

# The Very Hungry Transaction

---

**DANIEL COLSON**

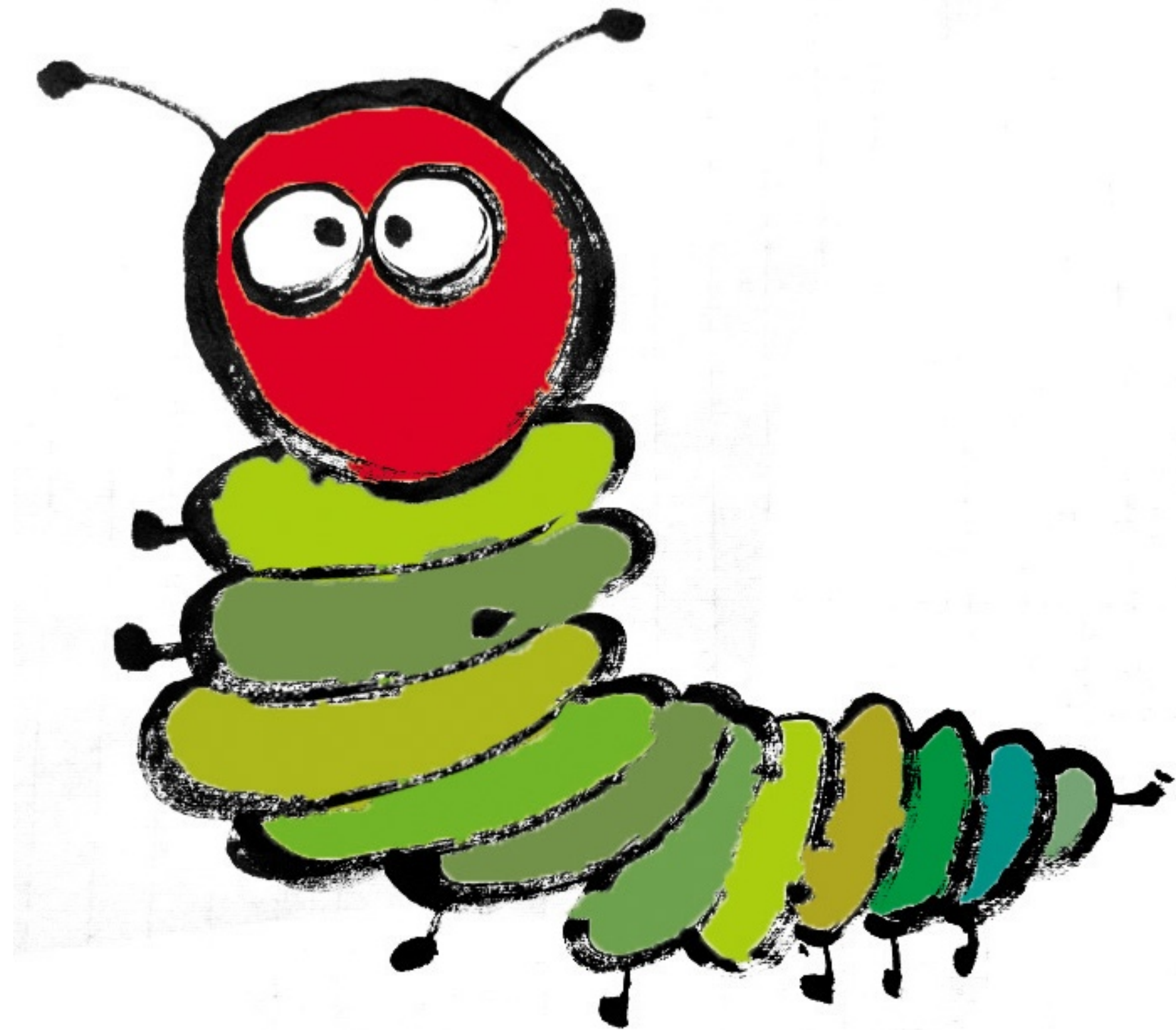
Senior Software Engineer

**GitHub**

# Cat R. Pillar

---

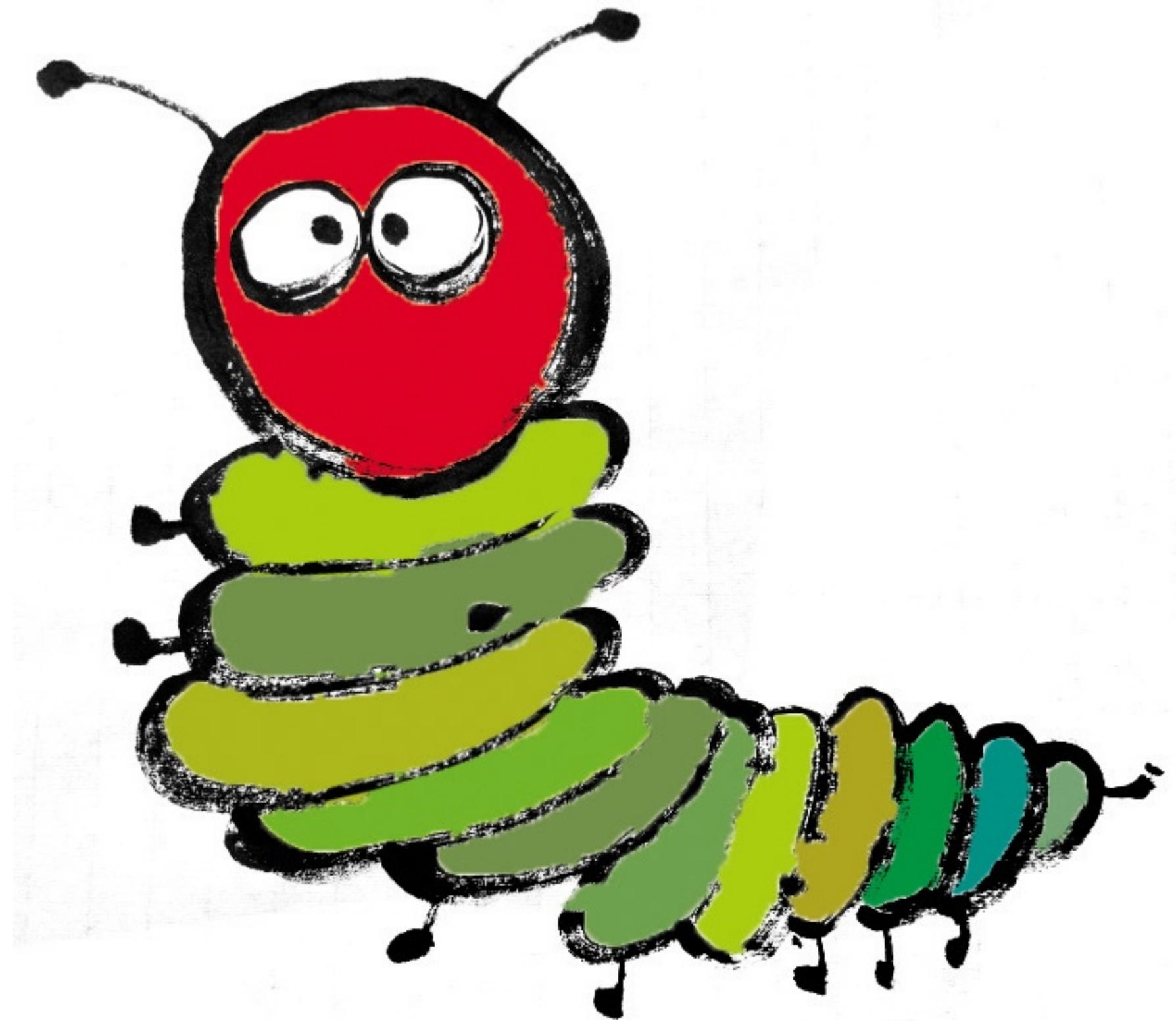
- Job — Developer at BugHub
- Dislikes — Birds, Software Bugs
- Likes — Leaves, Actual Bugs



# BugHub

---

- Large Online Community of Bugs
- More Bugs Than Any Other Site

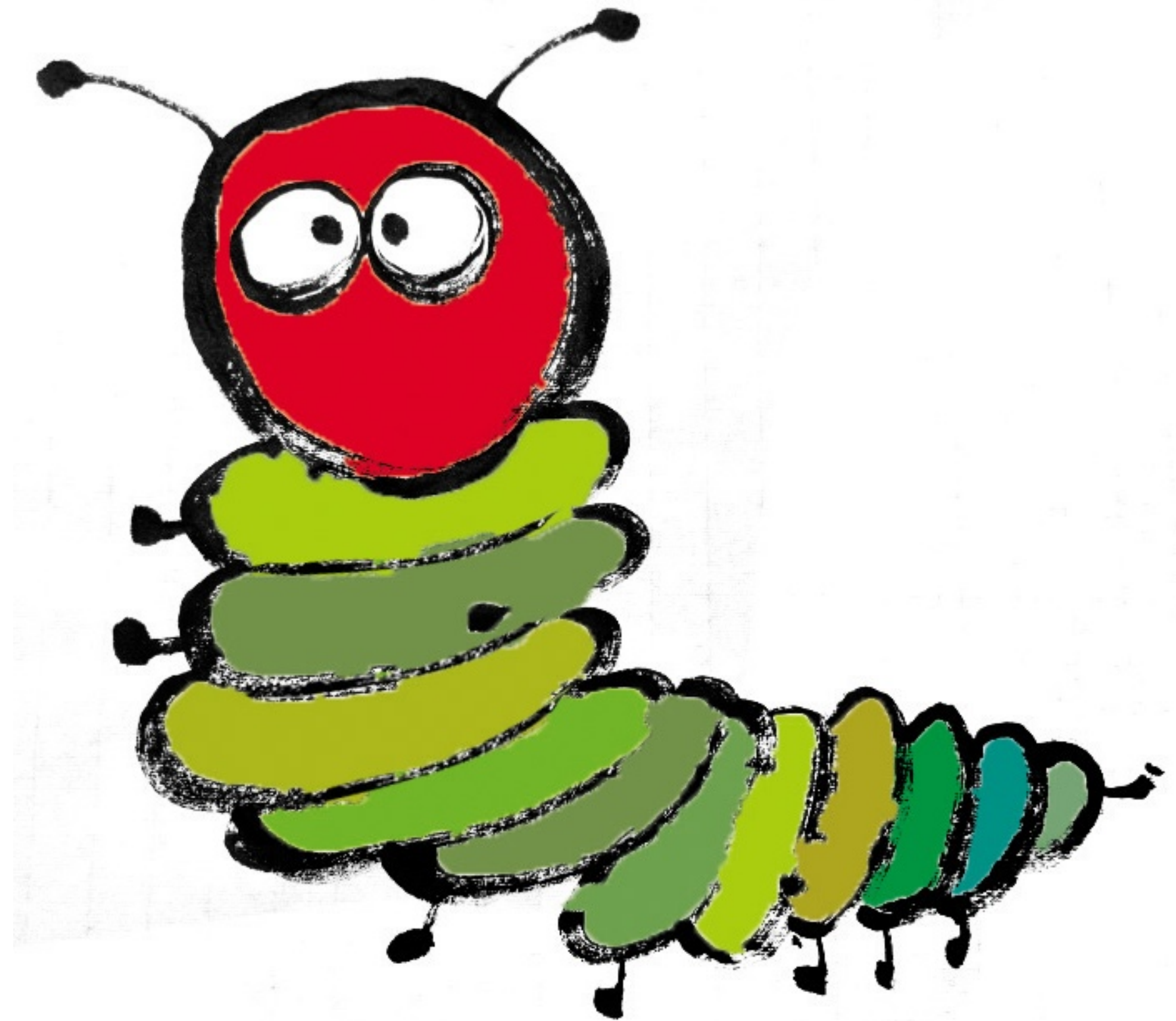




```
transaction do
```

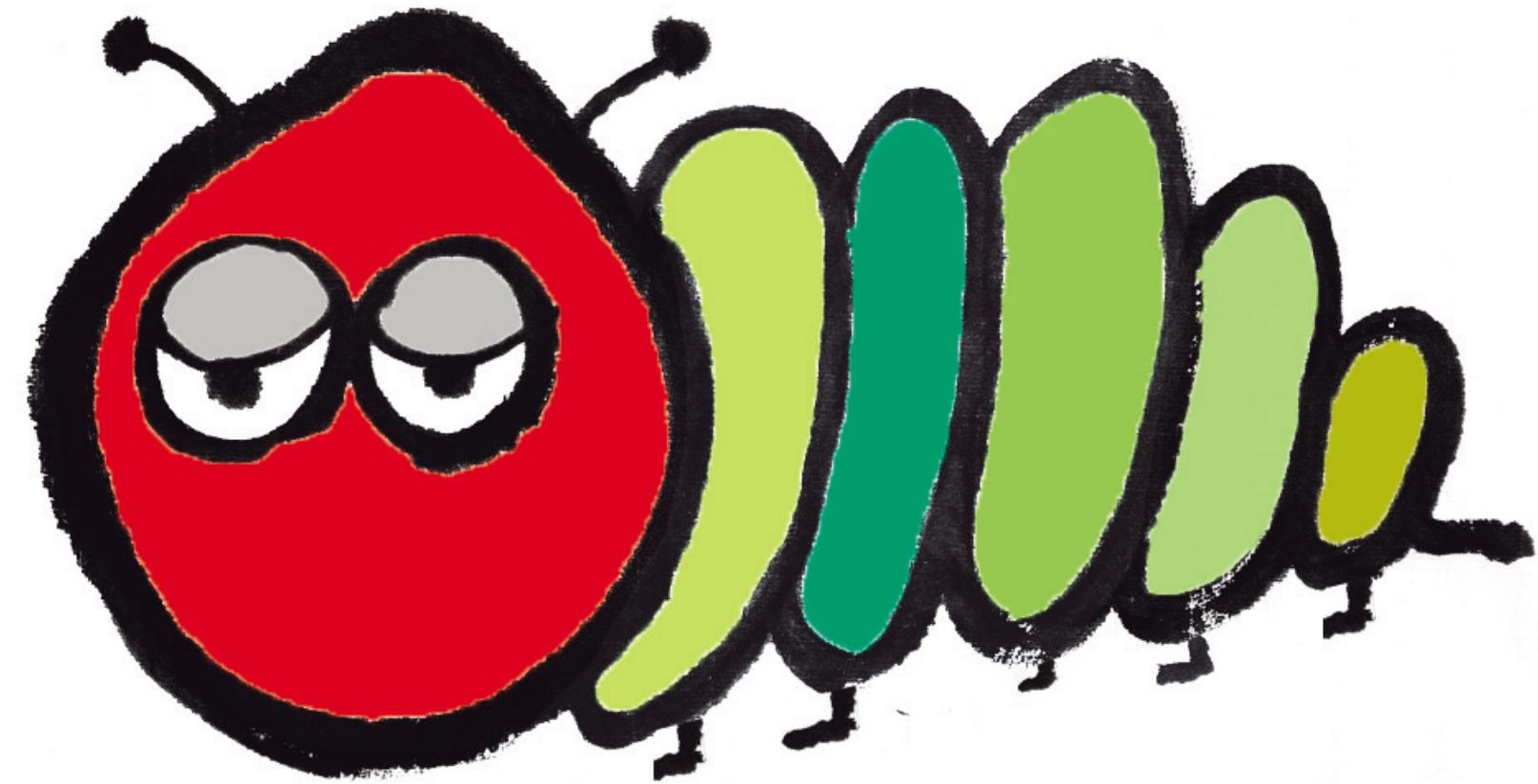
```
  ...
```

```
end
```



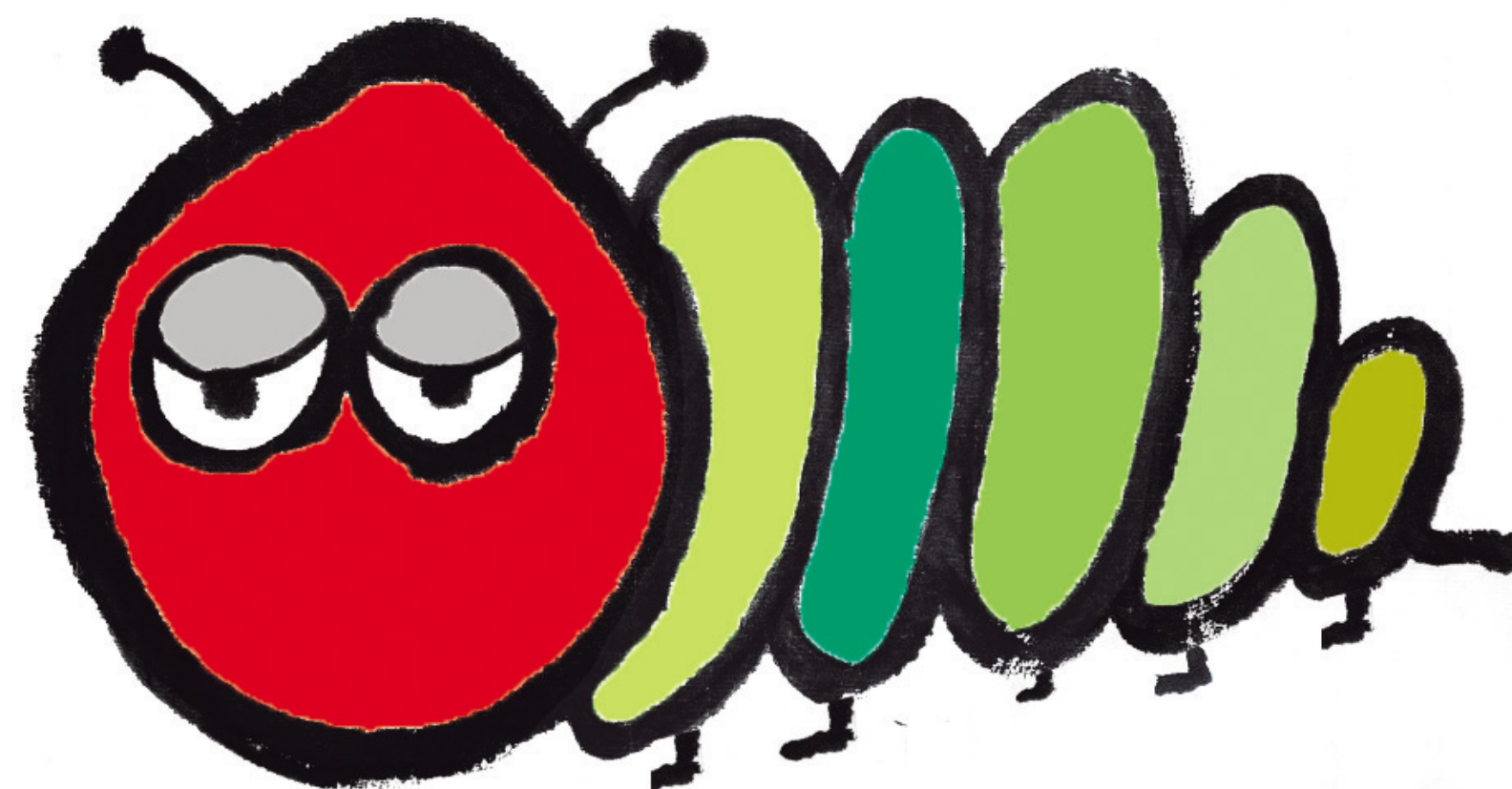
# Monday

A Transaction is Born



“

*I'm very hungry. I want  
groceries delivered to  
my chrysalis.*



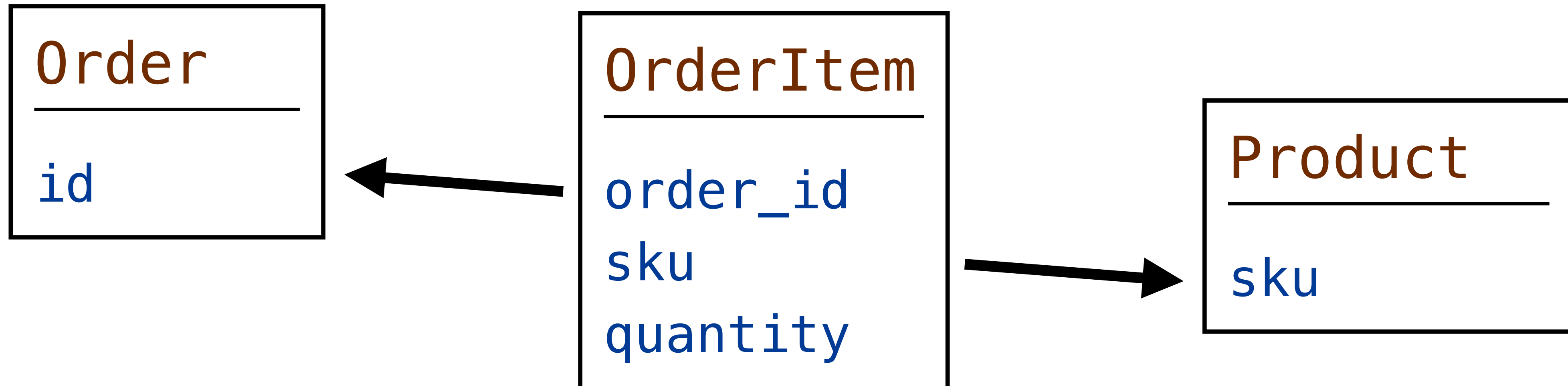
# Monday

---

```
def submit
  @order.save!
end
```

# Monday

---





# Monday

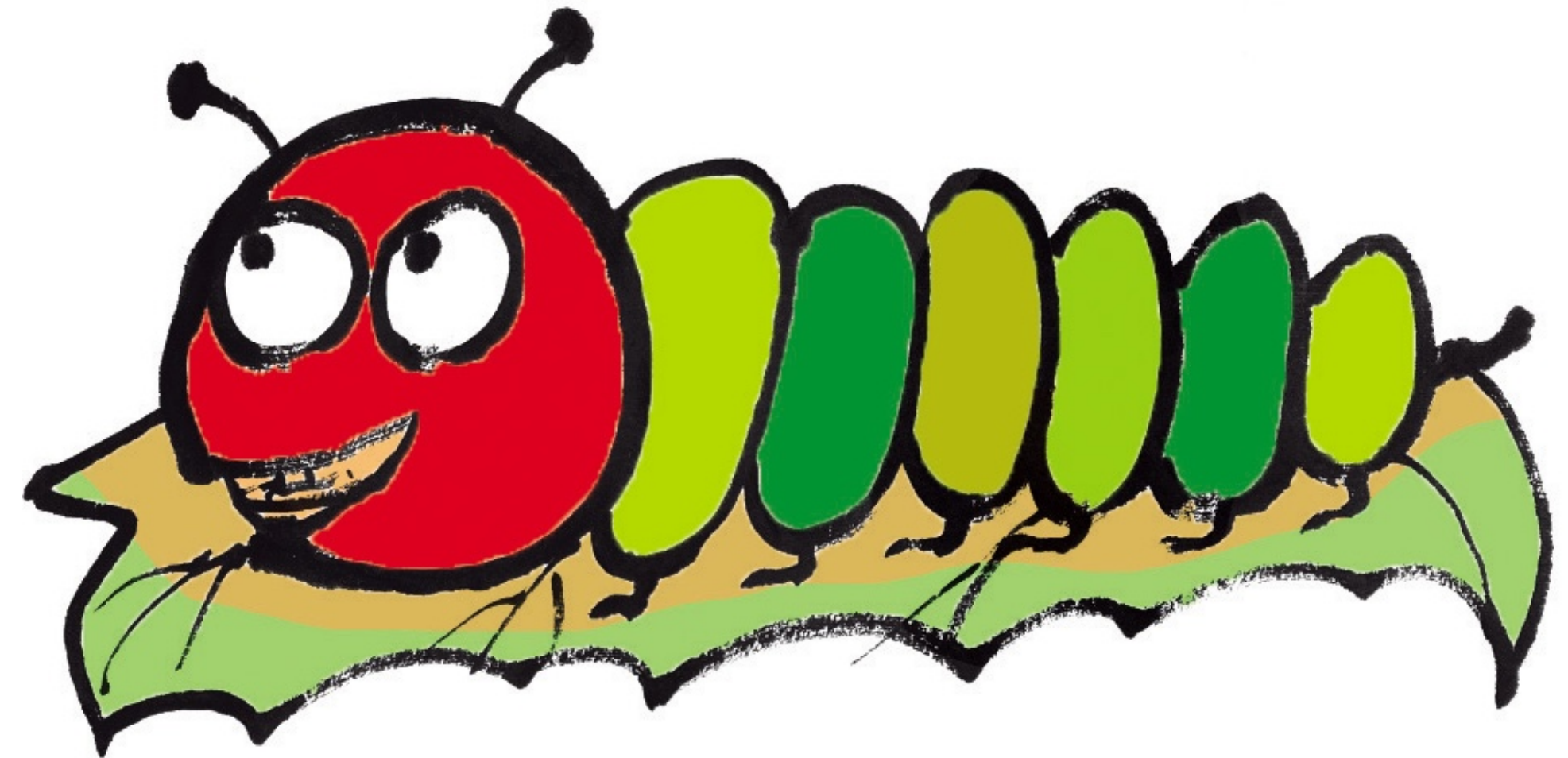
---

```
def submit
  @order.save!
end
```

```
=> BEGIN
=> INSERT INTO orders ...
=> INSERT INTO order_items ...
=> INSERT INTO order_items ...
...
=> COMMIT / ROLLBACK
```

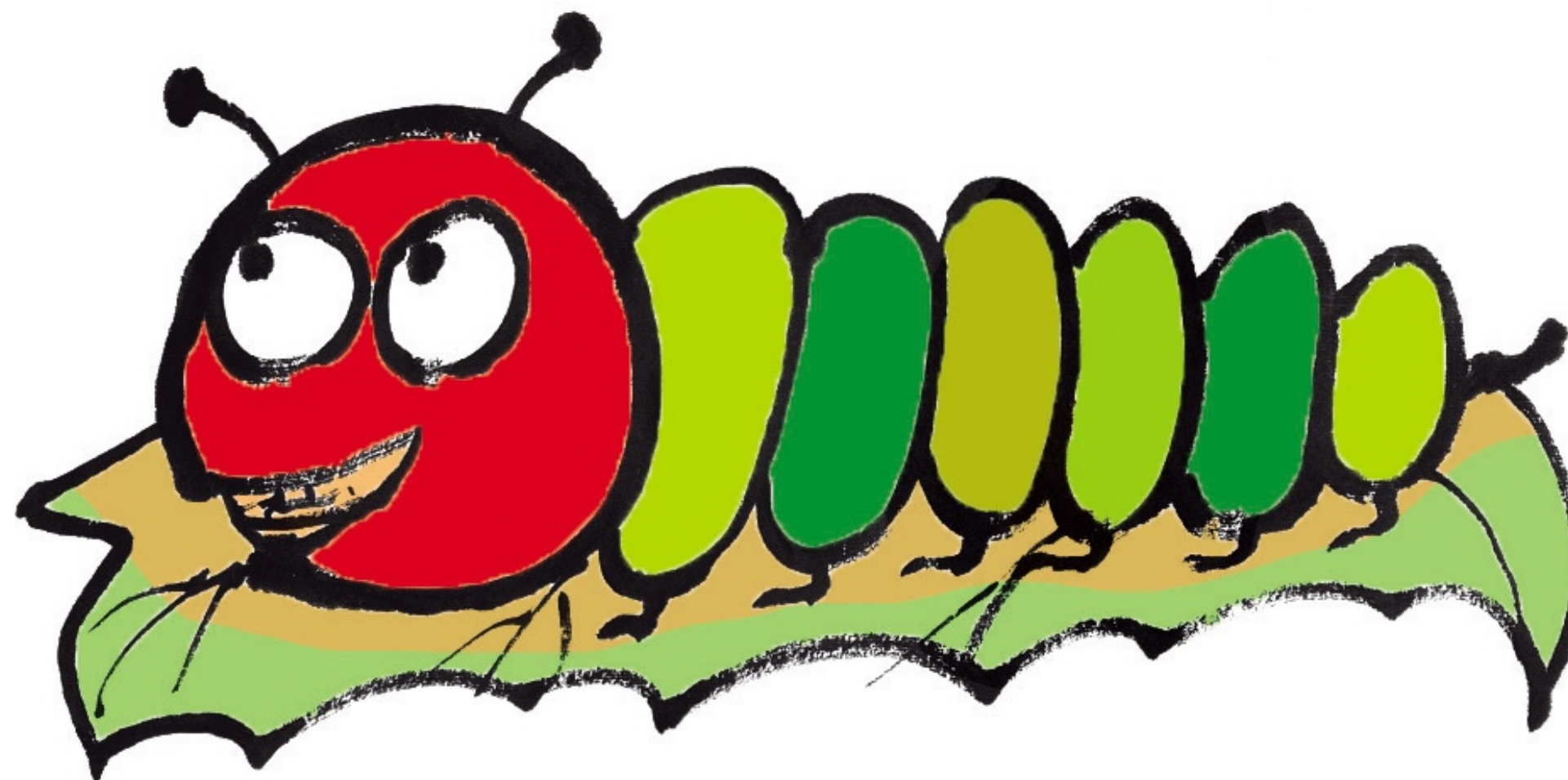
# Tuesday

Slow Transactions



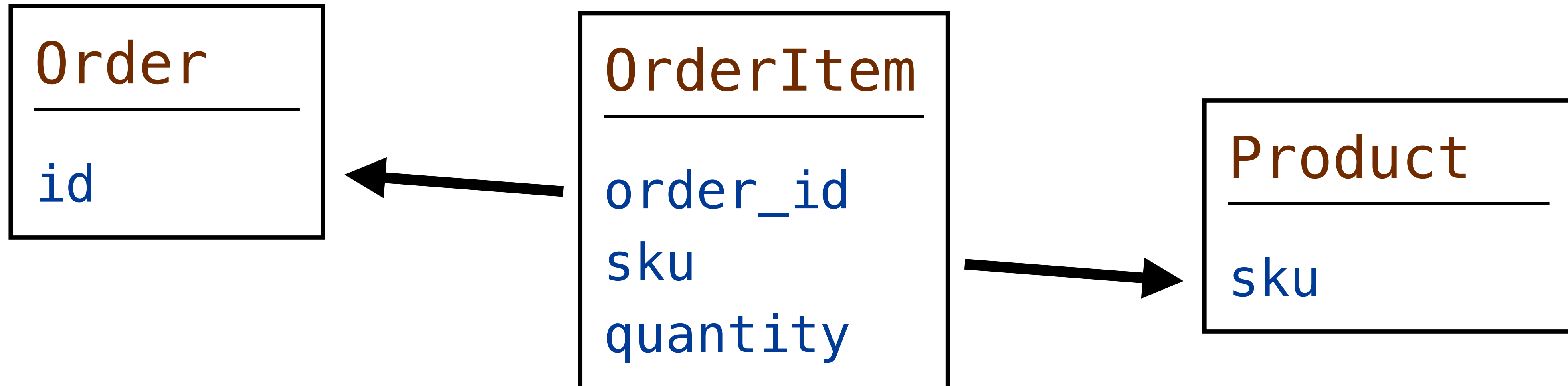
“

*I was so excited about my order, but it was canceled and now the items are not available.*



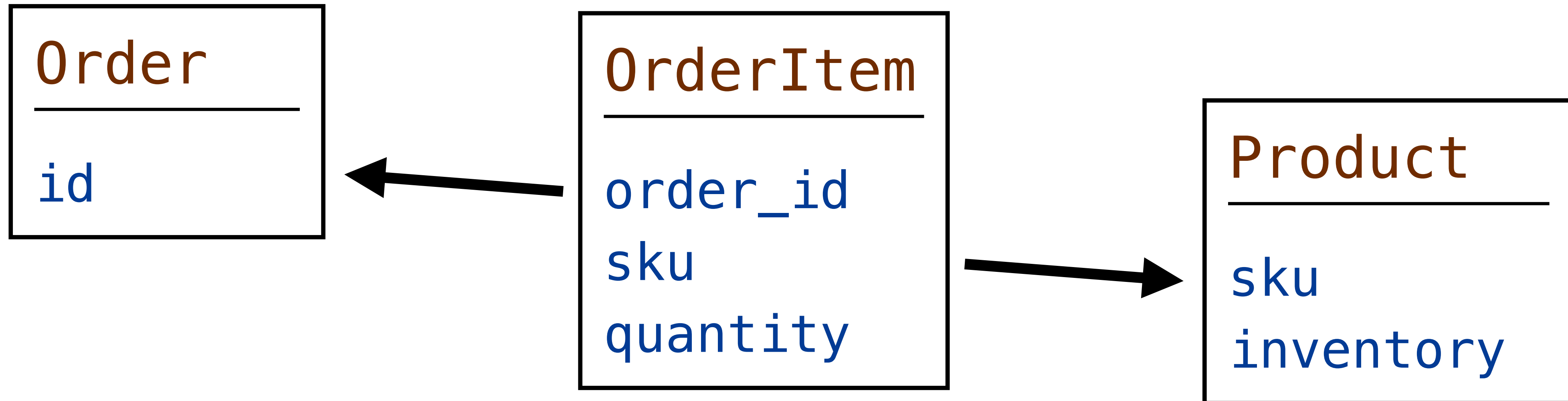
# Tuesday

---



# Tuesday

---





# Tuesday

---

```
def submit
  @order.save!
end
```

# Tuesday

---

```
def submit
  @order.save!
  @order.update_product_inventory
end
```

```
=> UPDATE products
     SET inventory = inventory - :order_item_quantity
     WHERE sku = :order_item_sku
```

# Tuesday

---

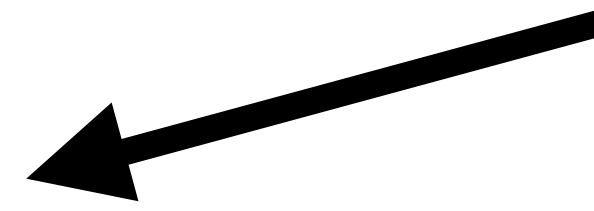
```
def submit
```

```
  @order.save!
```

```
  @order.update_product_inventory
```

```
end
```

Should be Atomic



# Tuesday

---

```
def submit
  Order.transaction do
    @order.save!
    @order.update_product_inventory
  end
end
```

# Tuesday

---

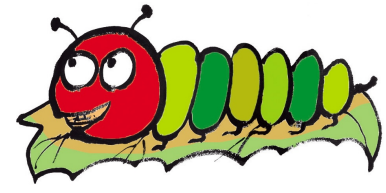
```
=> BEGIN
=> INSERT INTO orders ...
=> INSERT INTO order_items ...
=> INSERT INTO order_items ...
...
=> UPDATE products ...
=> UPDATE products ...
...
=> COMMIT
```



# Bug Report

*Increased 500 responses for placing orders*

# Tuesday



## Lock Contention



# Tuesday

---

## Lock Contention

### Bug 1

=> BEGIN

=> UPDATE . . . sku = APL1

 Row locked . . .

 Row locked . . .

 Row locked . . .

=> COMMIT

### Bug 2

=> BEGIN

=> UPDATE . . . sku = APL1

Waiting . . .

Waiting . . .

Waiting . . .

 Row locked . . .

# Tuesday

---

## Little Opportunity for Contention

```
=> BEGIN  
=> UPDATE products ... WHERE sku = APL1  
=> UPDATE products ... WHERE sku = ORG5  
=> COMMIT
```

|  
| < 1 millisecond  
|

# Tuesday

---

## Large Opportunity for Contention

=> BEGIN

=> UPDATE products ... WHERE sku = APL1

=> UPDATE products ... WHERE sku = ORG5

=> COMMIT



> 1 second



# Tuesday

---

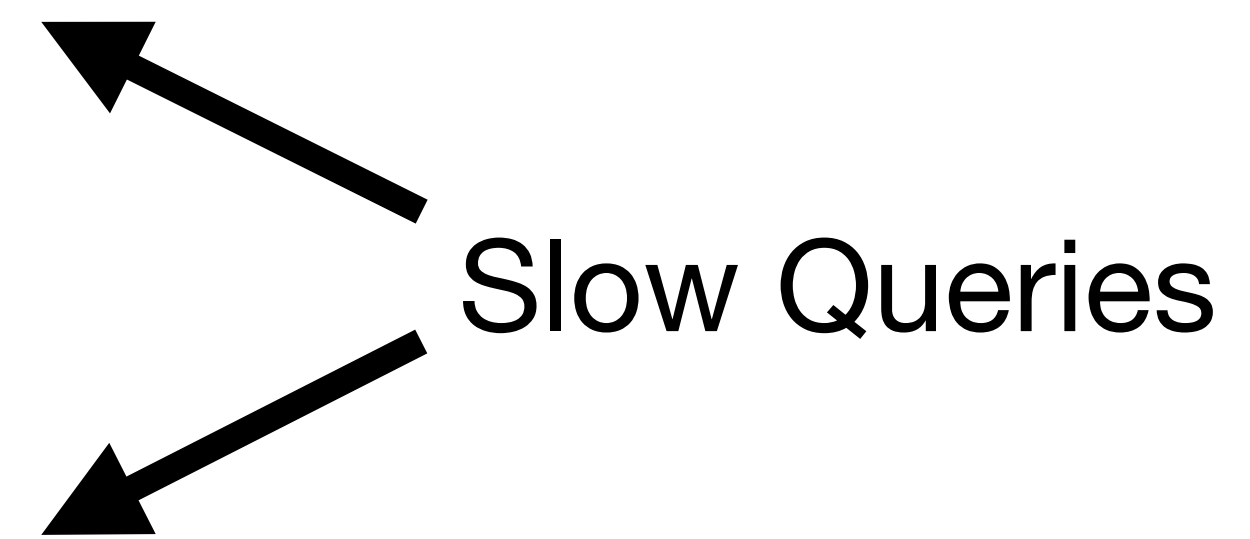
## Large Opportunity for Contention

=> BEGIN

=> UPDATE products ... WHERE sku = APL1

=> UPDATE products ... WHERE sku = ORG5

=> COMMIT



# Tuesday

---

## Large Opportunity for Contention

=> BEGIN

=> UPDATE products ... WHERE sku = APL1

=> UPDATE products ... WHERE sku = PLM3

=> UPDATE products ... WHERE sku = BRY4

=> UPDATE products ... WHERE sku = PER2

... (hundreds more)

=> UPDATE products ... WHERE sku = ORG5

=> COMMIT



Too Many Queries

# Tuesday

---

## Large Opportunity for Contention

=> BEGIN

=> UPDATE products ... WHERE sku = APL1

=> UPDATE products ... WHERE sku = PLM3

=> UPDATE products ... WHERE sku = BRY4

=> UPDATE products ... WHERE sku = PER2

... (hundreds more)

=> UPDATE products ... WHERE sku = ORG5

=> COMMIT



# Tuesday

---

```
def submit
  Order.transaction do
    @order.save!
    @order.update_product_inventory
  end
end
```

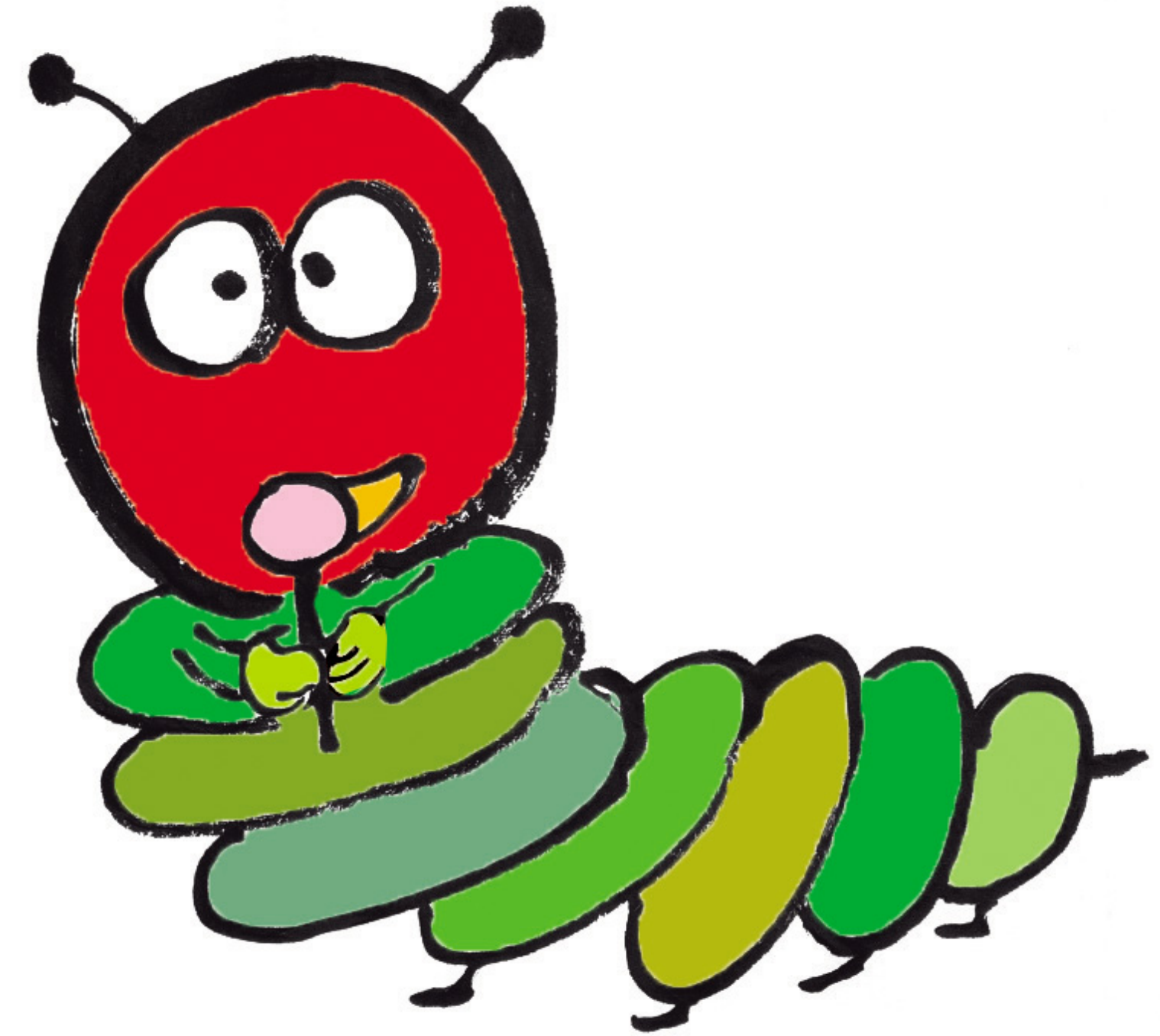
# Tuesday

---

```
def submit
  Order.transaction do
    @order.save!
    @order.update_product_inventory
  end
end
```

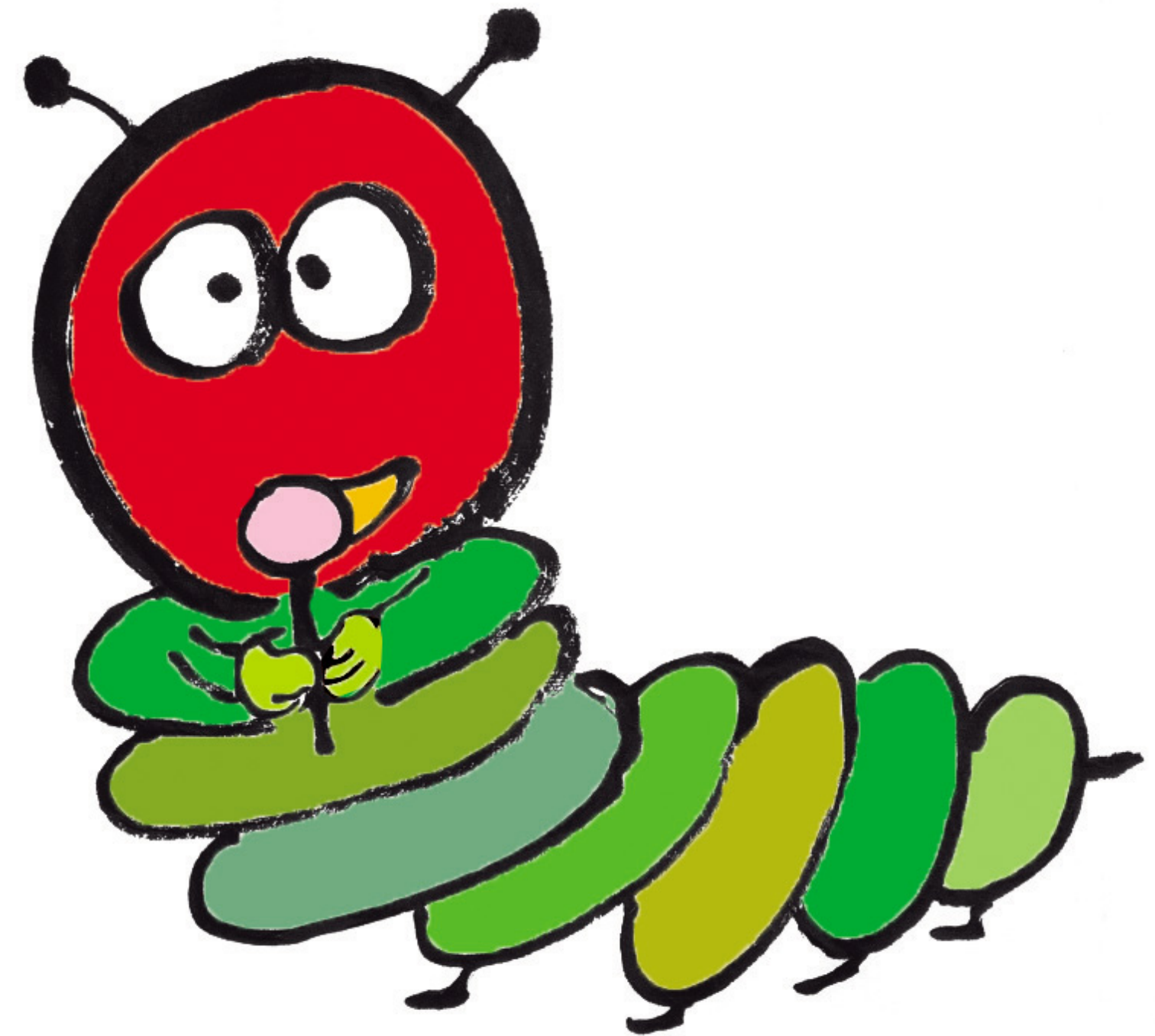
# Wednesday

Background Jobs



“

*I don't have time to waste. I want my orders to submit without delay.*





# Wednesday

---

```
class Order < ApplicationRecord
  after_create :finalize_order
end
```

# Wednesday

---

```
class Order < ApplicationRecord
  after_create :finalize_order

  def finalize_order
    # Sync with billing platform
    # Send confirmation email
    # etc.
  end
end
```

# Wednesday

---

```
class Order < ApplicationRecord
  after_create :finalize_order

  def finalize_order
    OrderFinalizationJob.perform_later
  end
end
```

# Bug Report

*Confirmation emails delayed by minutes*

# Wednesday

---

=> **BEGIN**

=> **INSERT INTO** orders ...

=> **UPDATE** products ...

=> **COMMIT**

=> Enqueue Job

=> Job Executes and Fails

=> Job Retries and Fails

=> Job Retries and Succeeds

# Wednesday

---

=> **BEGIN**

=> **INSERT INTO** orders . . .

=> **UPDATE** products . . .

=> **ROLLBACK**

=> Enqueue Job

=> Job Executes and Fails

=> Job Retries and Fails

=> Job Retries and Fails

=> Job Retries and Fails

# Wednesday

---

```
class Order < ApplicationRecord
  after_create :finalize_order

  def finalize_order
    OrderFinalizationJob.perform_later
  end
end
```



# Wednesday

---

```
class Order < ApplicationRecord
  after_create_commit :finalize_order

  def finalize_order
    OrderFinalizationJob.perform_later
  end
end
```

# Wednesday

---

=> BEGIN

=> INSERT INTO orders . . .

=> UPDATE products . . .

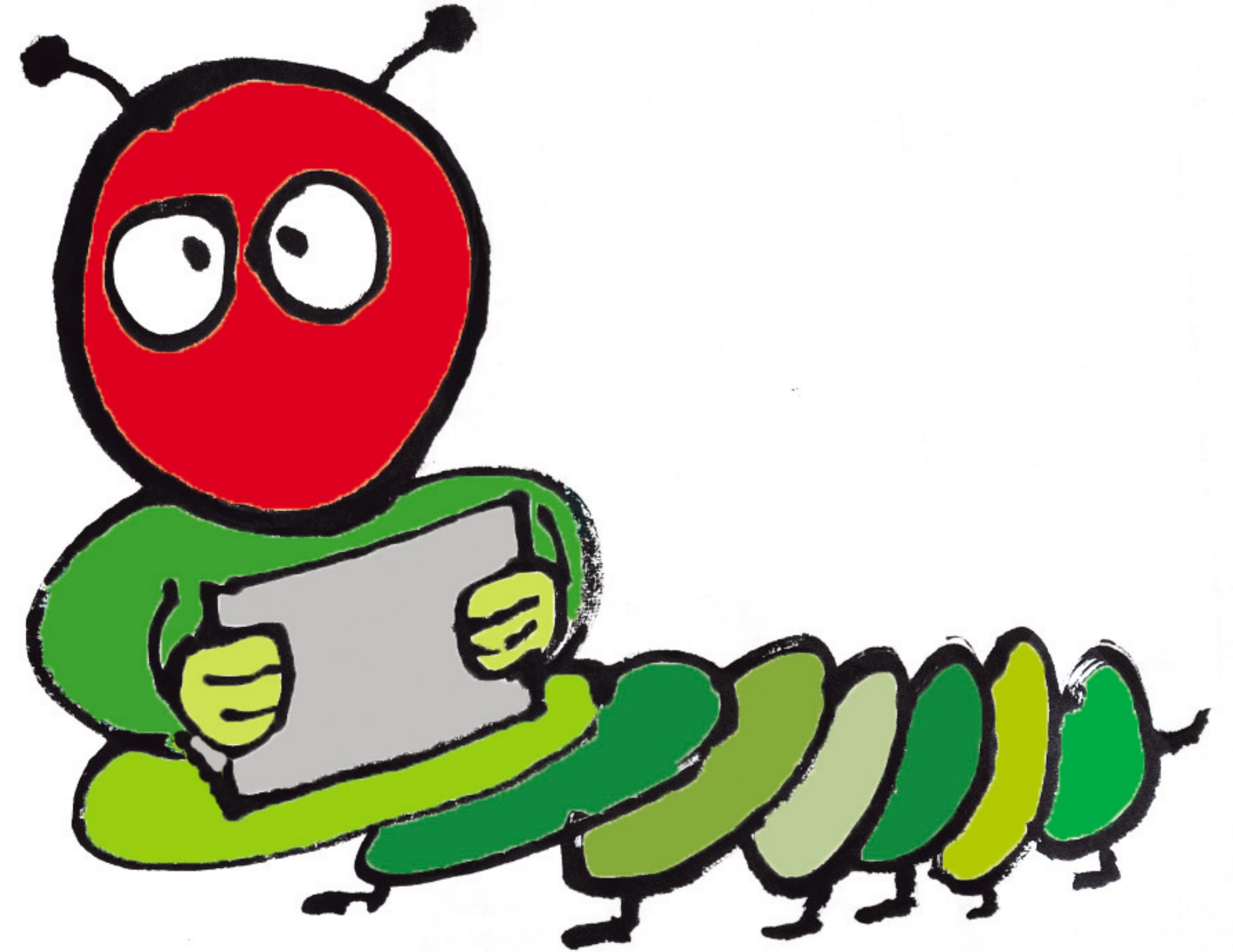
=> COMMIT

=> Enqueue Job

=> Job Executes and Succeeds

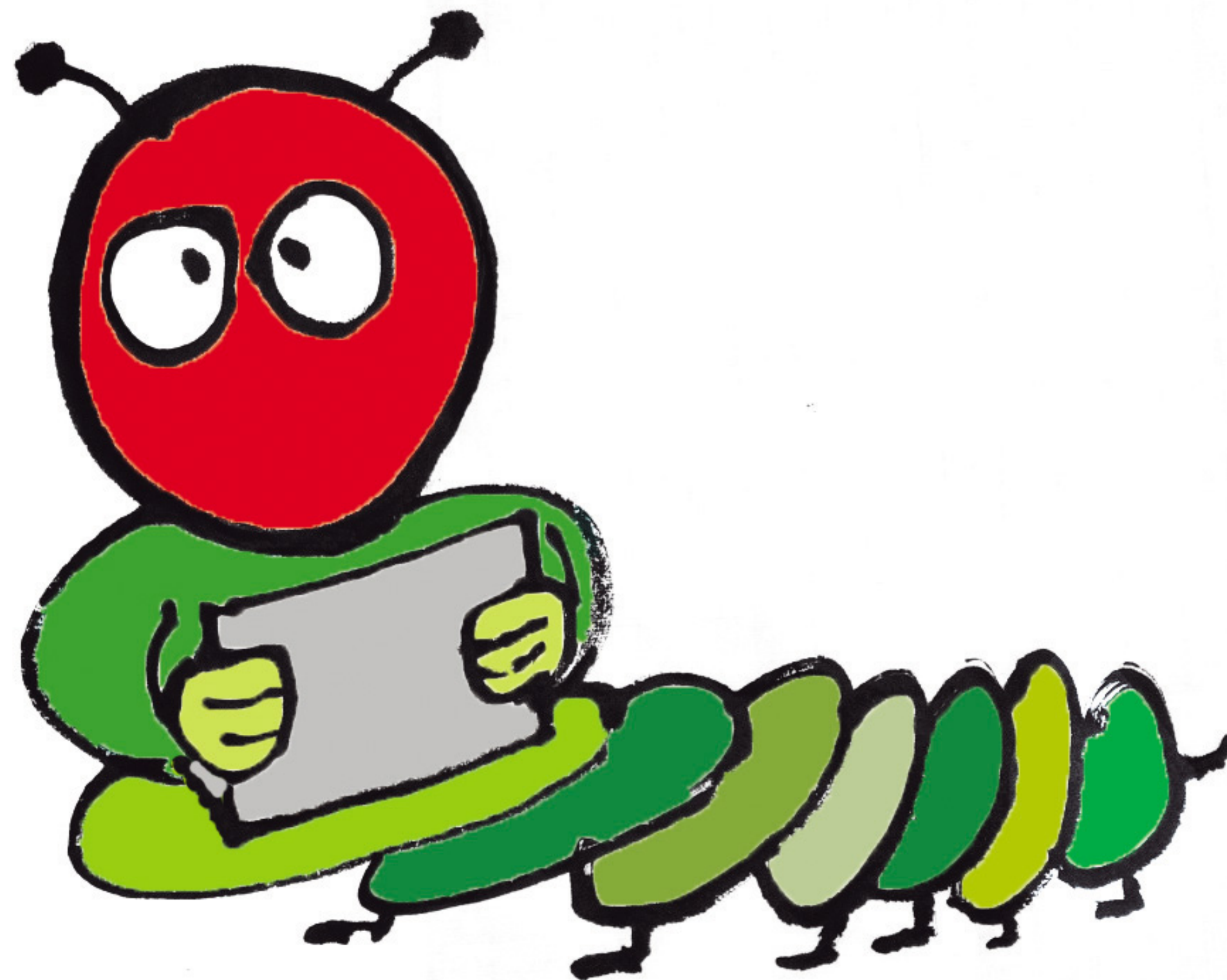
# Thursday

External Services



“

*I want my orders  
fulfilled quickly and  
accurately.*



# Thursday

---

```
def submit
  Order.transaction do
    @order.save!
    @order.update_product_inventory
  end
end
```

# Thursday

---

```
def submit
  Order.transaction do
    @order.save!
    FulfillmentClient.submit_order(@order)
    @order.update_product_inventory
  end
end
```

# Thursday

---

=> **BEGIN**

=> **INSERT INTO** orders . . .

=> External Call

=> **UPDATE** products . . .

=> **COMMIT**



# Bug Report

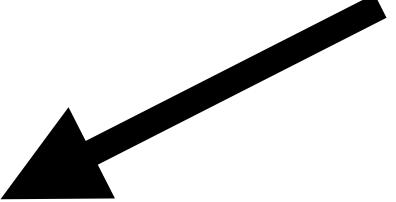
*Site-wide failure*

# Thursday

---

```
def submit
  Order.transaction do
    @order.save!
    FulfillmentClient.submit_order(@order)
    @order.update_product_inventory
  end
end
```

Responding Slowly



# Thursday

---

=> **BEGIN**

=> **INSERT INTO** orders ...



Slow External Call

=> **UPDATE** products ...

=> **COMMIT / ROLLBACK**

# Thursday

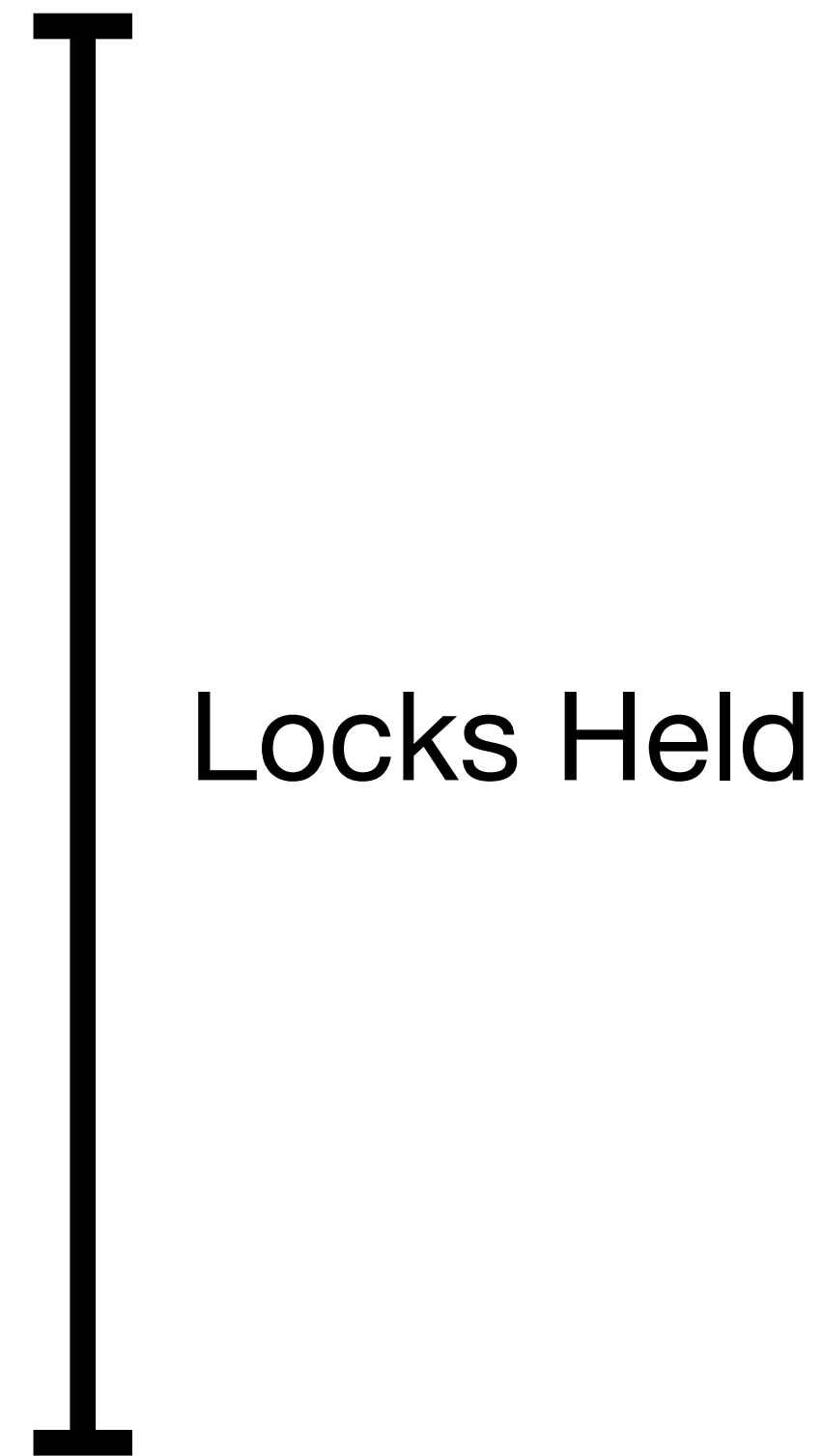
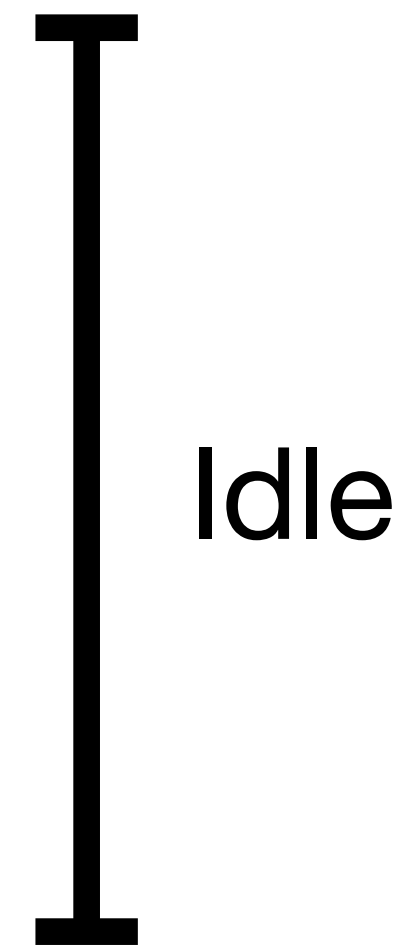
---

=> **BEGIN**

=> **INSERT INTO** orders ...

=> **UPDATE** products ...

=> **COMMIT / ROLLBACK**

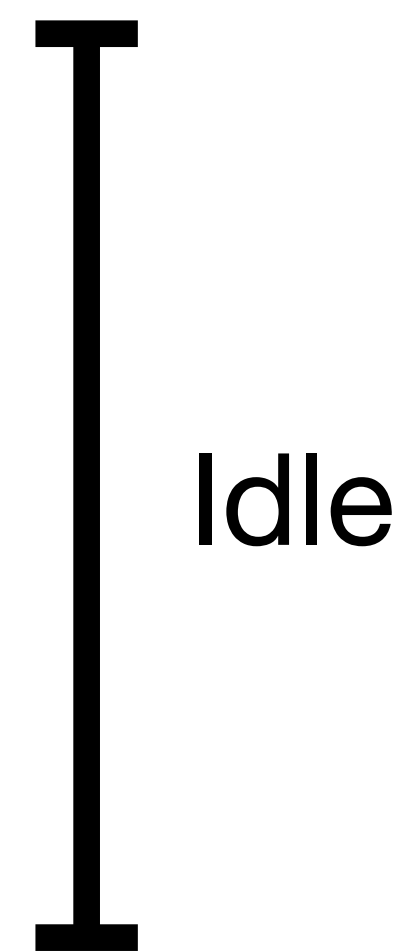


# Thursday

---

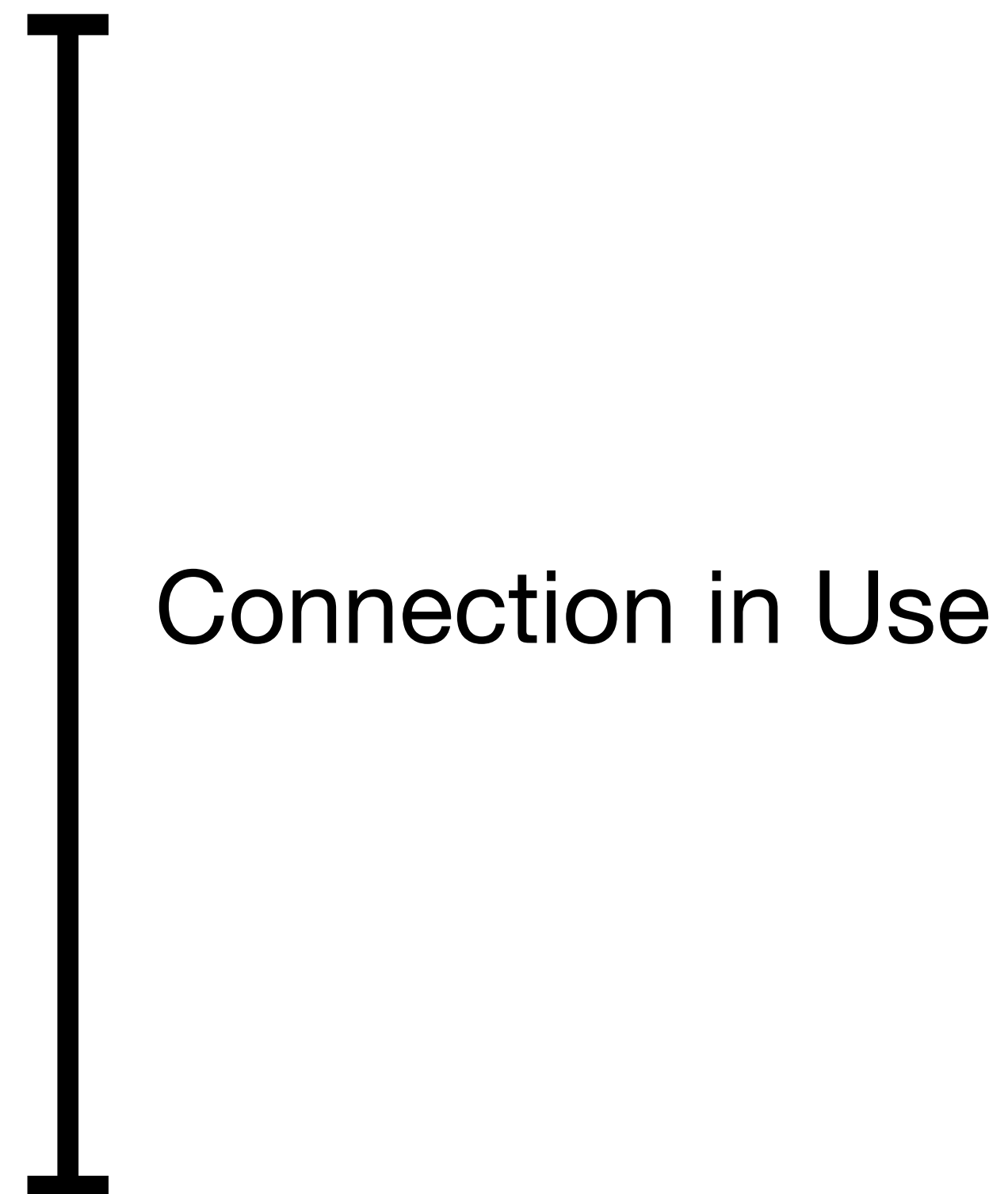
=> **BEGIN**

=> **INSERT INTO** orders ...

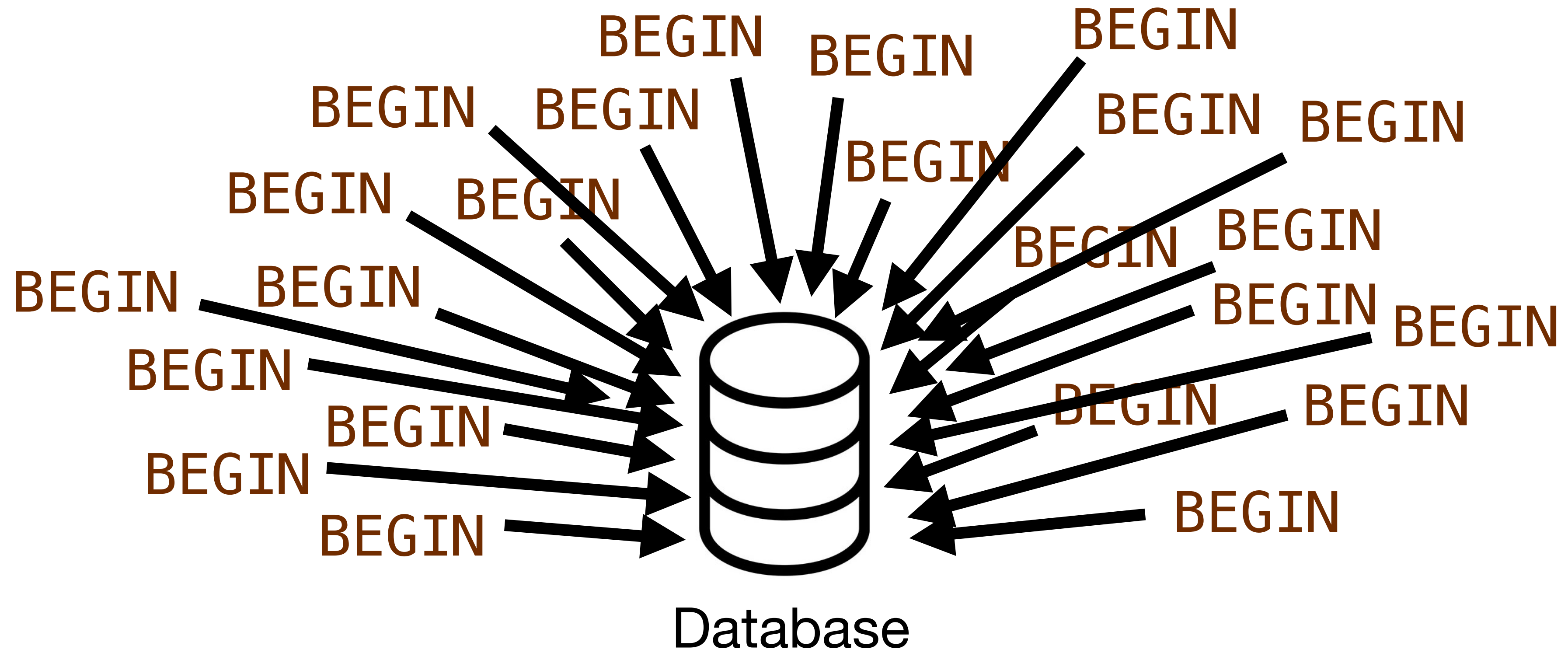
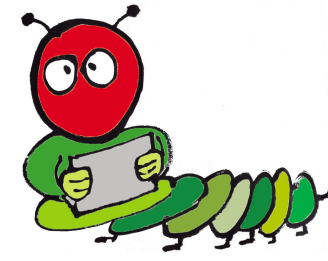


=> **UPDATE** products ...

=> **COMMIT / ROLLBACK**

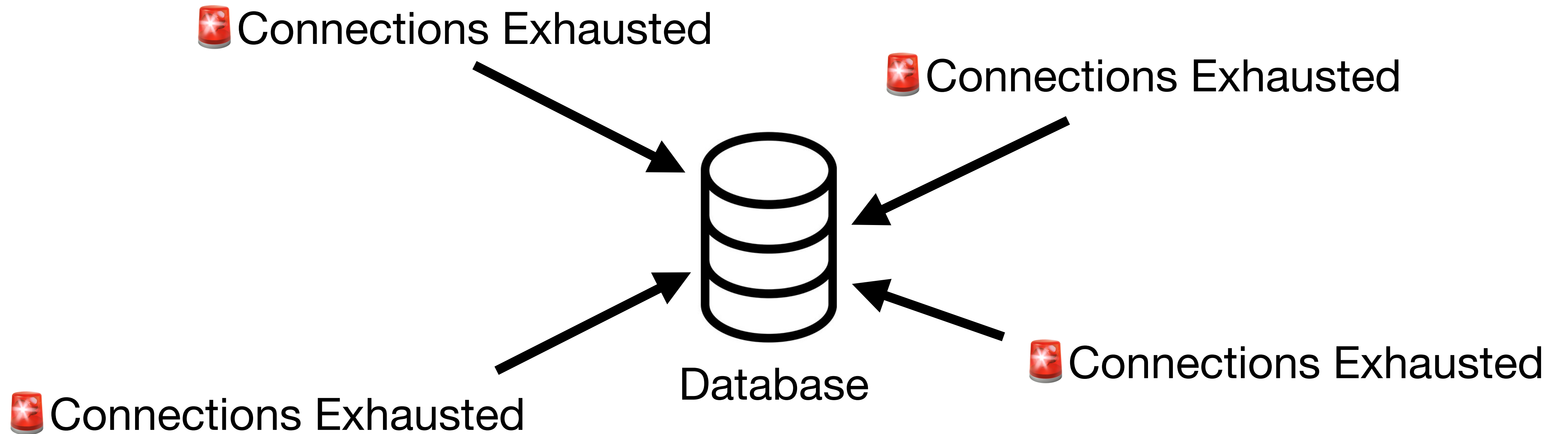


# Thursday



# Thursday

---



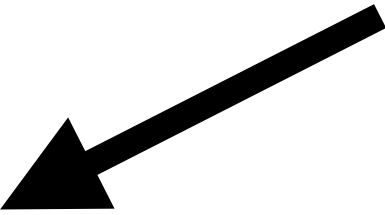


# Thursday

---

```
def submit
  Order.transaction do
    @order.save!
    FulfillmentClient.submit_order(@order)
    @order.update_product_inventory
  end
end
```

Configure Timeout




# Thursday

---

```
def submit
  FulfillmentClient.submit_order(@order)

  Order.transaction do
    @order.save!
    FulfillmentClient.submit_order(@order)
    @order.update_product_inventory
  end
end
```

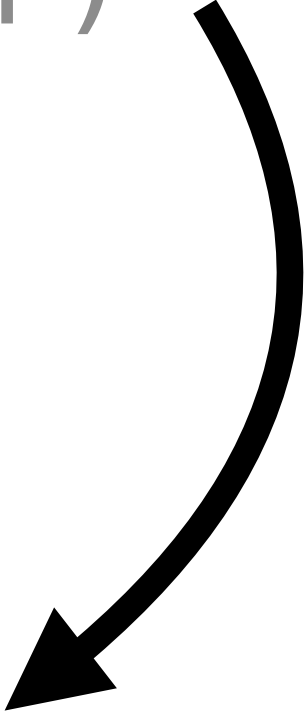


# Thursday

---

```
def submit
  Order.transaction do
    @order.save!
    FulfillmentClient.submit_order(@order)
    @order.update_product_inventory
  end
```

```
  FulfillmentClient.submit_order(@order)
end
```



# Thursday

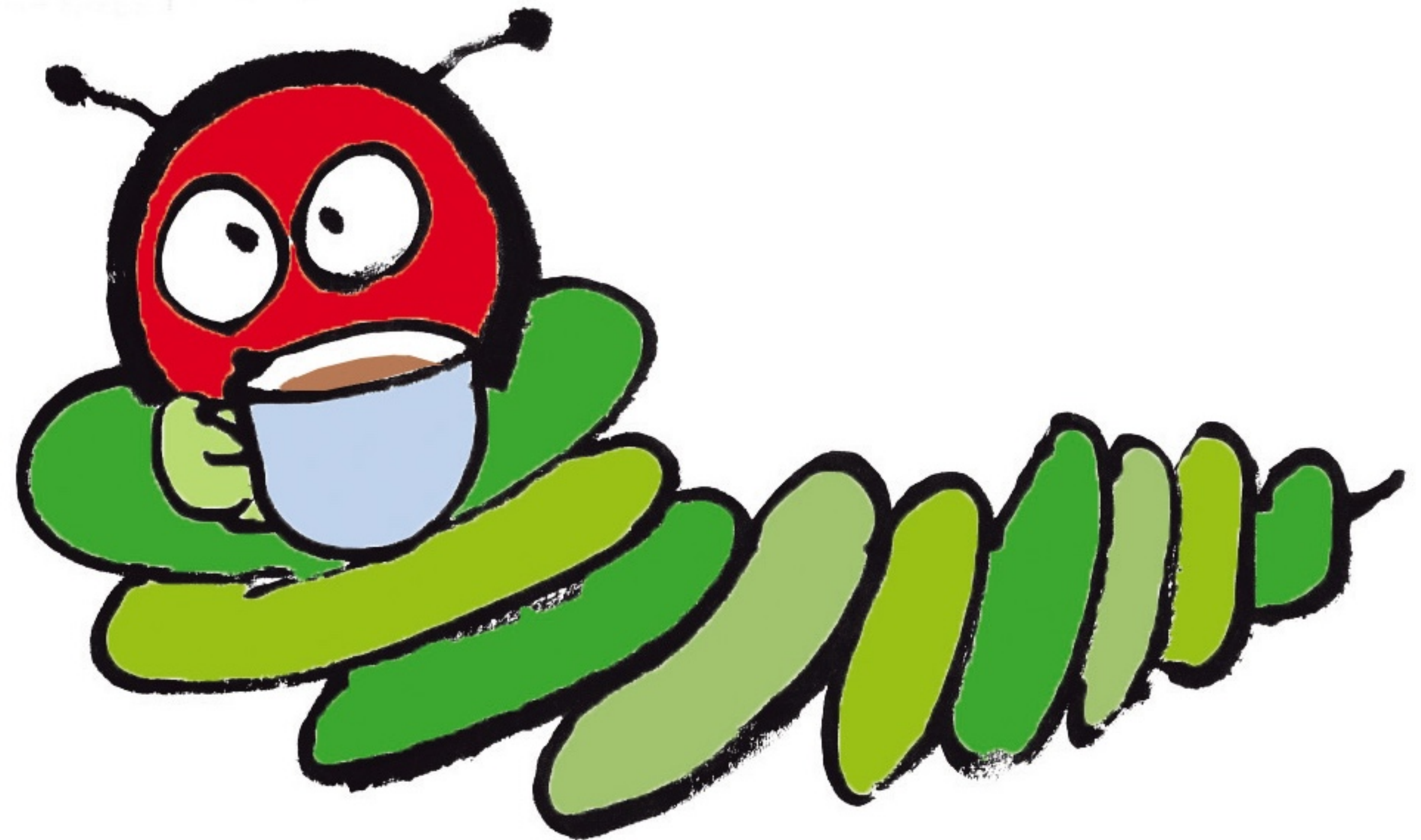
---

```
def submit
  Order.transaction do
    @order.save!
    @order.update_product_inventory
  end

  FulfillmentClient.submit_order(@order)
  @order.update!(status: :submit)
end
```

# Friday

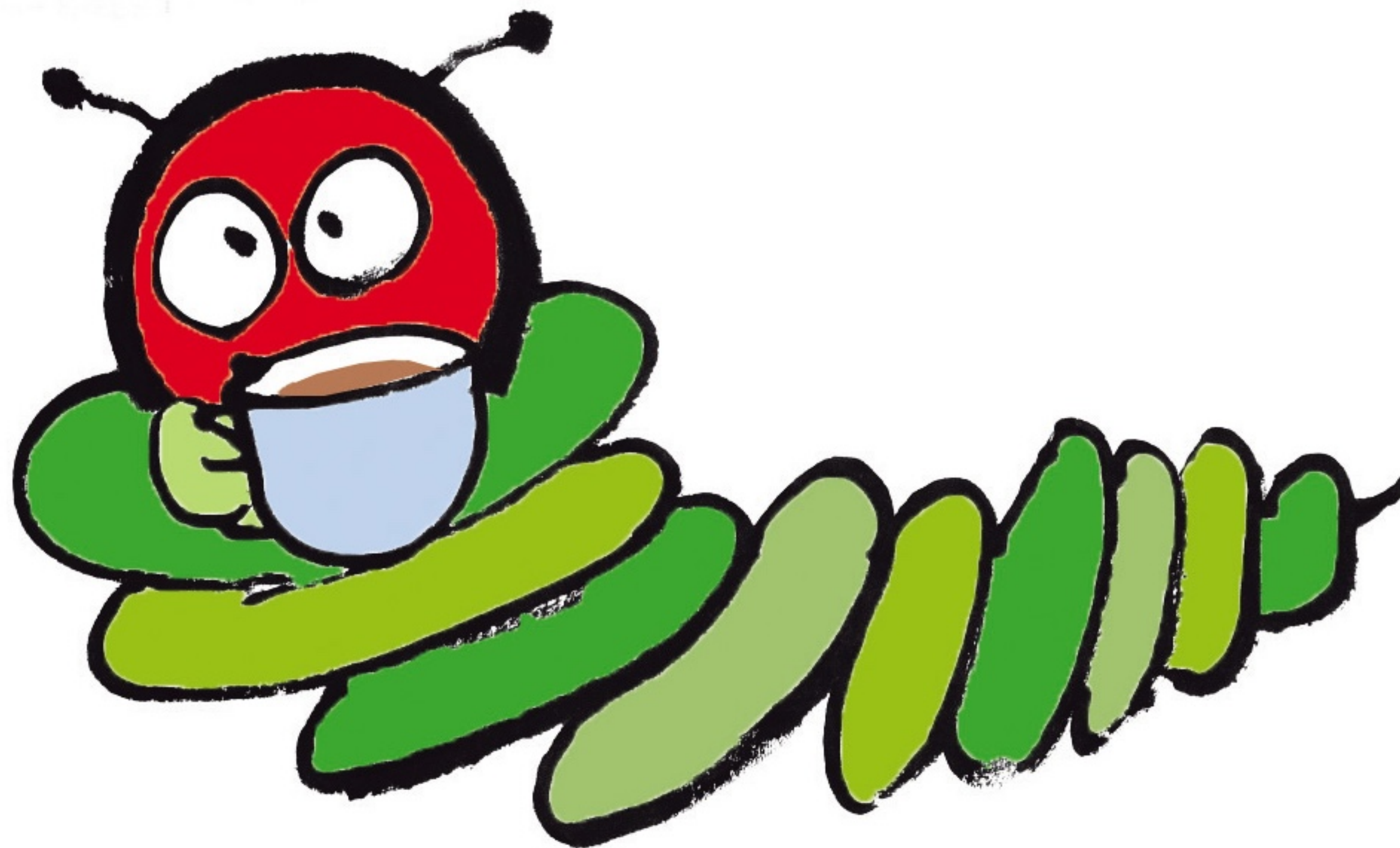
Multiple Databases



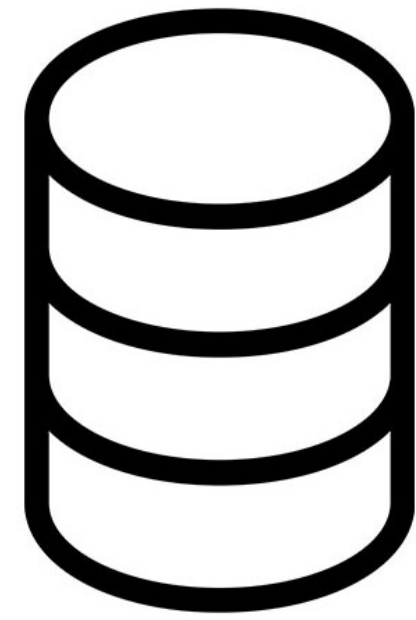
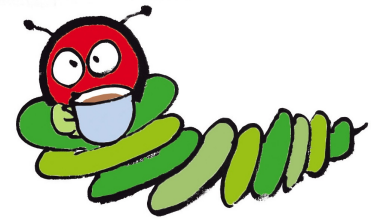


“

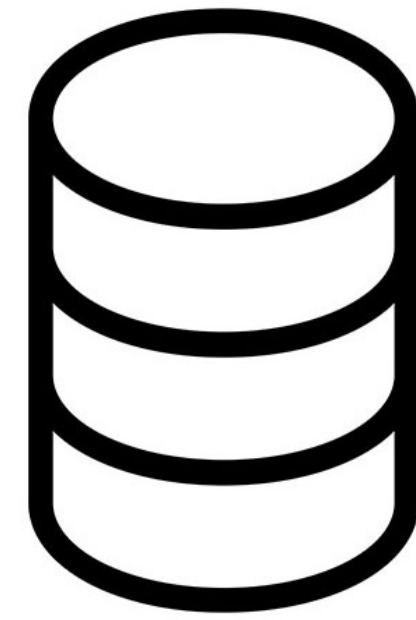
*I expect a reliable website. BugHub has been having a lot of problems lately.*



# Friday



Orders



Products

# Friday

---

```
def submit
  Order.transaction do
    @order.save!
    FulfillmentClient.submit_order(@order)
    @order.update_product_inventory
  end
end
```



# Friday

---

```
def submit
  Order.transaction do
    Product.transaction do
      @order.save!
      FulfillmentClient.submit_order(@order)
      @order.update_product_inventory
    end
  end
end
```

# Bug Report

*All of the above, but worse*

# Friday

---

```
def submit
```

```
  Order.transaction do
```

```
    Product.transaction do
```

```
      ...
```

```
    end
```

```
  end
```

```
end
```

← Problems Affect Both Databases

# Bug Report

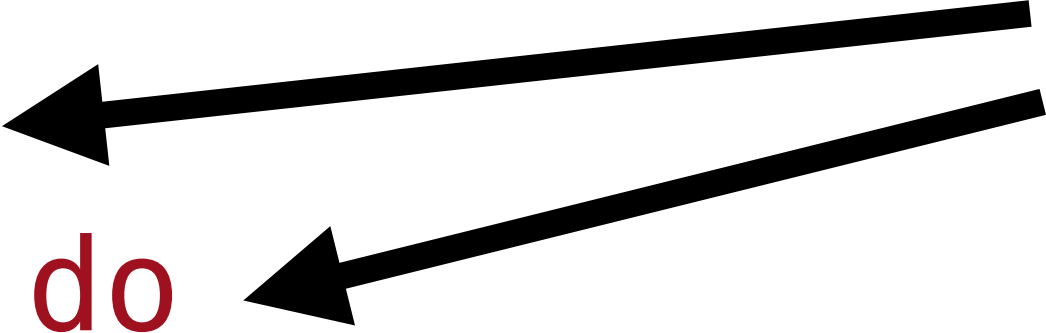
*Products incorrectly marked out-of-stock*

# Friday

---

```
def submit
  Order.transaction do
    Product.transaction do
      ...
    end
  end
end
```

Not Atomic



# Friday

---

## Orders Database

=> BEGIN

=> INSERT INTO orders ...

=> COMMIT

## Products Database

=> BEGIN

=> UPDATE products ...

=> COMMIT



# Friday

---

## Orders Database

=> BEGIN

=> INSERT INTO orders ...

=> ROLLBACK

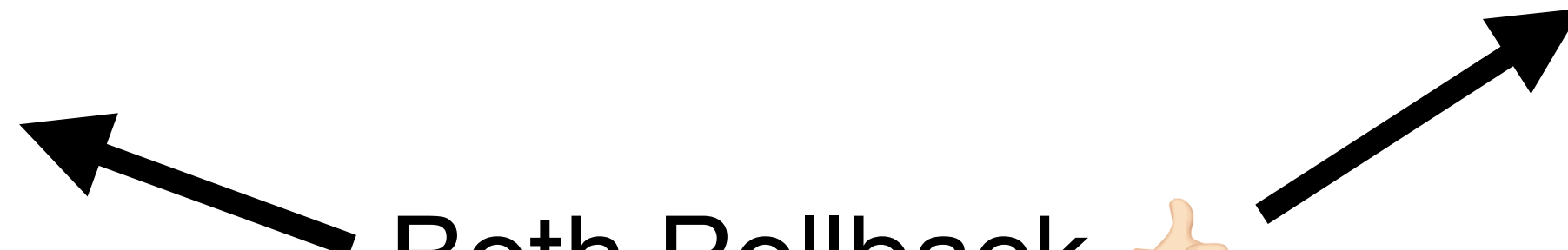
## Products Database

=> BEGIN

=> UPDATE products ...

=> ROLLBACK

Both Rollback 👍



# Friday

---

## Orders Database

=> BEGIN

=> INSERT INTO orders ...

=> ROLLBACK

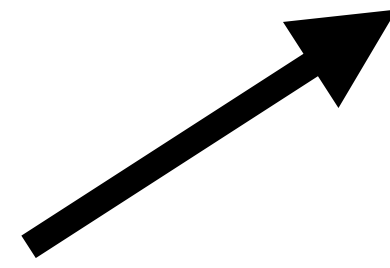
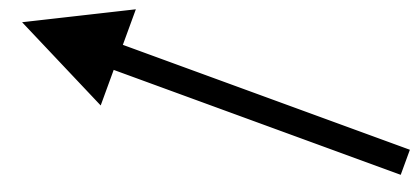
## Products Database

=> BEGIN

=> UPDATE products ...

=> COMMIT

Not Atomic 🙄





# Friday

---

## Orders Database

=> BEGIN

=> INSERT INTO orders ...

Idle

**X** Connection Failed

## Products Database

=> BEGIN

=> UPDATE products ...

=> COMMIT

# Friday

---

```
def submit
  Order.transaction do
    Product.transaction do
      ...
    end
  end
end
```

# Friday

---

```
def submit
  Product.transaction do
    ...
  end

  Order.transaction do
    ...
  end
end
```

# Friday

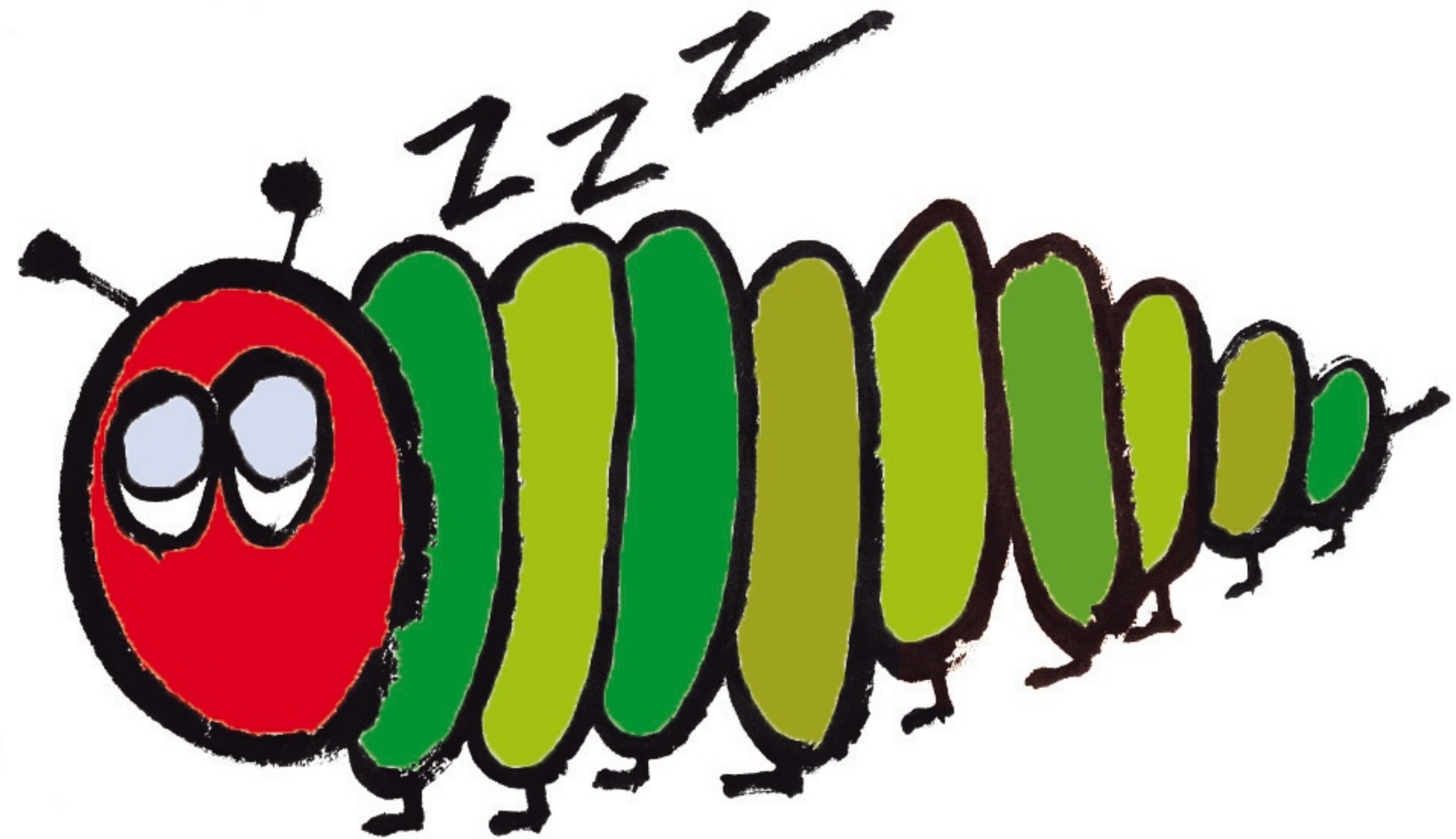
---

```
def submit
  Order.transaction do
    ...
  end

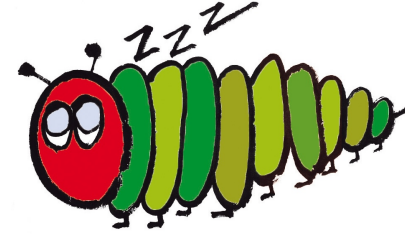
  Product.transaction do
    ...
  end
end
```

# Weekend

Rest and Reflect



# Weekend

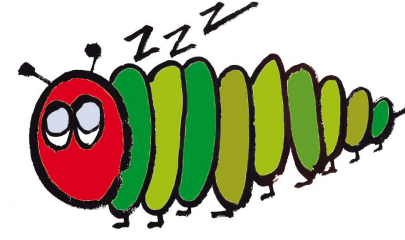


---

## **External Calls within Transactions are a Risk**

- Data Integrity Problems
- Cascading Failures
- Includes: HTTP requests, emails, jobs, queries to other databases, etc.

# Weekend



---

## **Slow Transactions are a Risk**

- Slow Requests
- Contention
- Resource Exhaustion



transaction they say you  
shouldn't worry about



enqueues jobs  
delivers emails  
makes you coffee  
sends you a postcard on  
christmas

your average transaction



```
INSERT INTO users  
  (id, username)  
VALUES (1, 'john');
```



**I SHOULD OPEN  
A TRANSACTION**

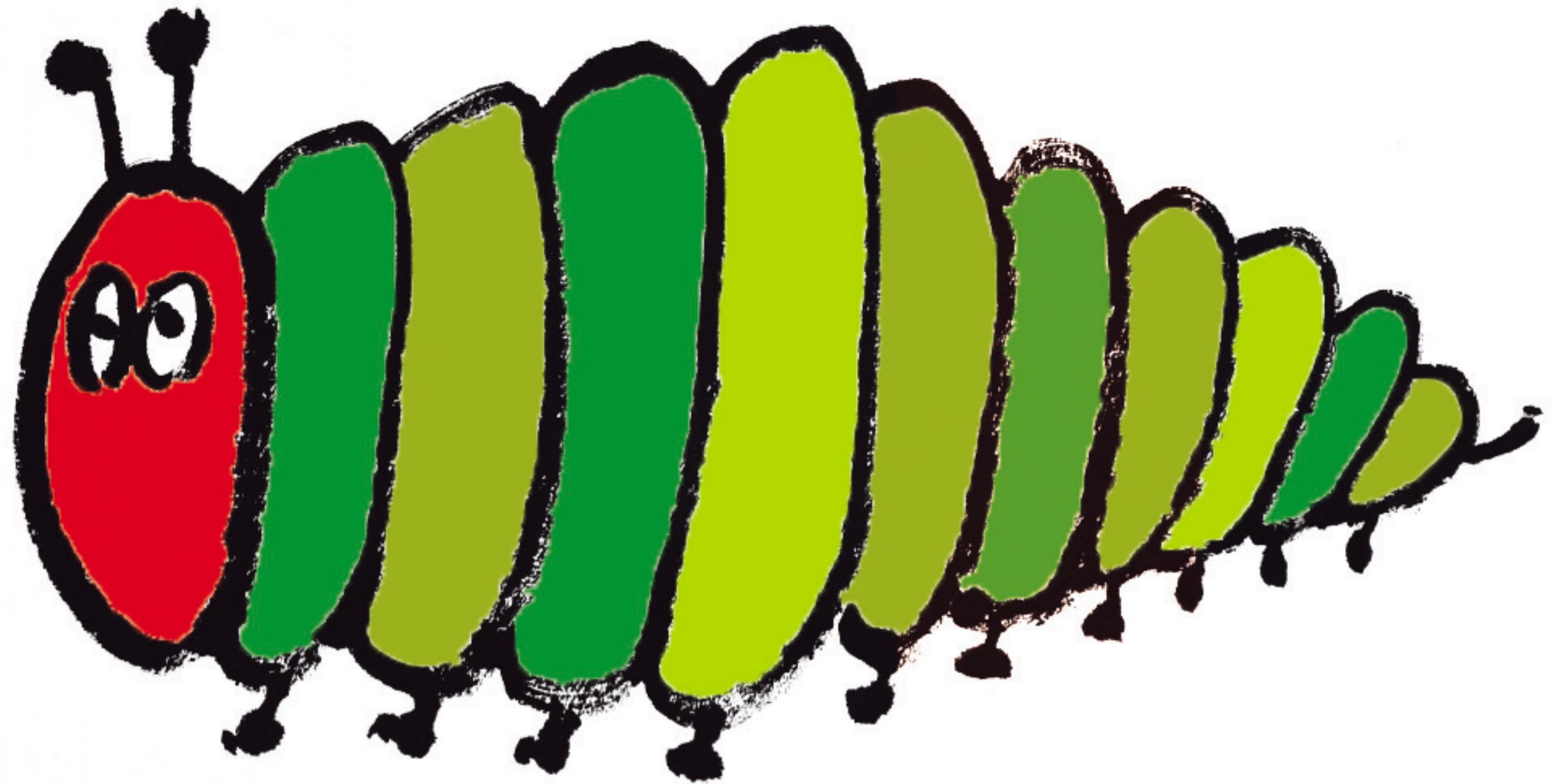


**THEY DON'T KNOW I'M ADDING  
MORE LATENCY TO THE TRANSACTION**



# Monday

Metamorphosis





# Monday

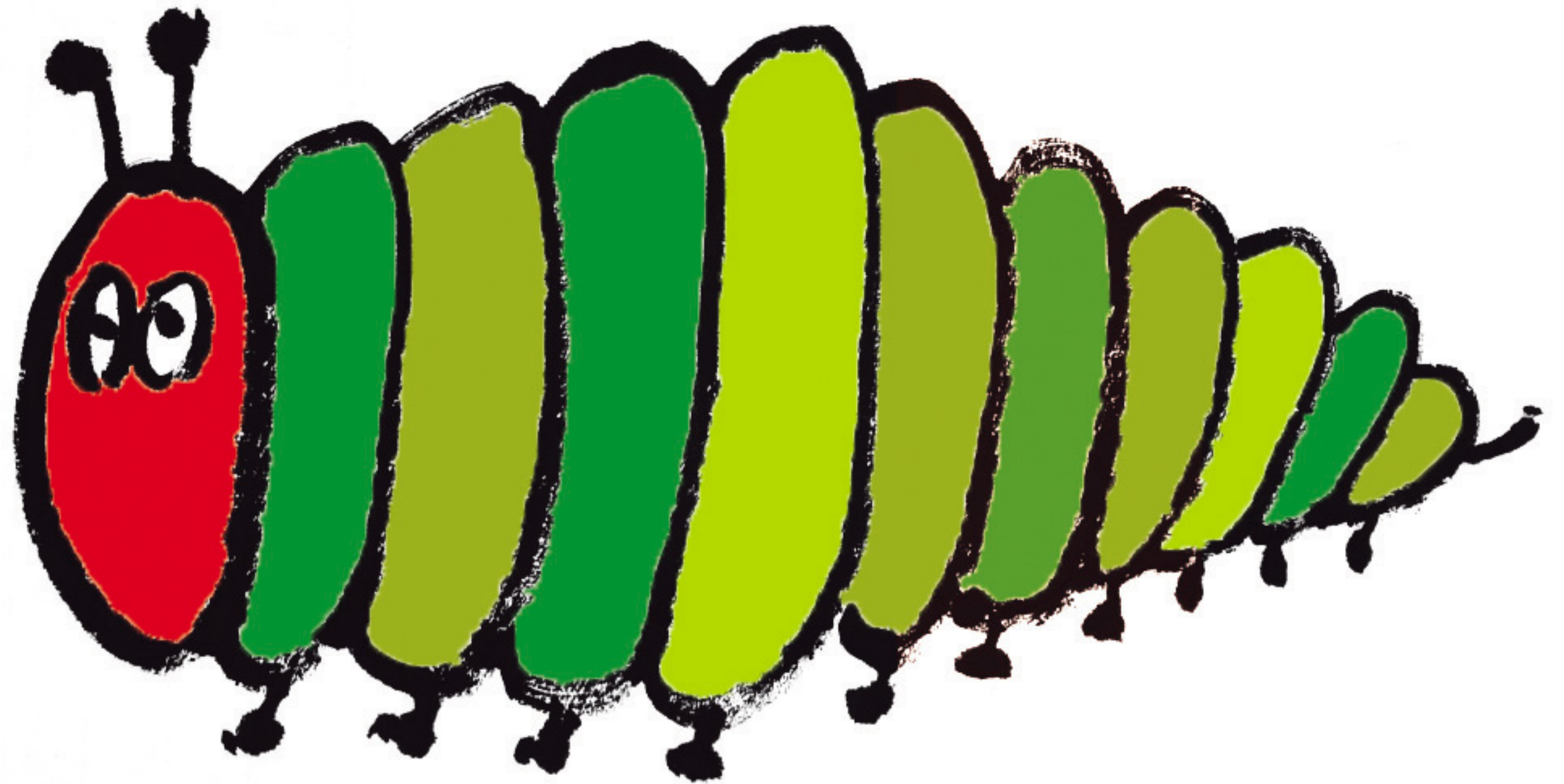
---

```
def submit
  Order.transaction do
    Product.transaction do
      Chrysalis.transaction do
        code.not_meant_for_reading!
        @order.save!
        FulfillmentClient.submit_order(@order)
        @order.update_product_inventory

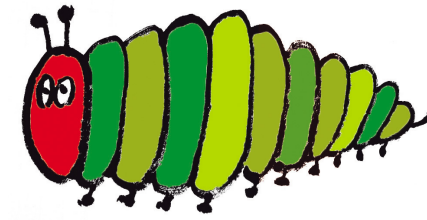
        @product_suggestions = SuggestionService.build_for(@bug)

        seriously.stop_reading_this!
        if @order.bug.caterpillar?
          @bug.very_hungry!
          basket = Chrysalis::Basket.create!(@bug)
          PromotionMailer.new_basket(basket).deliver_later
          @product_suggestions << Chrysalis.default_items_for(@bug)
        else
          @bug.book_suggestions.create!(
            Book.find_by(author: "Eric Carle", bug: @order.bug)
          )
        end

        @bug.suggest(@product_suggestions)
        @bug.friends.each do |friend|
          @order.share_with(friend) if @bug.sharing_orders?
        end
      rescue => e
        handle_transaction_error(e)
      ensure
        nobody.should_have_read_any_of_this
      end
    end
  end
end
```



# Monday



```
def submit
  Order.transaction do
    Product.transaction do
      Chrysalis.transaction do
        code.not_meant_for_reading!
        @order.save!
        FulfillmentClient.submit_order(@order)
        @order.update_product_inventory

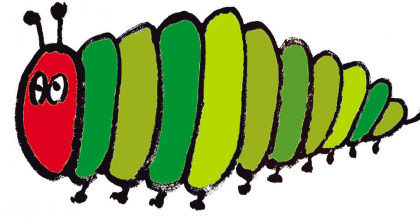
        @product_suggestions = SuggestionService.build_for(@bug)

        seriously.stop_reading_this!
        if @order.bug.caterpillar?
          @bug.very_hungry!
          basket = Chrysalis::Basket.create!(@bug)
          PromotionMailer.new_basket(basket).deliver_later
          @product_suggestions << Chrysalis.default_items_for(@bug)
        else
          @bug.book_suggestions.create!(
            Book.find_by(author: "Eric Carle", bug: @order.bug)
          )
        end
      end

      @bug.suggest(@product_suggestions)
      @bug.friends.each do |friend|
        @order.share_with(friend) if @bug.sharing_orders?
      end
    rescue => e
      handle_transaction_error(e)
    ensure
      nobody.should_have_read_any_of_this
    end
  end
end
end
end
```

## Small Incremental Changes

# Monday



```
def submit
  Order.transaction do
    Product.transaction do
      Chrysalis.transaction do
        code.not_meant_for_reading!
        @order.save!
        FulfillmentClient.submit_order(@order)
        @order.update_product_inventory

        @product_suggestions = SuggestionService.build_for(@bug)

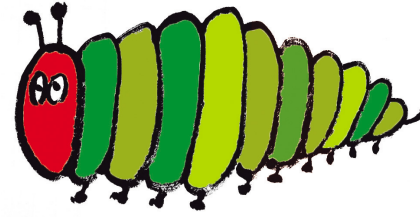
        seriously.stop_reading_this!
        if @order.bug.caterpillar?
          @bug.very_hungry!
          basket = Chrysalis::Basket.create!(@bug)
          PromotionMailer.new_basket(basket).deliver_later
          @product_suggestions << Chrysalis.default_items_for(@bug)
        else
          @bug.book_suggestions.create!(
            Book.find_by(author: "Eric Carle", bug: @order.bug)
          )
        end
      end

      @bug.suggest(@product_suggestions)
      @bug.friends.each do |friend|
        @order.share_with(friend) if @bug.sharing_orders?
      end
    rescue => e
      handle_transaction_error(e)
    ensure
      nobody.should_have_read_any_of_this
    end
  end
end
end
end
```

## Defer Until After Commit

- Rails 7.2 - *Automatically delay Active Job enqueues to after commit* [#51426](#)
- Rails 7.2 - *Allow to register transaction callbacks outside of a record* [#51474](#)

# Monday



```
def submit
  Order.transaction do
    Product.transaction do
      Chrysalis.transaction do
        code.not_meant_for_reading!
        @order.save!
        FulfillmentClient.submit_order(@order)
        @order.update_product_inventory

        @product_suggestions = SuggestionService.build_for(@bug)

        seriously.stop_reading_this!
        if @order.bug.caterpillar?
          @bug.very_hungry!
          basket = Chrysalis::Basket.create!(@bug)
          PromotionMailer.new_basket(basket).deliver_later
          @product_suggestions << Chrysalis.default_items_for(@bug)
        else
          @bug.book_suggestions.create!(
            Book.find_by(author: "Eric Carle", bug: @order.bug)
          )
        end
      end

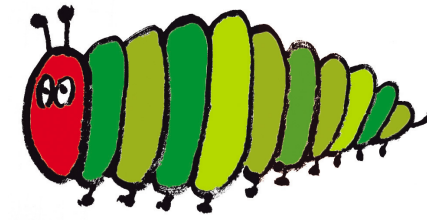
      @bug.suggest(@product_suggestions)
      @bug.friends.each do |friend|
        @order.share_with(friend) if @bug.sharing_orders?
      end
    rescue => e
      handle_transaction_error(e)
    ensure
      nobody.should_have_read_any_of_this
    end
  end
end
end
end
```

## Identify Other Problematic Transactions

- Database Observability
- Rails 7.1 - *Instrument Active Record transactions* [#49192](#)



# Monday



```
def submit
  Order.transaction do
    Product.transaction do
      Chrysalis.transaction do
        code.not_meant_for_reading!
        @order.save!
        FulfillmentClient.submit_order(@order)
        @order.update_product_inventory

        @product_suggestions = SuggestionService.build_for(@bug)

        seriously.stop_reading_this!
        if @order.bug.caterpillar?
          @bug.very_hungry!
          basket = Chrysalis::Basket.create!(@bug)
          PromotionMailer.new_basket(basket).deliver_later
          @product_suggestions << Chrysalis.default_items_for(@bug)
        else
          @bug.book_suggestions.create!(
            Book.find_by(author: "Eric Carle", bug: @order.bug)
          )
        end
      end

      @bug.suggest(@product_suggestions)
      @bug.friends.each do |friend|
        @order.share_with(friend) if @bug.sharing_orders?
      end
    rescue => e
      handle_transaction_error(e)
    ensure
      nobody.should_have_read_any_of_this
    end
  end
end
end
end
```

## Prevent Further Problematic Transactions

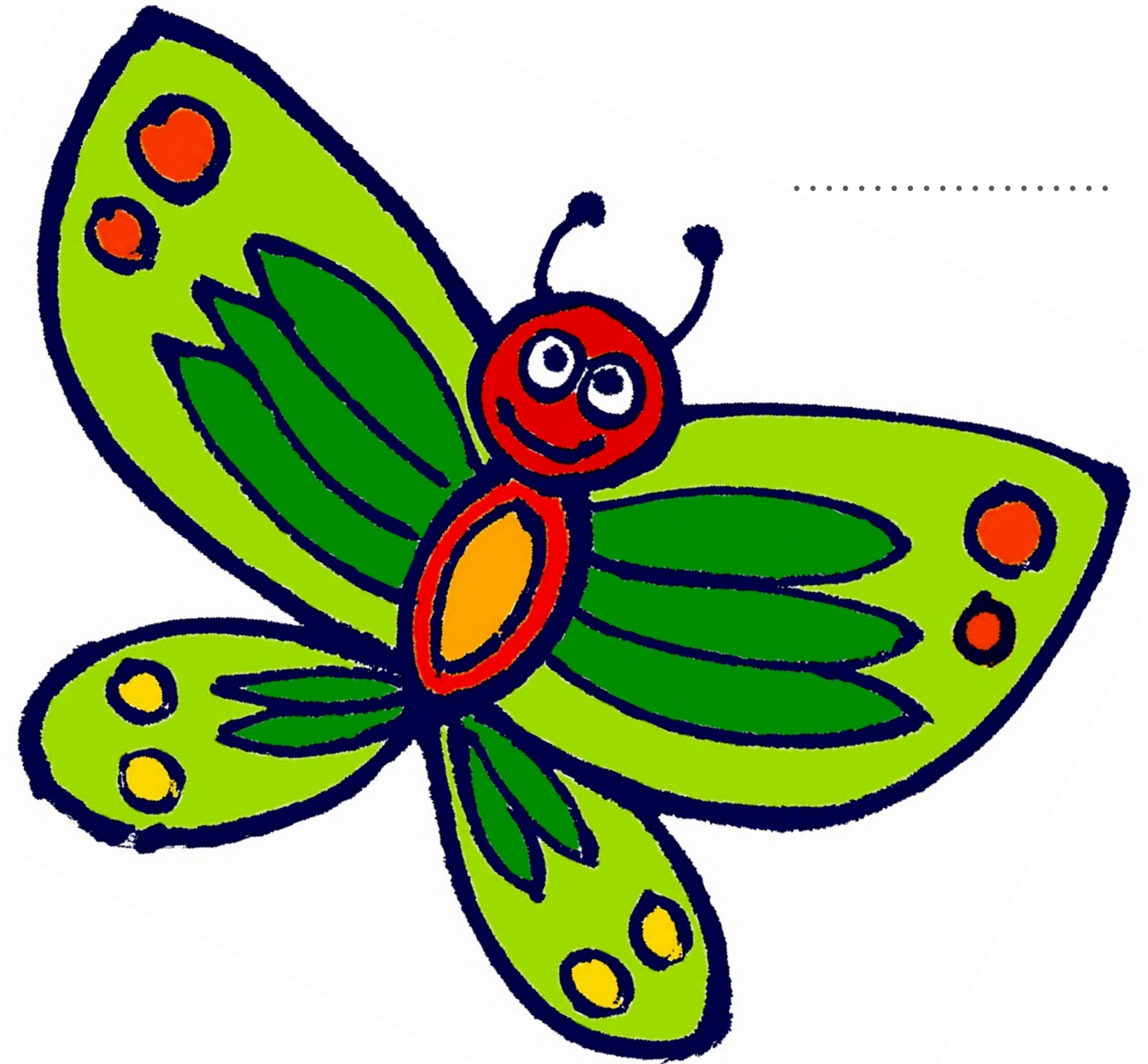


# Safe Transactions

---

## Keep Transactions Short

- Ideally < 1 Second

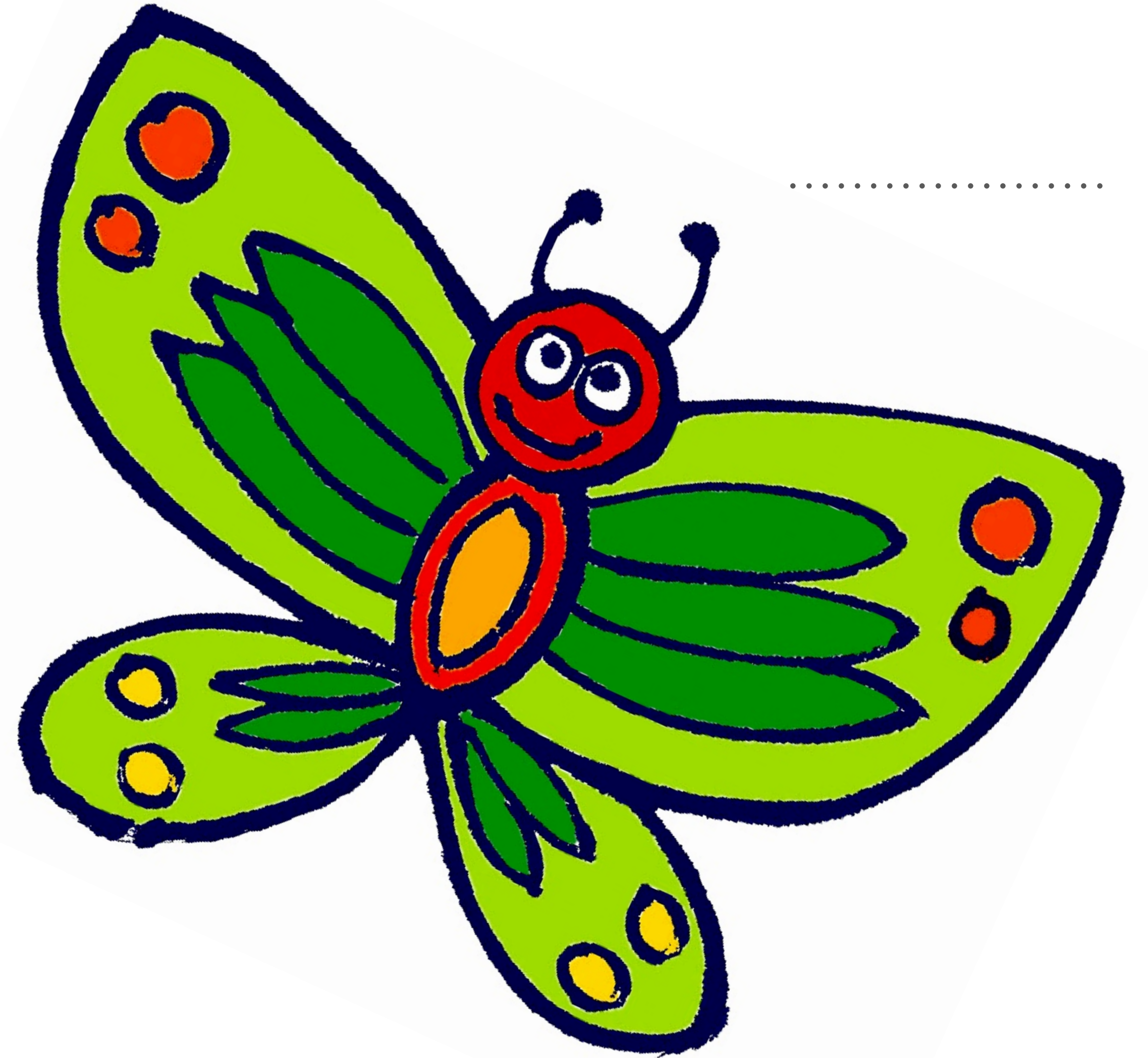




# Safe Transactions

---

**Fast Queries**



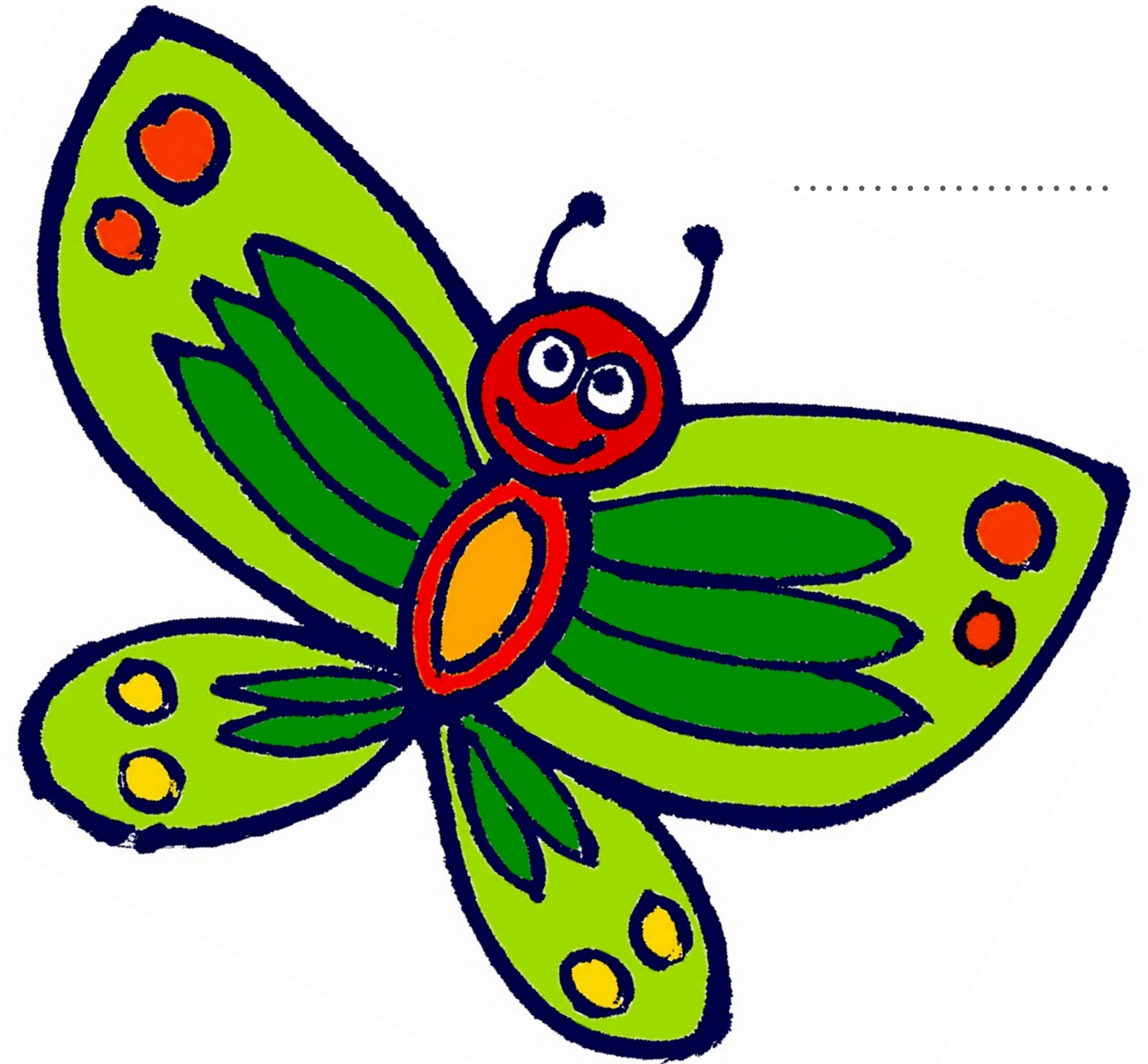


# Safe Transactions

---

## Limit # of Queries

- Ideally  $< 100$

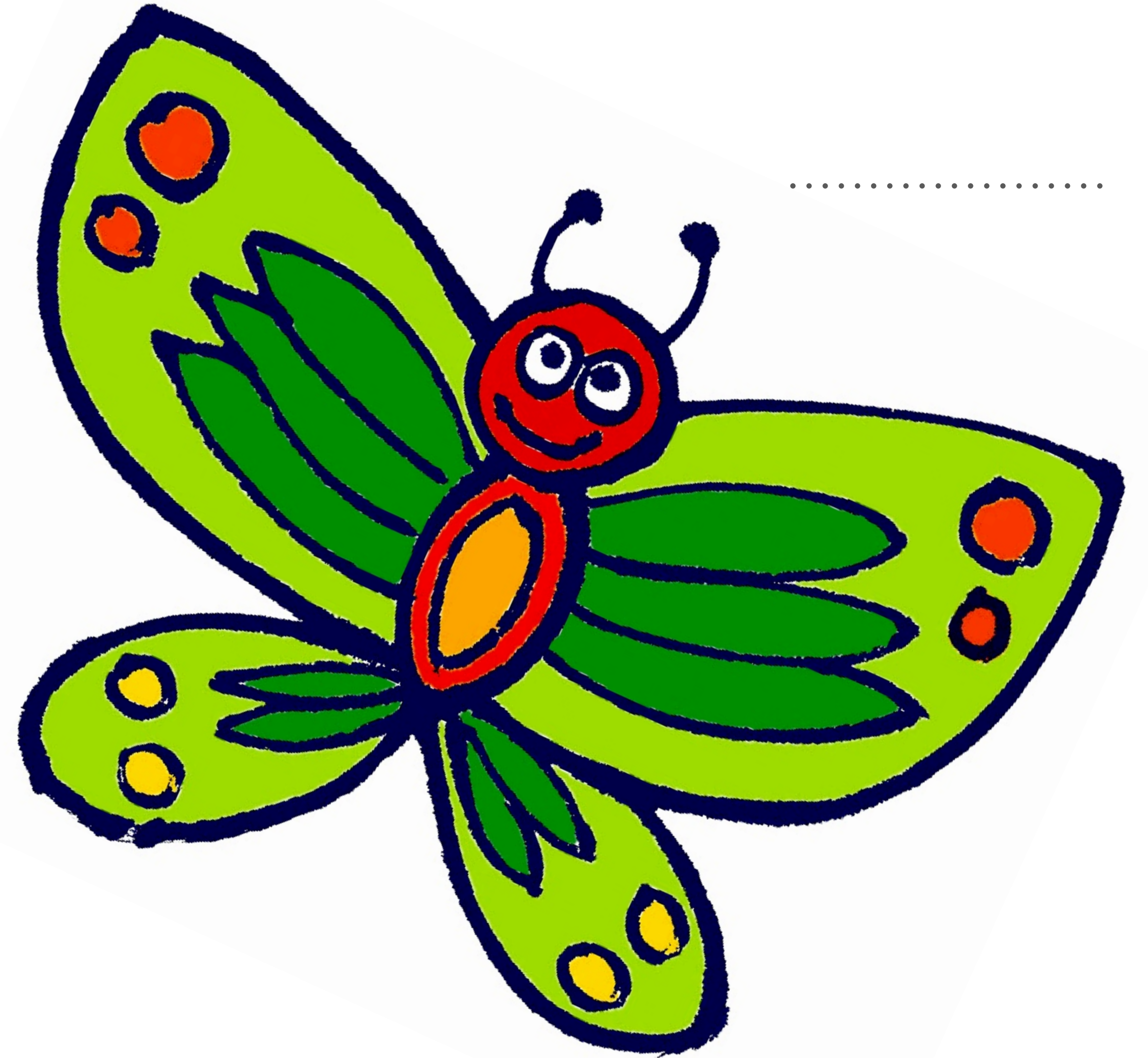




# Safe Transactions

---

**No External Calls**



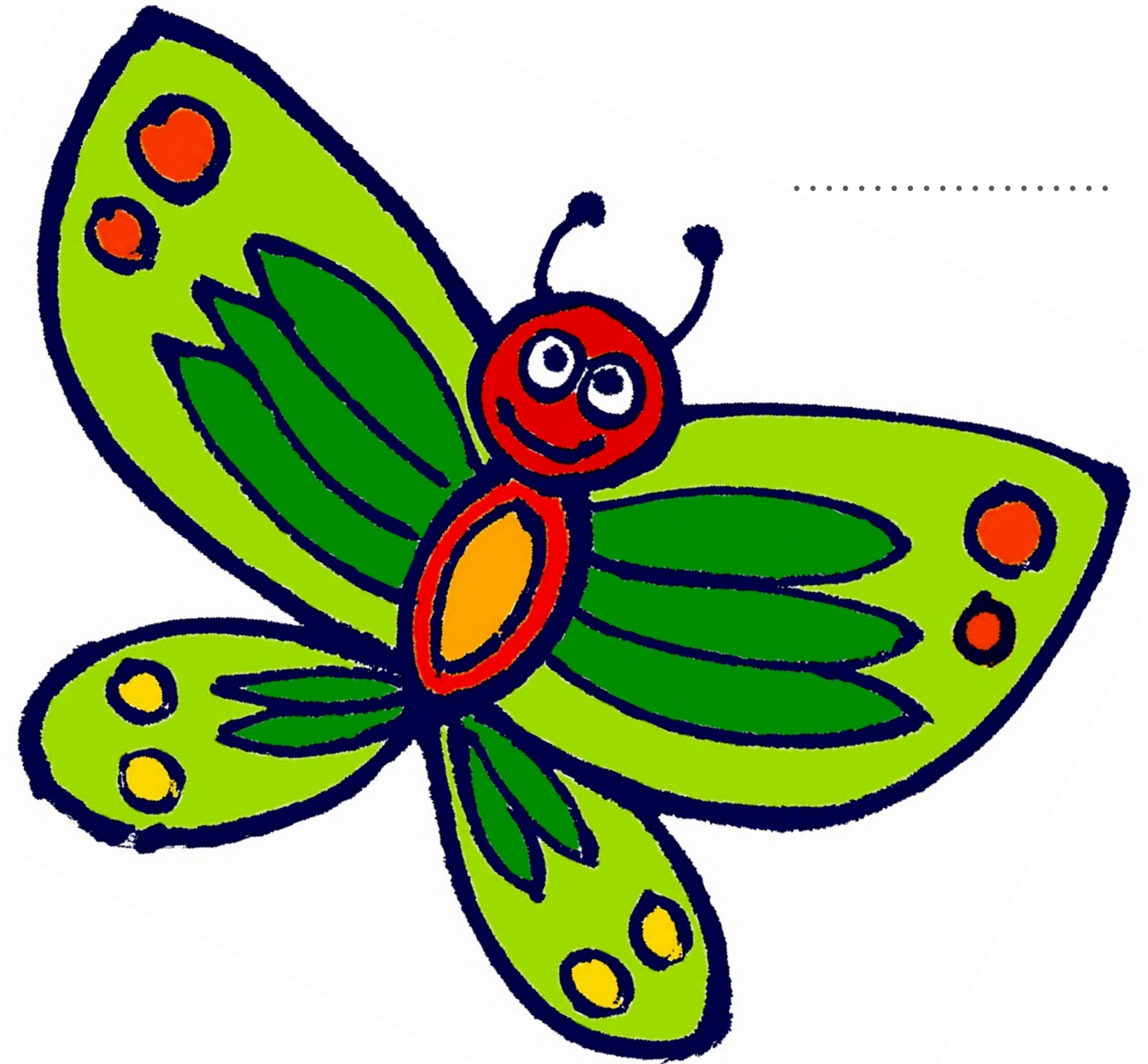


# Safe Transactions

---

## As Little Code as Possible

- Default to `after_commit` callbacks

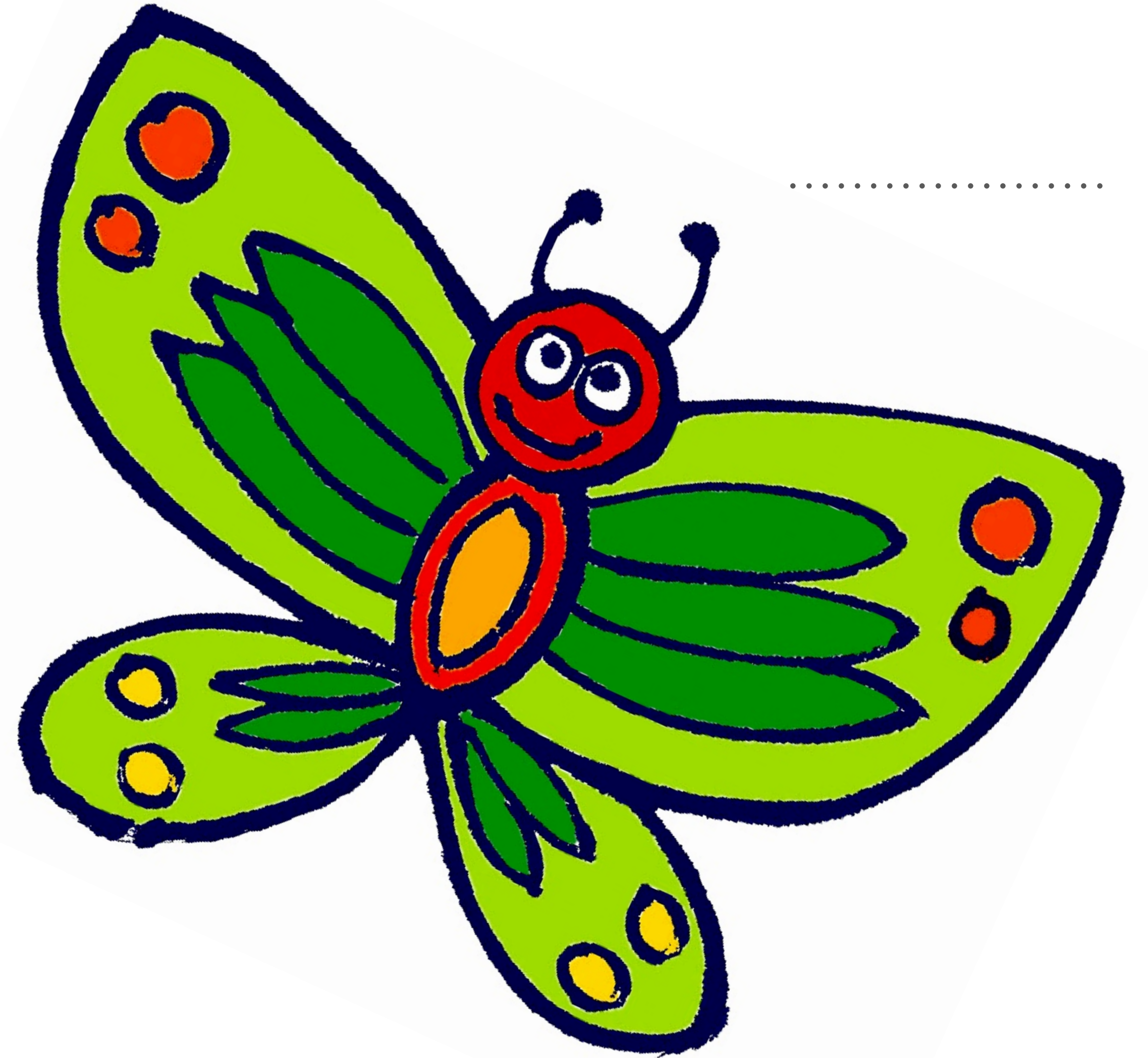




# Safe Transactions

---

**Do You Really Need a Transaction?**



# Daniel Colson

---

@composerinteralia

Illustrated by ChangHo Kim and DongBeom Kim