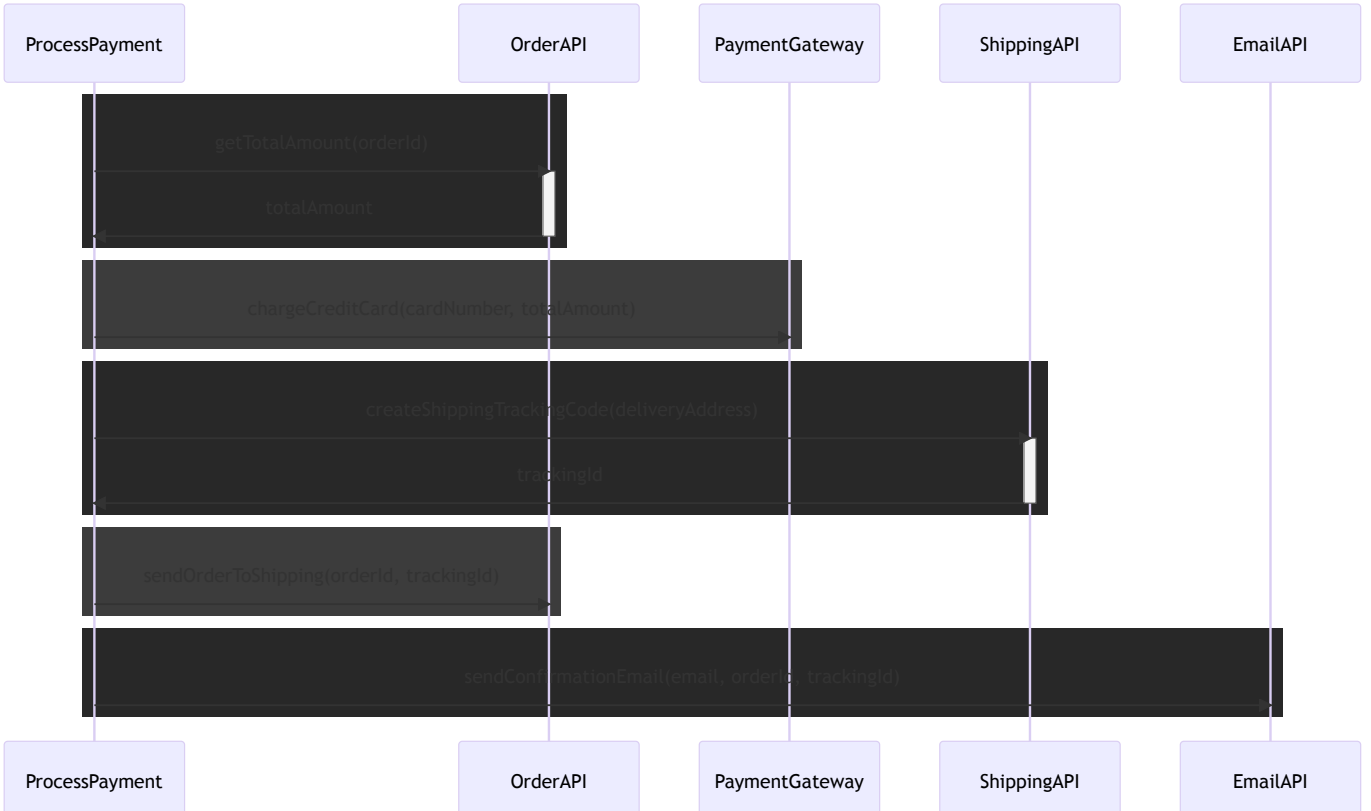
bstract

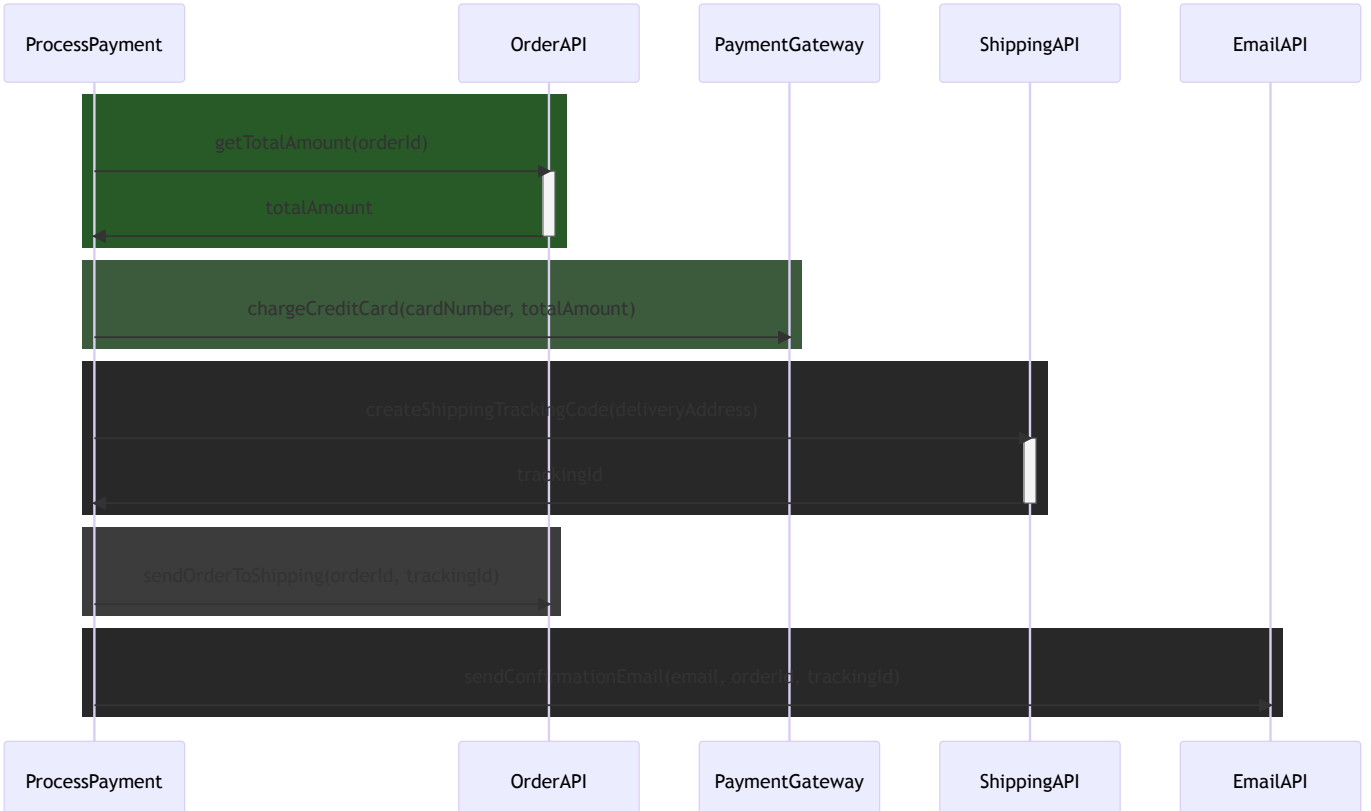
sactions (LLTs) hold on to
or relatively long periods of
elaying the termination of
common transactions To
ms we propose the notion of
aga if it can be written as a
ons that can be interleaved
ons The database manage-
ees that either all the tran-
re successfully completed or
ctions are run to amend a
oth the concept of saga and

the majority of other tran
it accesses many database
computations, it pauses fo
or a combination of these
LLTs are transactions
account statements at a
process claims at an ins
transactions to collect st
database [Gray81a]

In most cases, LLTs
mance problems Since th
system must execute them
preserving the cor


ProcessPayment
OrderTable
PaymentsTable
ShippingTable
NotificationsTable
Transaction 1
Transaction 2
Transaction 3
Transaction 4
Transaction 5
ProcessPayment
OrderTable
PaymentsTable
ShippingTable
NotificationsTable


ProcessPayment
OrderAPI
PaymentGateway
ShippingAPI
EmailAPI
getTotalAmount(orderId)
totalAmount
chargeCreditCard(cardNumber, totalAmount)
createShippingTrackingCode(deliveryAddress)
trackingId
sendOrderToShipping(orderId, trackingId)
sendConfirmationEmail(email, orderId, trackingId)
ProcessPayment
OrderAPI
PaymentGateway
ShippingAPI
EmailAPI


ProcessPayment
OrderAPI
PaymentGateway
ShippingAPI
EmailAPI
getTotalAmount(orderId)
totalAmount
chargeCreditCard(cardNumber, totalAmount)
createShippingTrackingCode(deliveryAddress)
trackingId
sendOrderToShipping(orderId, trackingId)
sendConfirmationEmail(email, orderId, trackingId)
ProcessPayment
OrderAPI
PaymentGateway
ShippingAPI
EmailAPI

Defining an activity

- Work unit of a Workflow
- Can interact with external systems
- Executes an Effect
- Uniquely identified inside Workflow
- Requires schemas for success and failure

```
const getTotalAmount = (id: string) =>
Activity.make(
    `get-amount-due-${id}`, // identifier
    Schema.number, // success schema
    Schema.struct({ code: Schema.number,
      message: Schema.string }) // error schema
  )(
    pipe(
      Http.request.get(`/get-total-amount/${id}`)
      Http.client.fetchOk(),
      Effect.andThen((response) => response.json),
      Effect.mapError(() => ({
        code: 500,
        message: `API Fetch error`
        })
      )
    )
  )
```

Defining a workflow

- Is started by a Request
- Requires schemas for success and failure
- Has a payload of information

```
class ProcessPaymentRequest extends
  Schema.TaggedRequest<ProcessPaymentRequest>()(
  `ProcessPaymentRequest`, // tag
  Schema.never, // failure
  Schema.boolean, // success
  {
    orderId: Schema.string,
    cardNumber: Schema.string,
    email: Schema.string,
    deliveryAddress: Schema.string
  }
) {
}
```

Defining a workflow

```
const processPaymentWorkflow = Workflow.make(
  ProcessPaymentRequest,
  (_) => `ProcessPayment@${_.orderId}`,
  ({ cardNumber, deliveryAddress, email, orderId }) =>
    Effect.gen(function*($) {
      const totalAmount = yield* $(getTotalAmount(orderId))
      yield* $(chargeCreditCard(cardNumber, totalAmount))
      const trackingId = yield* $(createShippingTrackingCode(deliveryAddress))
      yield* $(sendOrderToShipping(orderId, trackingId))
      yield* $(sendConfirmationEmail(email, orderId, trackingId))
    })
)
```

- Coordinator of activities
- Durable execution
- Requires deterministic code

WTF is deterministic?

Given a set of input, the output of the function must be always the same and predictable, without triggering any side effects that may later affect the computation.

Deterministic

- Math & Logic ops
- Seed based random

Non-Deterministic

- Math.random()
- new Date()
- R/W Global Shared State (also DB)

Determinism & Workflows

```
const processPaymentWorkflow = Workflow.make(
  ProcessPaymentRequest,
  (_) ⇒ "ProcessPayment@" + _.orderId,
  ({ cardNumber, deliveryAddress, email, orderId }) ⇒
    Effect.gen(function*($) {
      const totalAmount = yield* $(Effect.succeed(42.1) /* getTotalAmount(orderId) */)
      yield* $(Effect.unit /* chargeCreditCard(cardNumber, totalAmount) */)
      const trackingId = yield* $(createShippingTrackingCode(deliveryAddress))
      yield* $(sendOrderToShipping(orderId, trackingId))
      yield* $(sendConfirmationEmail(email, orderId, trackingId))
    })
)
```

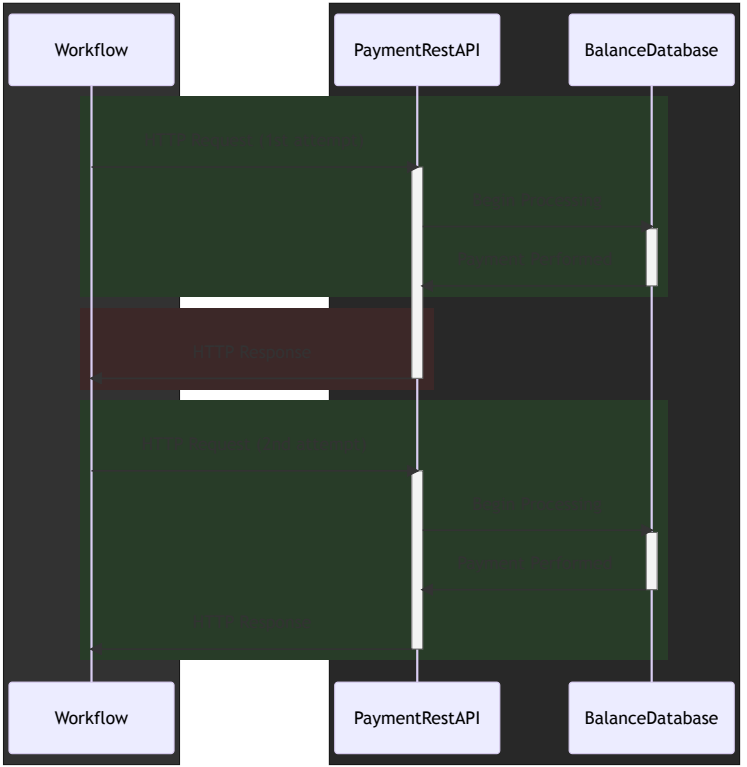
Running a Workflow

```
const main = Effect.gen(function*($) {
  const workflows = Workflow.union(processPaymentWorkflow, requestRefundWorkflow)
  const engine = yield* $(WorkflowEngine.makeScoped(workflows))
  yield* $(
    engine.sendDiscard(
      new ProcessPaymentRequest({
        orderId: "order-1",
        cardNumber: "my-card",
        deliveryAddress: "My address, 5, Italy",
        email: "my@email.com"
      })
    )
  )
})
runMain(
  pipe(
    main,
    Effect.provide(DurableExecutionJournalPostgres.DurableExecutionJournalPostgres)
  )
)
```

Activity is a Black Box

The double-payment problem

- Any transport may fail
- Not getting a response does not mean it has not been processed

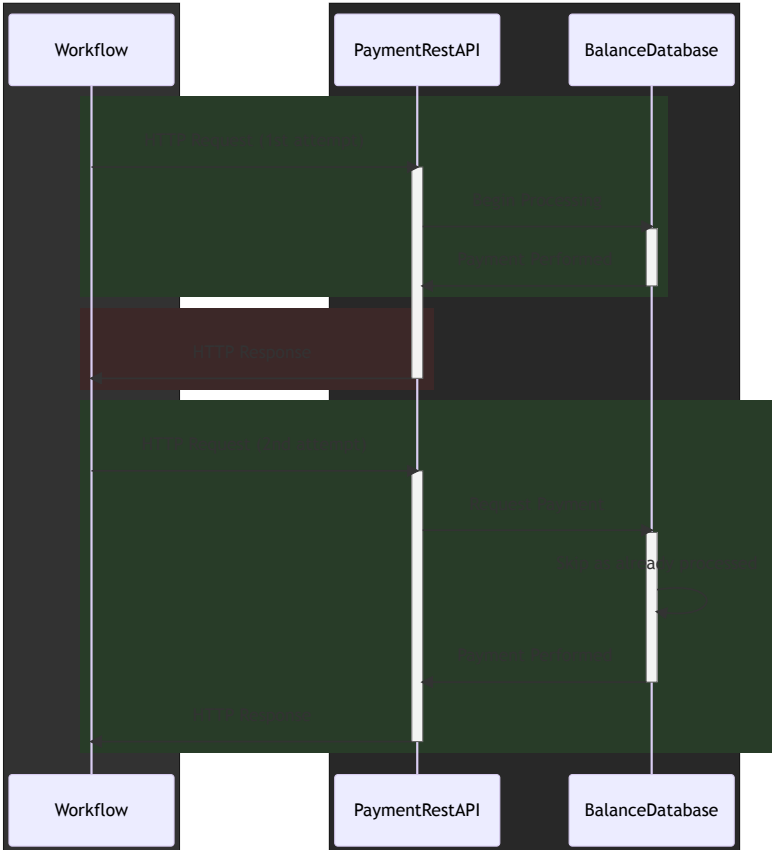


Idempotency

Multiple invocation of the same function, result on a state change as they were executed only the first time.

Idempotence using IDs

```
const chargeCreditCard =
 (cardNumber: string, amountDue: number) ⇒
  pipe(
    Activity.persistenceId,
    Effect.flatMap(
      (persistenceId) ⇒ callPaymentGateway(
        persistenceId,
        cardNumber,
        amountDue
      )
    ),
    Activity.make(
      "charge-credit-card",
      Schema.void,
      Schema.never
    )
  )
```



```ts
const createShippingTrackingCode = (deliveryAddress: string) =>
  pipe(
    ShipmentApiConfig,
    Effect.flatMap((apiConfig) => callShipmentApi(apiConfig.apiSecret, apiConfig.apiSecret, deliveryAddress)),
    Activity.make("create-tracking-code", Schema.string, Schema.never)
  )
```

cluster-pg — node /opt/homebrew/bin/pnpm example examples/simple-workflow.ts — 144×22

```
mattiamanzati@MacBook-Pro-di-Mattia-2 cluster-pg % pnpm example examples/simple-workflow.ts
```

Yielding workflow execution

```
const getTotalAmount = (id: string) =>
  pipe(
    Http.request.get(`/get-total-amount/${id}`)
    .pipe(
      Http.client.fetchOk(),
      Effect.andThen((response) => response.json),
      Effect.retry(
        Schedule.exponential(1000).pipe(
          Schedule.compose(Schedule.recurs(32)),
        ),
      ),
      Effect.catchAllCause(() => pipe(
        Effect.logError("Something is wrong with the OrderAPI right now"),
        Effect.zipRight(Workflow.yieldExecution)
      ))
    ),
    Activity.make(`get-amount-due-${id}`, Schema.number, Schema.never)
  )
```

Fixing workflows

```
const processPaymentWorkflow = Workflow.make(
  ProcessPaymentRequest,
  (_) ⇒ `ProcessPayment@${_.orderId}`,
  ({ cardNumber, deliveryAddress, email, orderId }) ⇒
    Effect.gen(function*($) {
      const version = yield* $(getCurrentVersion(2))
      // ^- for already running WFs, will resolve with previous value (1)
      // ...
      if(version ⩾ 2){
        yield* $(checkTrustedCardNumber(cardNumber))
      }
      yield* $(chargeCreditCard(cardNumber, totalAmount))
      // ...
    })
)

const getCurrentVersion = (definitionVersion: number) ⇒ pipe(
  Effect.succeed(definitionVersion),
  Activity.make("get-current-version", Schema.number, Schema.void)
)
```

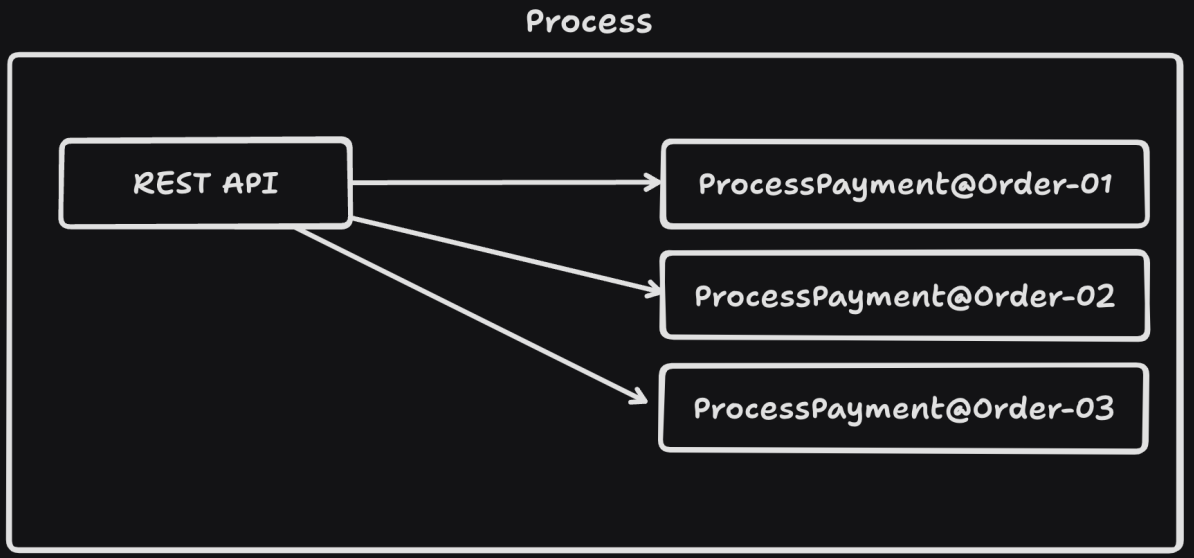
All or Nothing

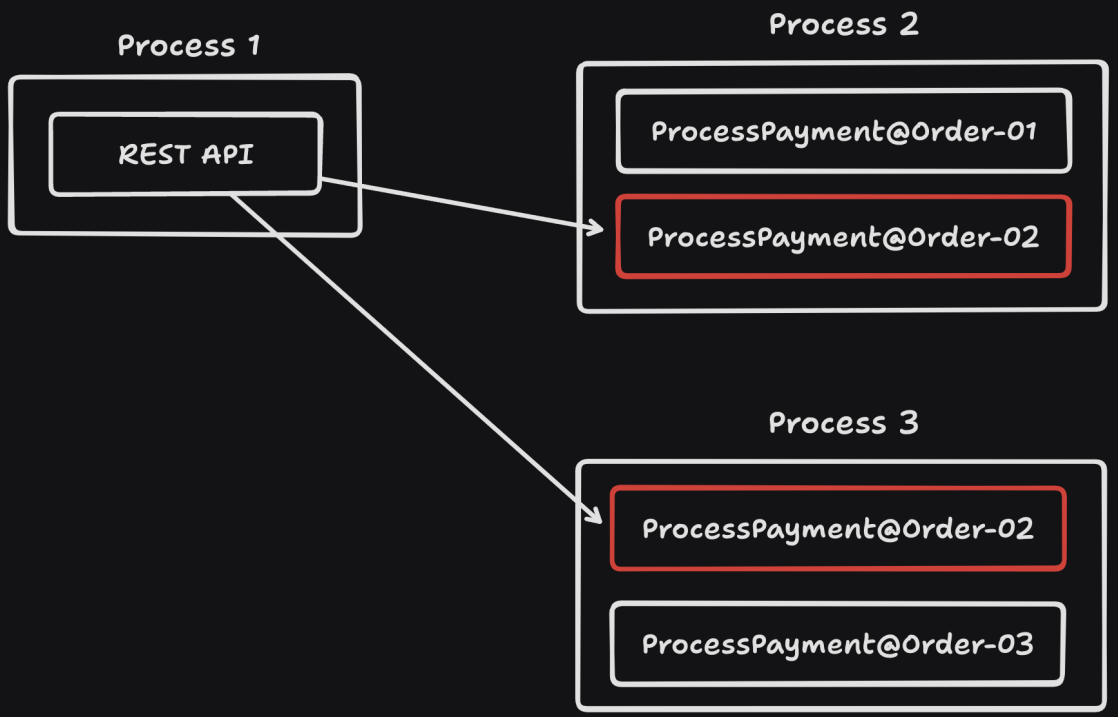
Compensating actions

```
const processPaymentWorkflow = Workflow.make(
  ProcessPaymentRequest,
  (_) ⇒ `ProcessPayment@${_.orderId}`,
  ({ cardNumber, deliveryAddress, email, orderId }) ⇒
    pipe(
      getTotalAmount(orderId),
      Effect.flatMap(totalAmount ⇒ pipe(
        chargeCreditCard(cardNumber, totalAmount),
        Effect.flatMap(() ⇒ createShippingTrackingCode(deliveryAddress)),
        Effect.tap(trackingId ⇒ sendOrderToShipping(orderId, trackingId)),
        Effect.tap(trackingId ⇒ sendConfirmationEmail(email, orderId, trackingId)),
        Effect.catchTag("OutOfStockError", () ⇒ refundCreditCard(cardNumber, totalAmount))
      ))
    )
)
```

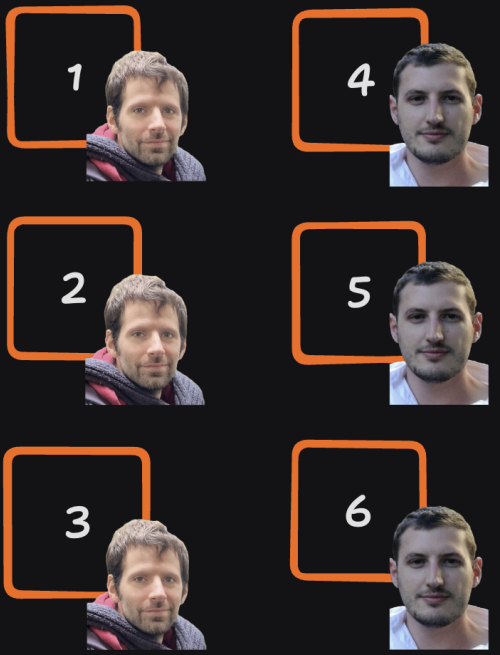
Effect Cluster Workflows

Building durable and reliable Effects for your applications


Process
REST API
ProcessPayment@Order-01
ProcessPayment@Order-02
ProcessPayment@Order-03


Process 1
REST API
Process 2
ProcessPayment@Order-01
ProcessPayment@Order-02
Process 3
ProcessPayment@Order-02
ProcessPayment@Order-03


1
4
2
5
3
6


1
4
2
5
3
6

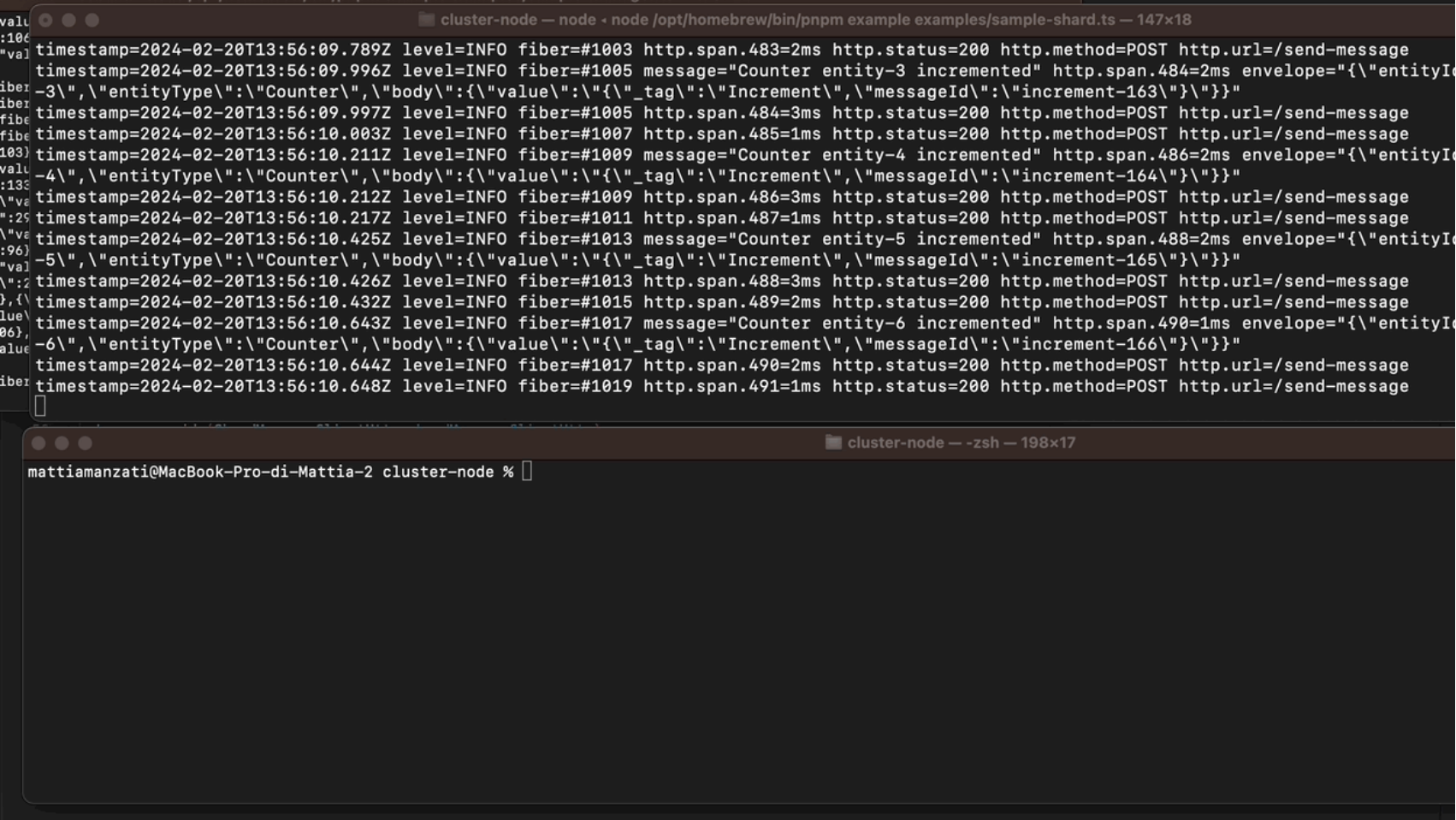

1
4
2
5
3
6


RIP


1
4
2
5
3
6

Effect Cluster Sharding and Location Transparency

Safely distribute work and refer to entity without knowing their location


cluster-node — node ◂ node /opt/homebrew/bin/pnpm example examples/sample-shard.ts — 147×18
timestamp=2024-02-20T13:56:09.789Z level=INFO fiber=#1003 http.span.483=2ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:09.996Z level=INFO fiber=#1005 message="Counter entity-3 incremented" http.span.484=2ms envelope="{\"entityId
-3\",\"entityType\":\"Counter\",\"body\":{\"value\":\"{\"_tag\":\"Increment\",\"messageId\":\"increment-163\"}\"}}"
timestamp=2024-02-20T13:56:09.997Z level=INFO fiber=#1005 http.span.484=3ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:10.003Z level=INFO fiber=#1007 http.span.485=1ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:10.211Z level=INFO fiber=#1009 message="Counter entity-4 incremented" http.span.486=2ms envelope="{\"entityId
-4\",\"entityType\":\"Counter\",\"body\":{\"value\":\"{\"_tag\":\"Increment\",\"messageId\":\"increment-164\"}\"}}"
timestamp=2024-02-20T13:56:10.212Z level=INFO fiber=#1009 http.span.486=3ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:10.217Z level=INFO fiber=#1011 http.span.487=1ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:10.425Z level=INFO fiber=#1013 message="Counter entity-5 incremented" http.span.488=2ms envelope="{\"entityId
-5\",\"entityType\":\"Counter\",\"body\":{\"value\":\"{\"_tag\":\"Increment\",\"messageId\":\"increment-165\"}\"}}"
timestamp=2024-02-20T13:56:10.426Z level=INFO fiber=#1013 http.span.488=3ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:10.432Z level=INFO fiber=#1015 http.span.489=2ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:10.643Z level=INFO fiber=#1017 message="Counter entity-6 incremented" http.span.490=1ms envelope="{\"entityId
-6\",\"entityType\":\"Counter\",\"body\":{\"value\":\"{\"_tag\":\"Increment\",\"messageId\":\"increment-166\"}\"}}"
timestamp=2024-02-20T13:56:10.644Z level=INFO fiber=#1017 http.span.490=2ms http.status=200 http.method=POST http.url=/send-message
timestamp=2024-02-20T13:56:10.648Z level=INFO fiber=#1019 http.span.491=1ms http.status=200 http.method=POST http.url=/send-message
cluster-node — -zsh — 198×17
mattiamanzati@MacBook-Pro-di-Mattia-2 cluster-node %

```ts
import * as RecipientType from "@effect/cluster/RecipientType"
import * as Schema from "@effect/schema/Schema"

export class Increment extends Schema.TaggedClass<Increment>()("Increment", {
  messageId: Schema.string
}) {}

export class Decrement extends Schema.TaggedClass<Decrement>()("Decrement", {
  messageId: Schema.string
}) {}

export class GetCurrent extends
  Schema.TaggedRequest<GetCurrent>()("GetCurrent", Schema.never, Schema.number, {
  messageId: Schema.string
}) {}

export const CounterMsg = Schema.union(Increment, Decrement, GetCurrent)
export type CounterMsg = Schema.Schema.To<typeof CounterMsg>

export const CounterEntity = RecipientType.makeEntityType("Counter", CounterMsg, (_) => _.messageId)
```

```
Sharding.registerEntity(
  CounterEntity
)(
  RecipientBehaviour.fromFunctionEffectStateful(
    () ⇒ Effect.succeed(0),
    (_, message, stateRef) ⇒ {
      switch (message._tag) {
        case "Increment":
          return pipe(Ref.update(stateRef, (count) ⇒ count + 1), Effect.as(MessageState.Processed(Opt
        case "Decrement":
          return pipe(Ref.update(stateRef, (count) ⇒ count - 1), Effect.as(MessageState.Processed(Opt
        case "GetCurrent":
          return pipe(
            Ref.get(stateRef),
            Effect.exit,
            Effect.map((result) ⇒ MessageState.Processed(Option.some(result)))
          )
      }
    }
  )
)
```

Today's Recap

Effect cluster provides all the basic building blocks you'll need to build distributed workflows and systems

`@effect/cluster`

- Sharding
- Shard Manager
- Messaging utilities

`@effect/cluster-node`

- Cluster specific utilities for NodeJS

`@effect/cluster-workflow`

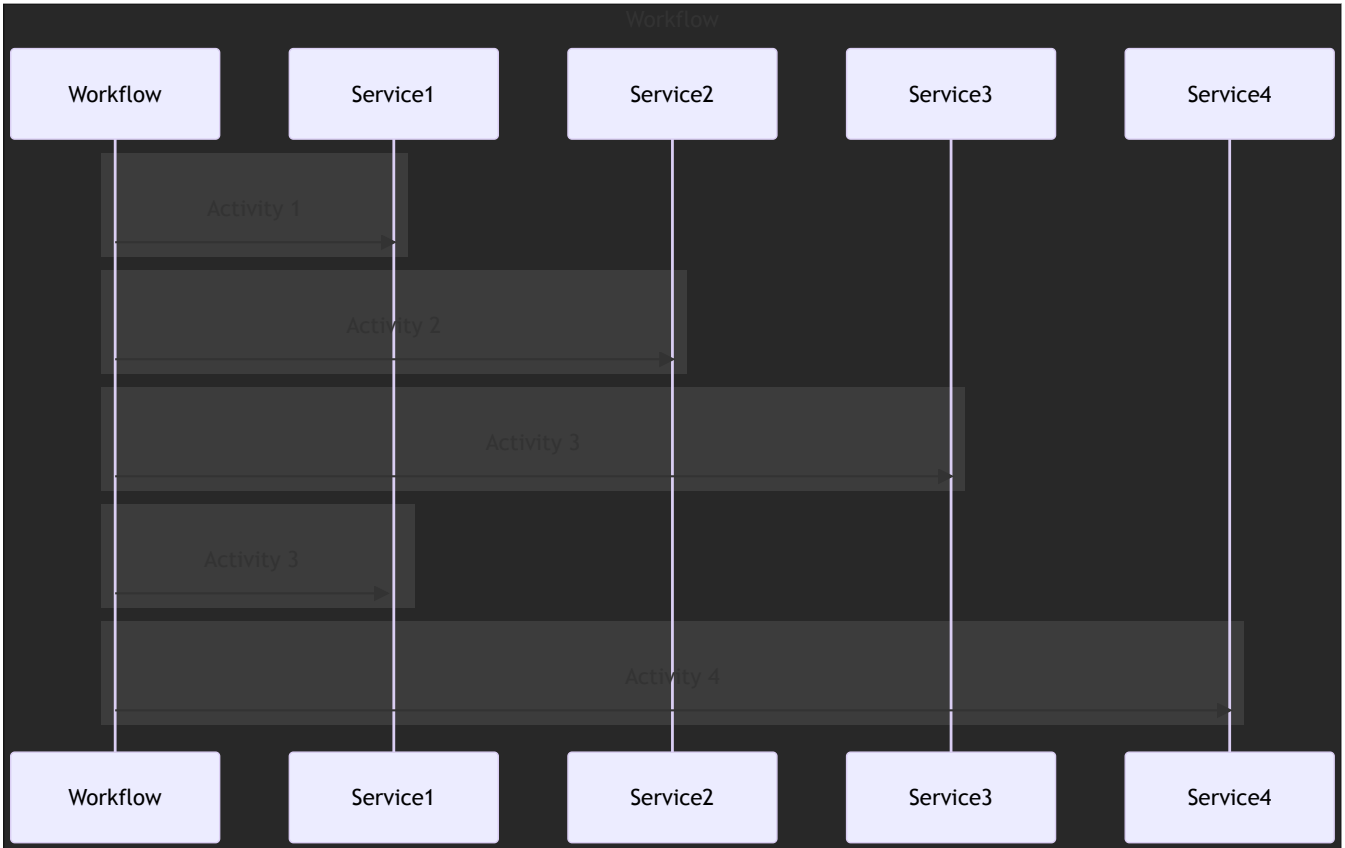
- Activity
- Workflow
- WorkflowEngine

`@effect/cluster-pg`

- Postgres based persistency for Workflow and Sharding

Happy Effect-ing!

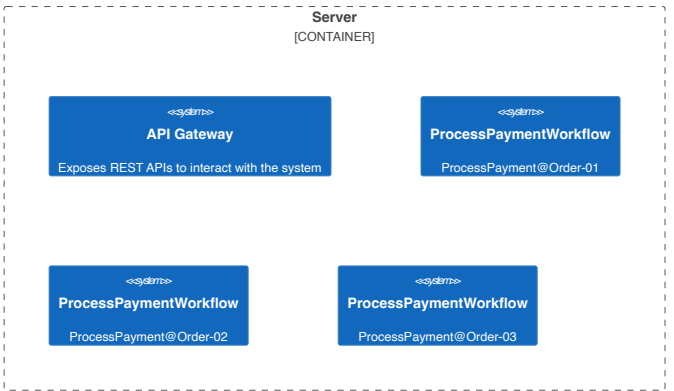
Thanks for your time!


Workflow
Workflow
Service1
Service2
Service3
Service4
Activity 1
Activity 2
Activity 3
Activity 3
Activity 4
Workflow
Service1
Service2
Service3
Service4

Everyday Code

```
const processPayment =
  (cardNumber: CardNumber, deliveryAddress: DeliveryAddress,
  email: EmailAddress, orderId: OrderId) ⇒
    Effect.gen(function*($) {
      // get total order amount
      const totalAmount = yield* $(getTotalAmount(orderId))
      // charge the credit card
      yield* $(chargeCreditCard(cardNumber, totalAmount))
      // create a tracking id
      const trackingId = yield* $(createShippingTrackingCode(deliveryAddress))
      // send the order to shipment
      yield* $(sendOrderToShipping(orderId, trackingId))
      // send a confirmation email
      yield* $(sendConfirmationEmail(email, orderId, trackingId))
    })
```

Scaling the system



Ensuring only one execution of each workflow instance