
Program will begin shortly…

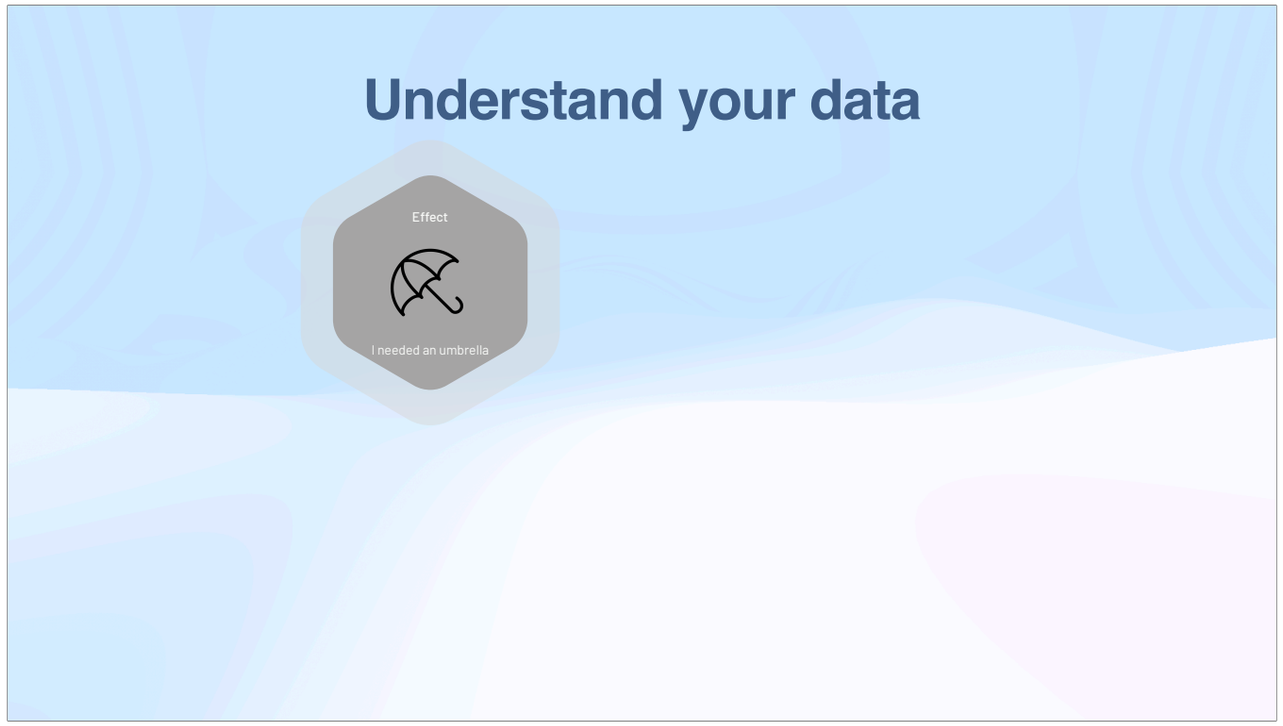

Program will begin shortly…


**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.

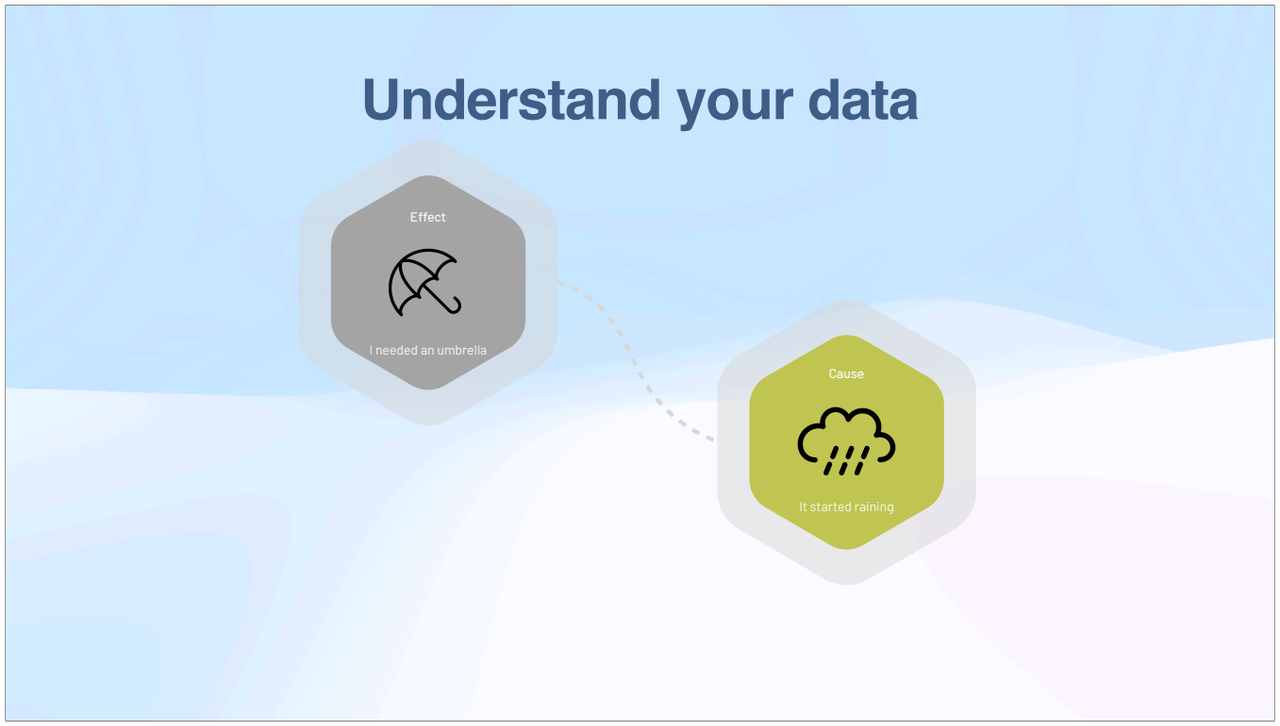

**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.

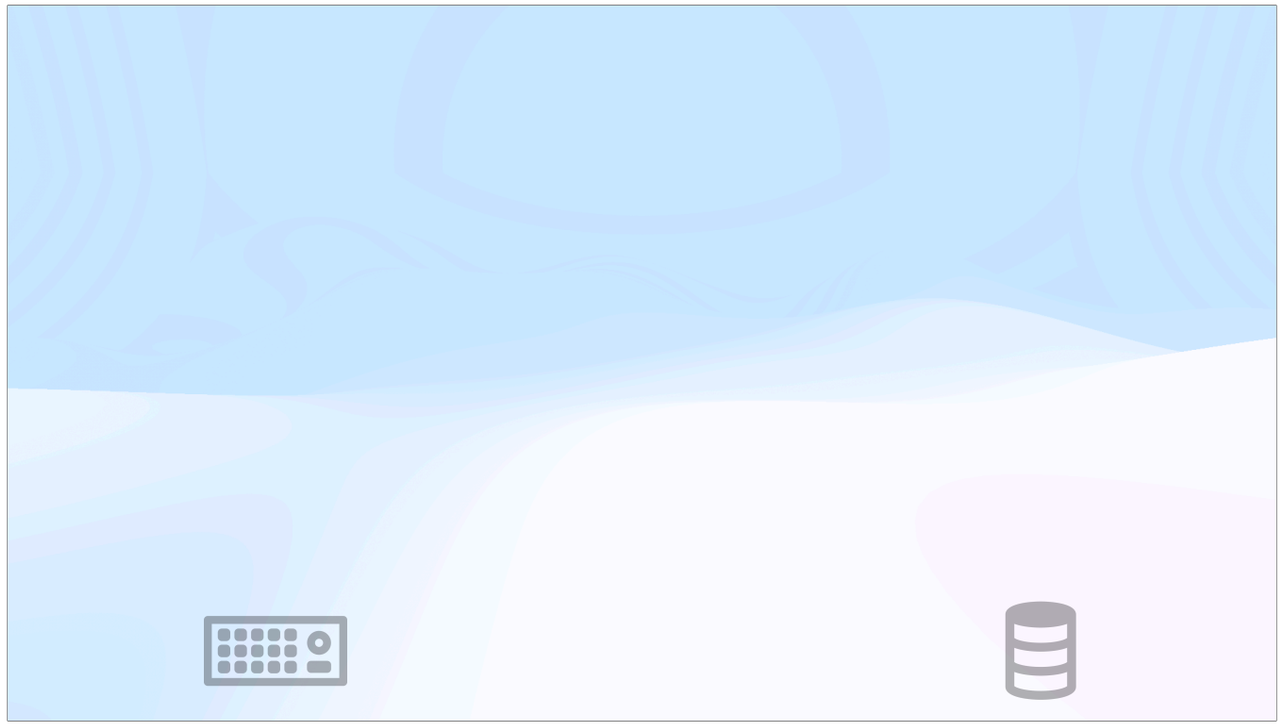

**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.
Welcome to Vestfold Dev
Let's talk about Event Sourcing

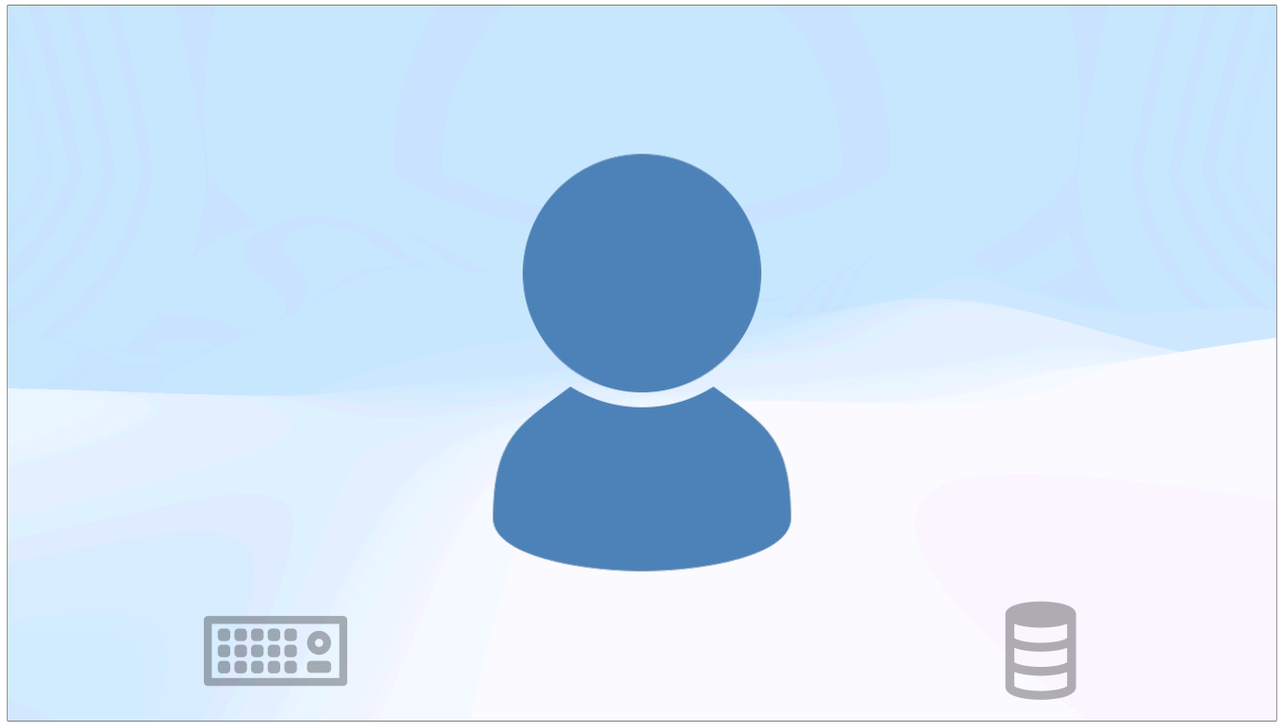
01 Event Sourcing



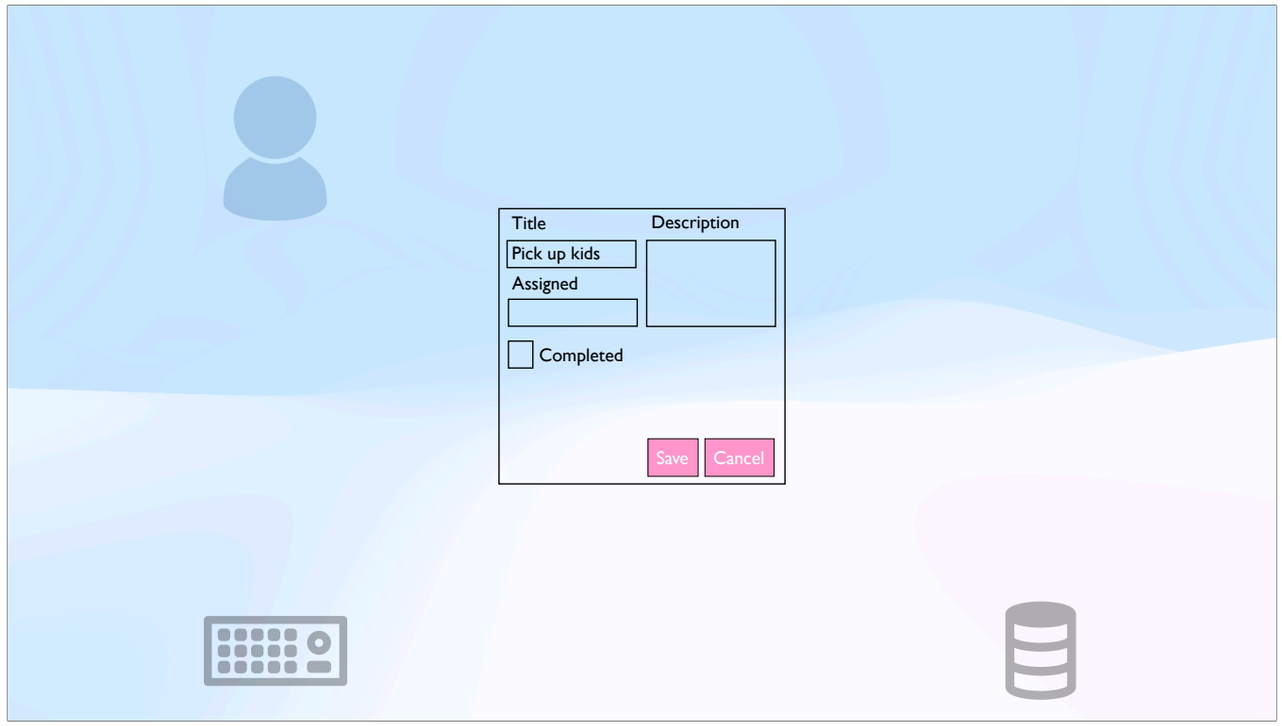
In traditional CRUD systems all we’re seeing is the effect; the conclusion. By default you can’t see how we got there, unless you look at logs or other unstructured artifacts. You might formalize audit trails. With event sourcing we’re flipping it on its head and focusing in on the cause, “how we got there”.
With this we can represent the conclusion, the current state in many ways.

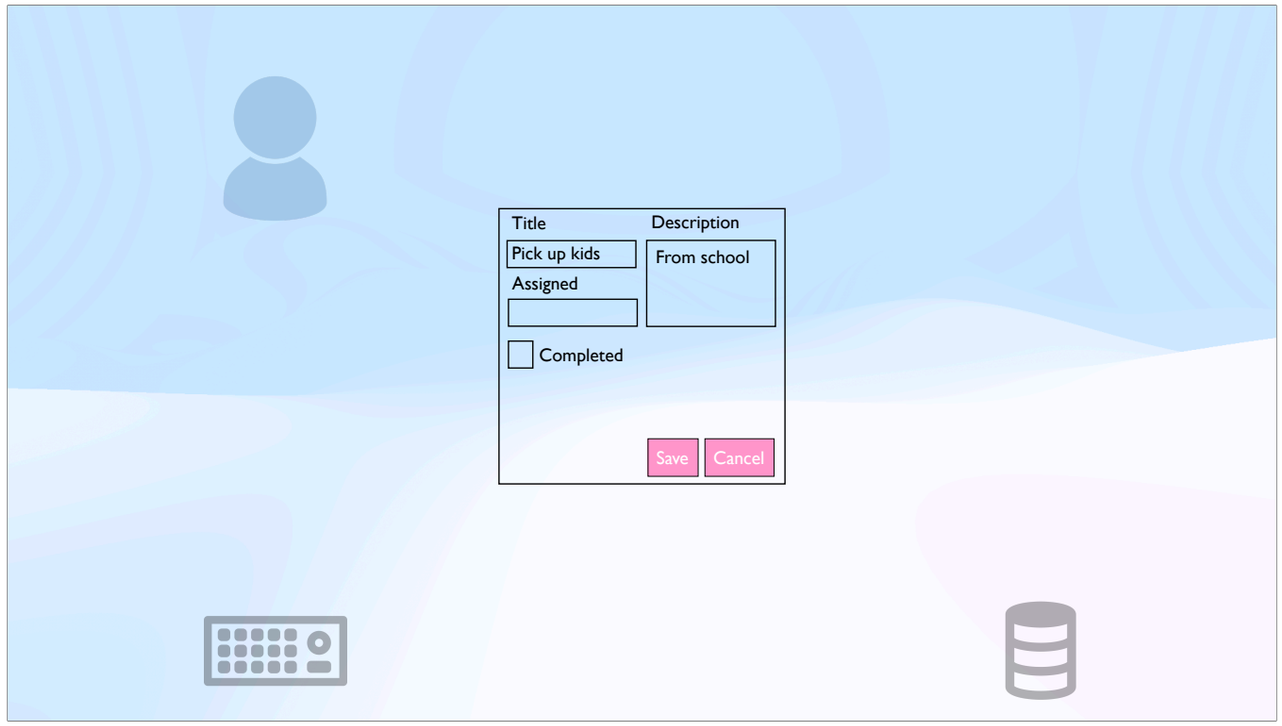

Understand your data
Effect
I needed an umbrella
Cause
It started raining

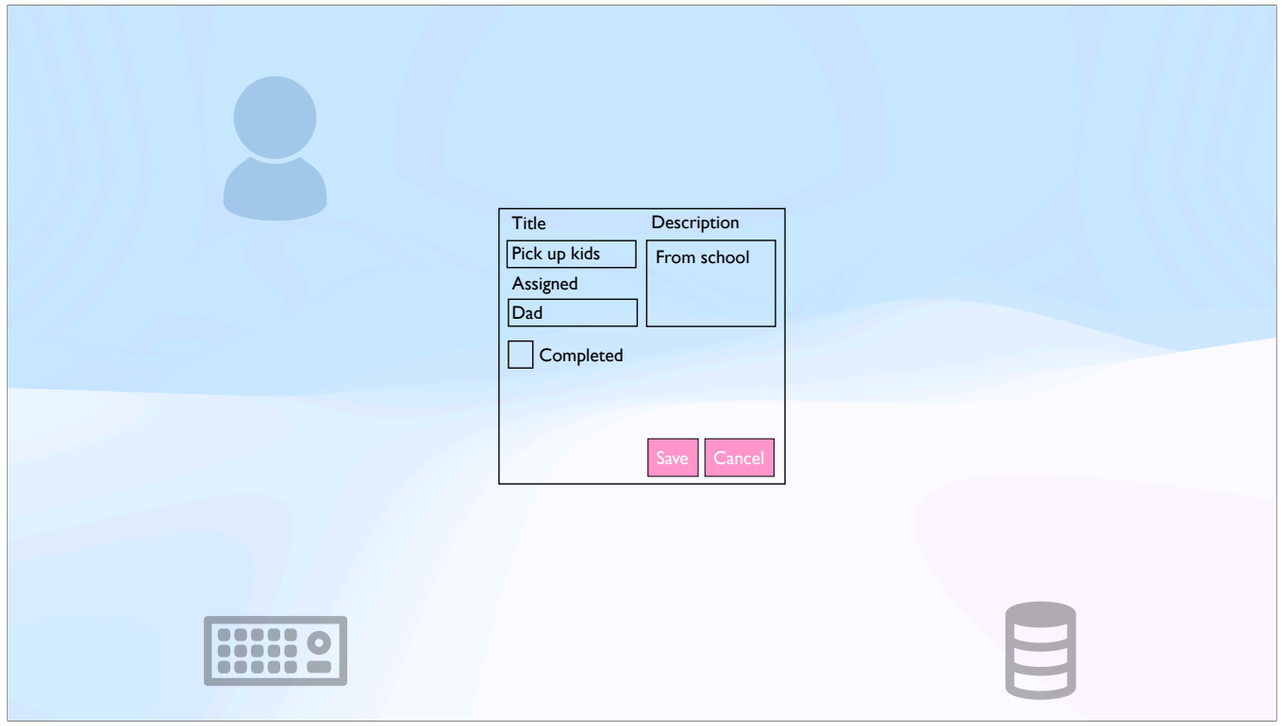




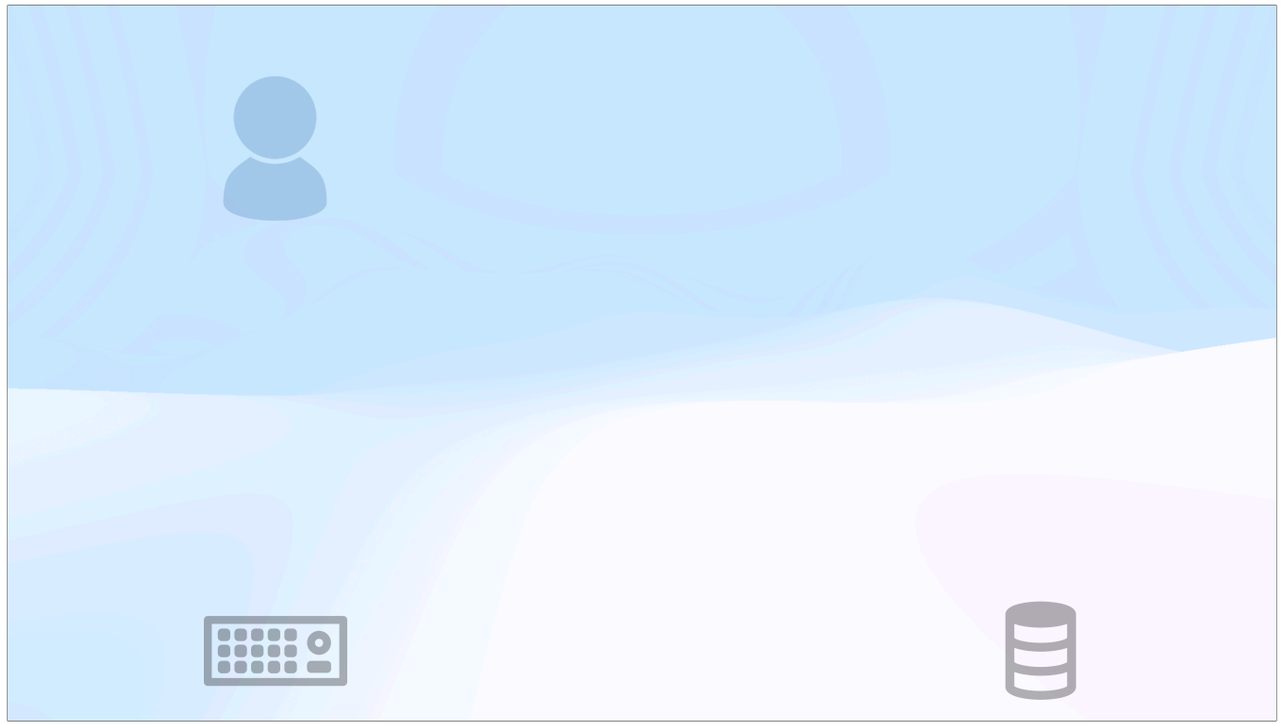

ID
Title
Action
123
Pick up kids
Edit
456
Make dinner
Edit
789
Go to sleep
Edit


Title
Description
Pick up kids
Assigned
Completed
Save
Cancel


Title
Description
Pick up kids
From school
Assigned
Completed
Save
Cancel


Title
Description
Pick up kids
From school
Assigned
Dad
Completed
Save
Cancel

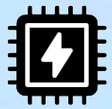

Title
Description
Pick up kids
From school
Assigned
Dad
Completed
Save
Cancel



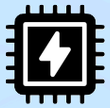
Events

The meaning behind event is not ubiquituous, its context sensitive.
IoT = They are in fact in most cases not events, but measurements - we ask a sensor to perform a measurement.

Events

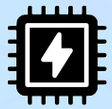


Events



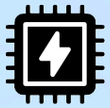
```
        break;
    case WM_MOUSEMOVE:
        
        CaptureMousePosition(lParam, hWnd);

        break;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        //TextOut(GetDC(hWnd), 100, 100, szBuffer, _countof(szBuffer));
```

Events



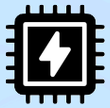
```
        break;
    case WM_MOUSEMOVE:
        
        CaptureMousePosition(lParam, hWnd);

        break;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        //TextOut(GetDC(hWnd), 100, 100, szBuffer, _countof(szBuffer));
```

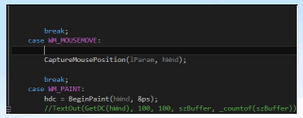
Events



```
        break;
    case WM_MOUSEMOVE:
        
        CaptureMousePosition(lParam, hWnd);

        break;
    case WM_PAINT:
        hdc = BeginPaint(hWnd, &ps);
        //TextOut(GetDC(hWnd), 100, 100, szBuffer, _countof(szBuffer));
```

Events



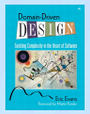

break;
case WM_MOUSEMOVE:
CapturedMousePosition(lParam, hWnd);
break;
case WM_PAINT:
hdc = BeginPaint(hWnd, &ps);
//TextOut(GetDC(hWnd), 100, 100, szBuffer, _countof(szBuffer));


HTML

"An event represents a **meaningful business decision taken**, which alters the state of the system"

Yves Goeleven

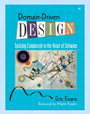
Domain Events



- Capture something meaningful in the domain
- Has a clear name
- Name is past tense - something that has happened
- Hold just the properties that represents a meaningful change
- Represents a state change in your system

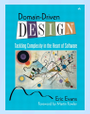
An event represents a meaningful business decision taken which alters the state of the system

Domain Events

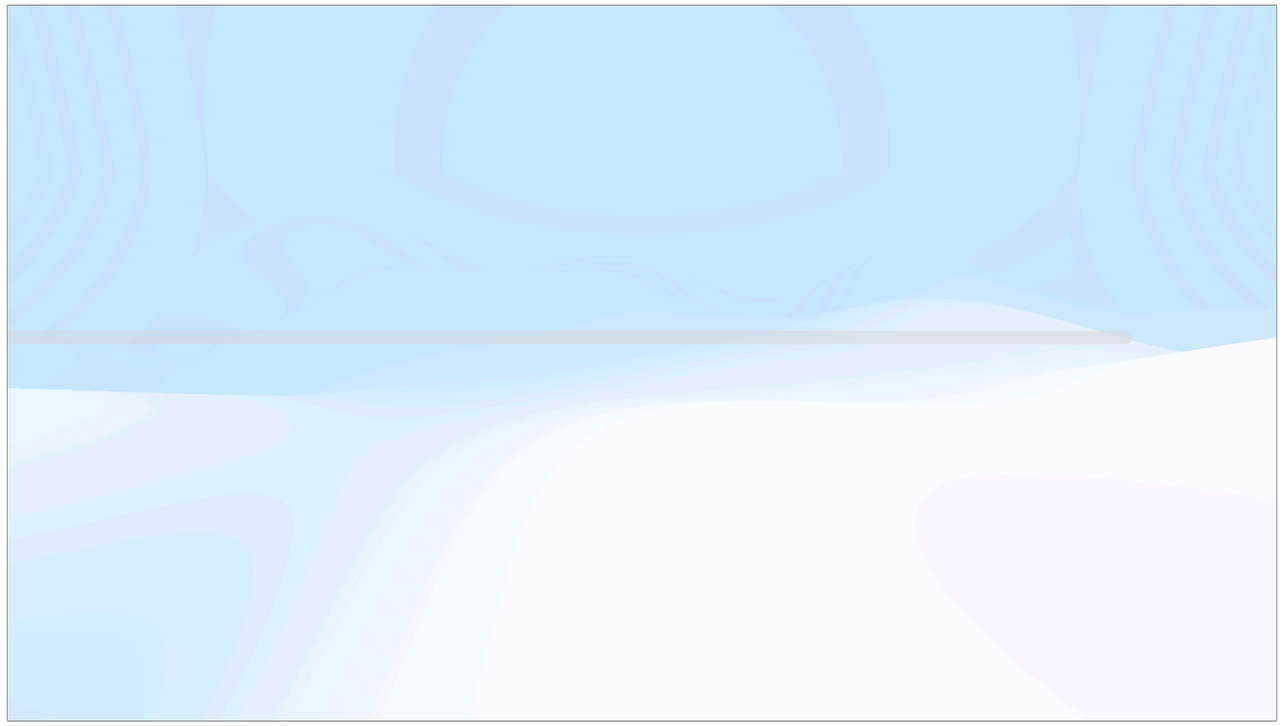


- Capture something meaningful in the domain
- Has a clear name
- Name is past tense - something that has happened
- Hold just the properties that represents a meaningful change
- Represents a state change in your system
- This is ***not*** a CRUD operation (No EmployeeUpdatedEvent)

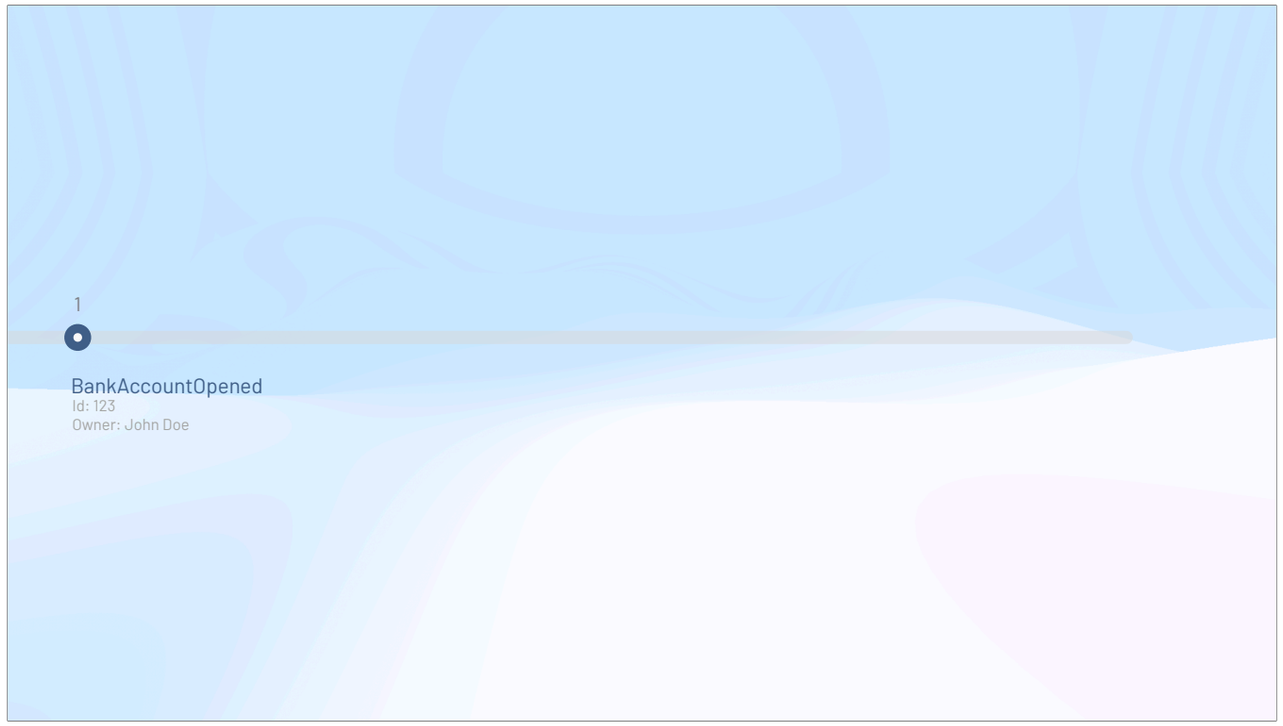
Domain Events

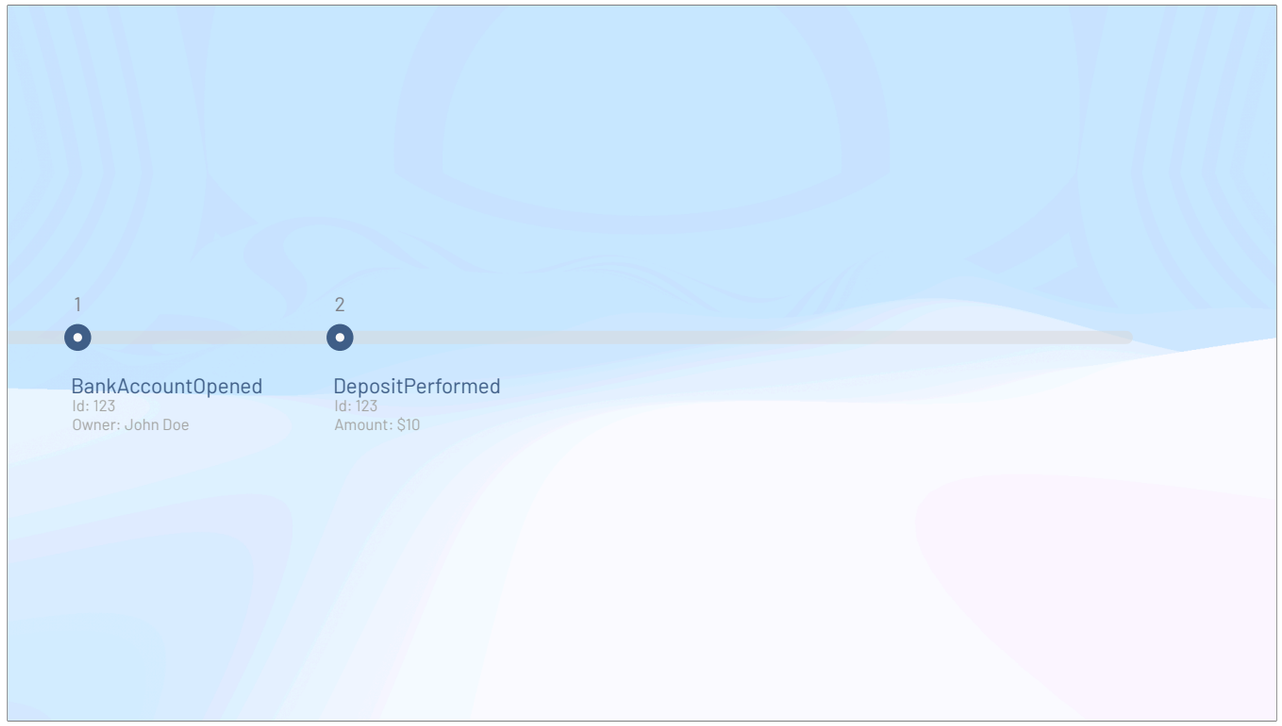


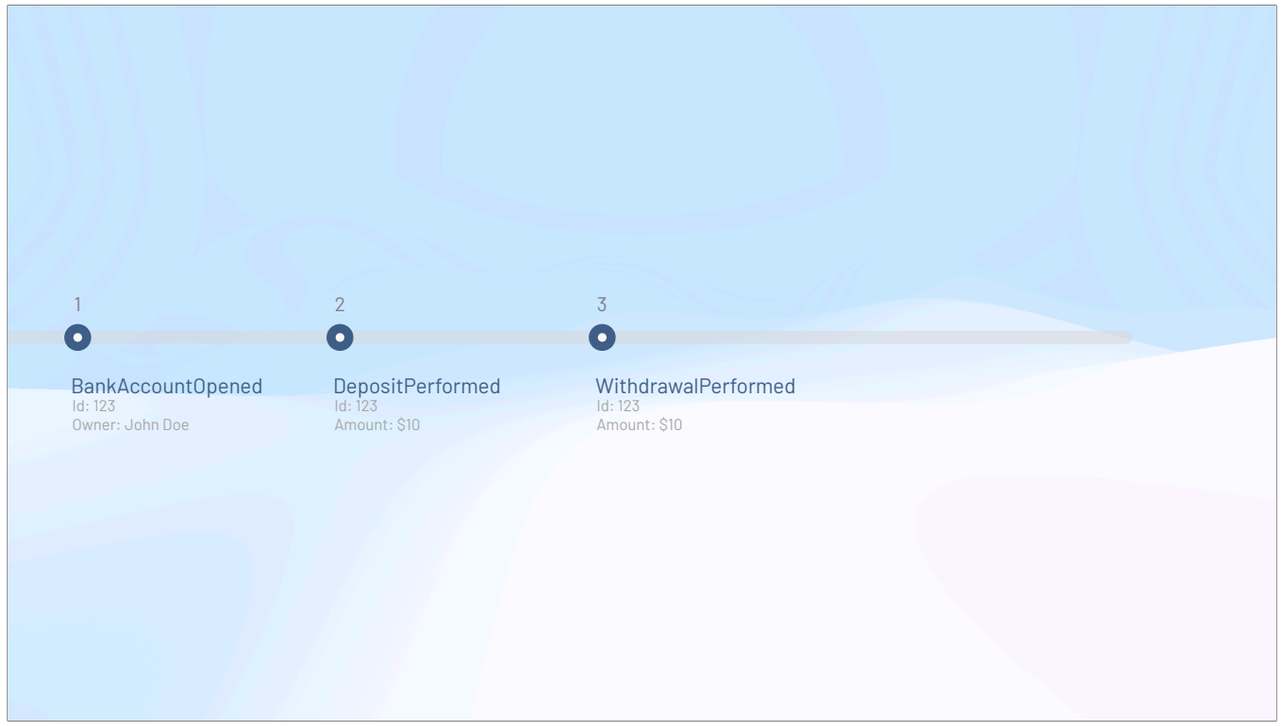
- Capture something meaningful in the domain
- Has a clear name
- Name is past tense - something that has happened
- Hold just the properties that represents a meaningful change
- Represents a state change in your system
- This is ***not*** a CRUD operation (No EmployeeUpdatedEvent)
- This is ***not*** a “TemperatureValue” type of IOT event

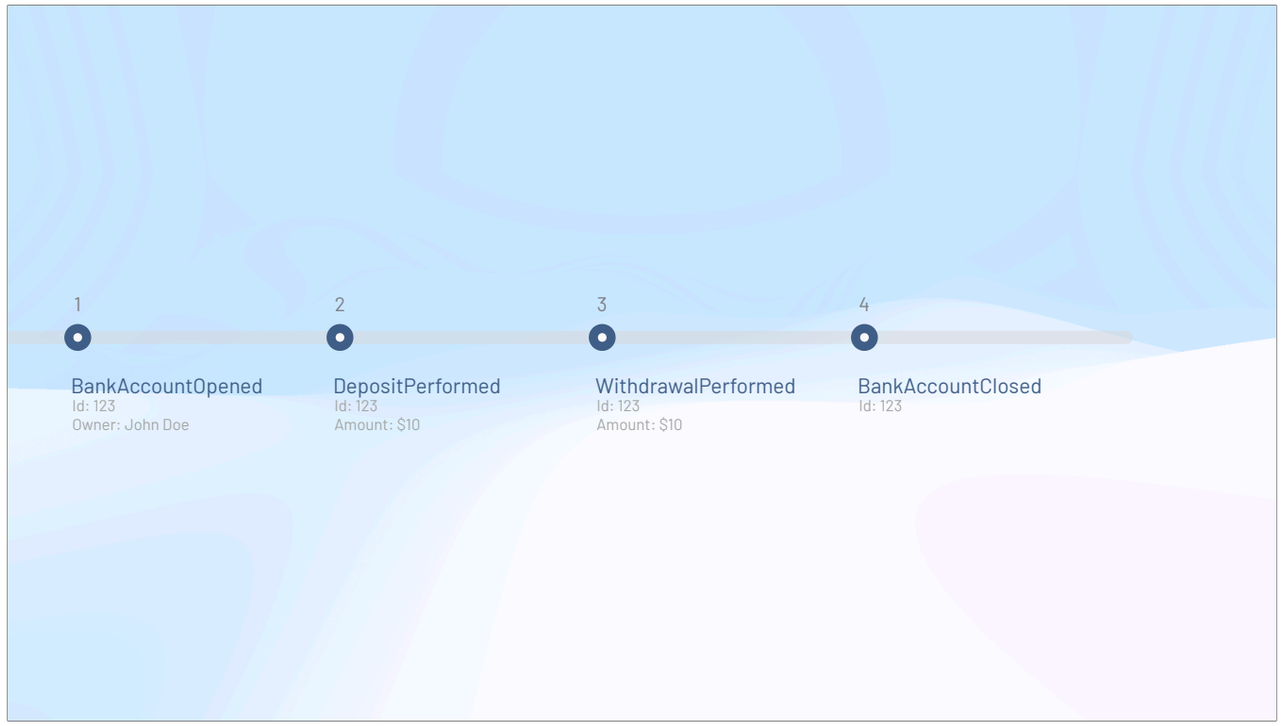


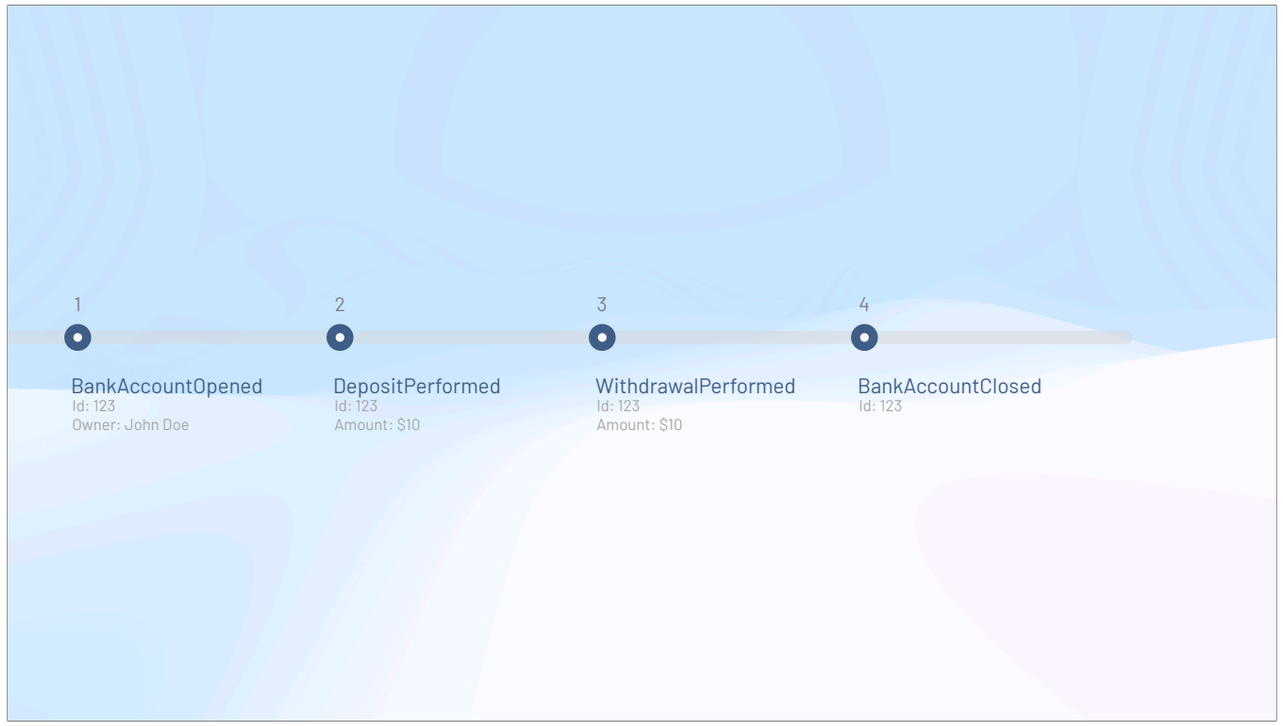
Terminology:
Log - the Event Log
Streams
Sequences

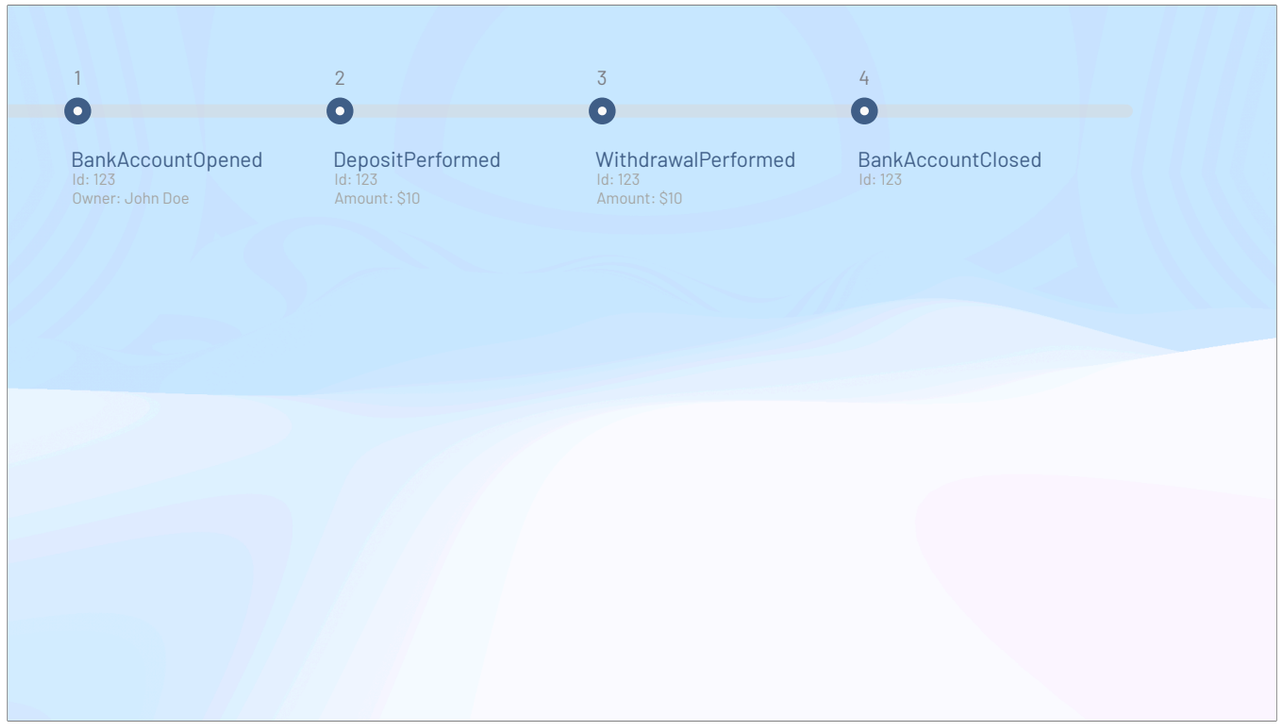

1
BankAccountOpened
Id: 123
Owner: John Doe

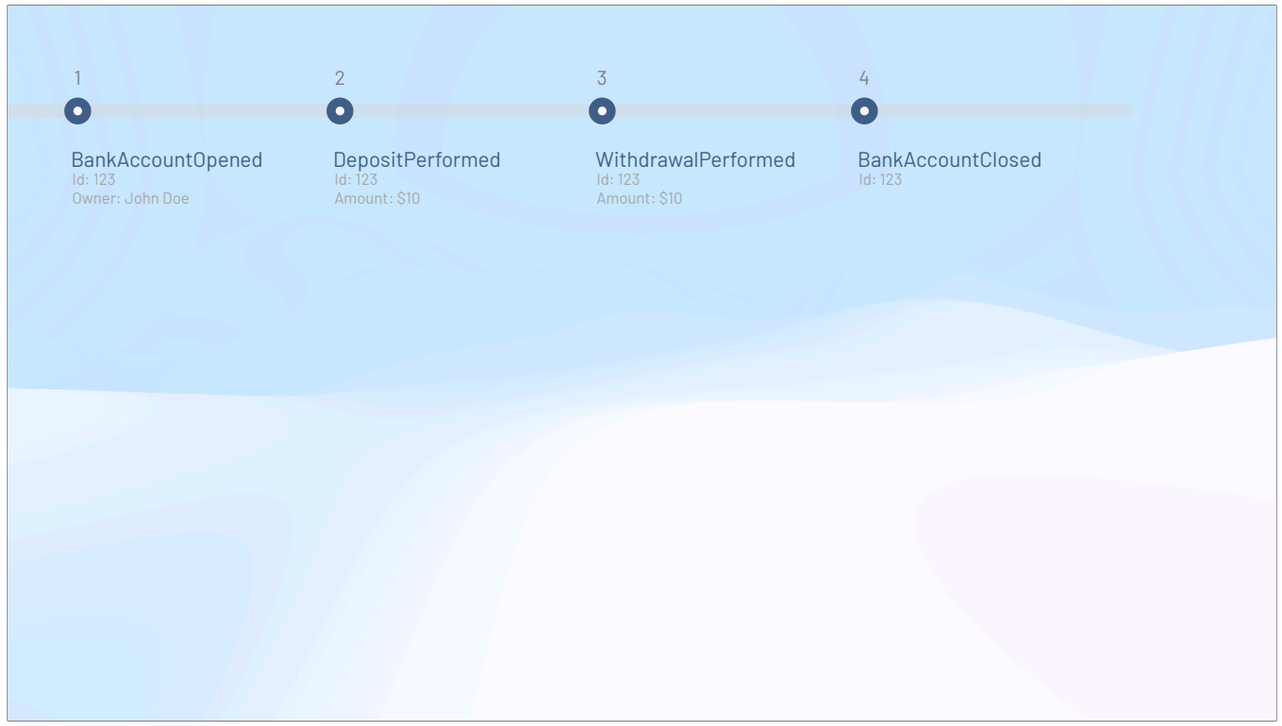

1
2
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10


1
2
3
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10
WithdrawalPerformed
Id: 123
Amount: $10

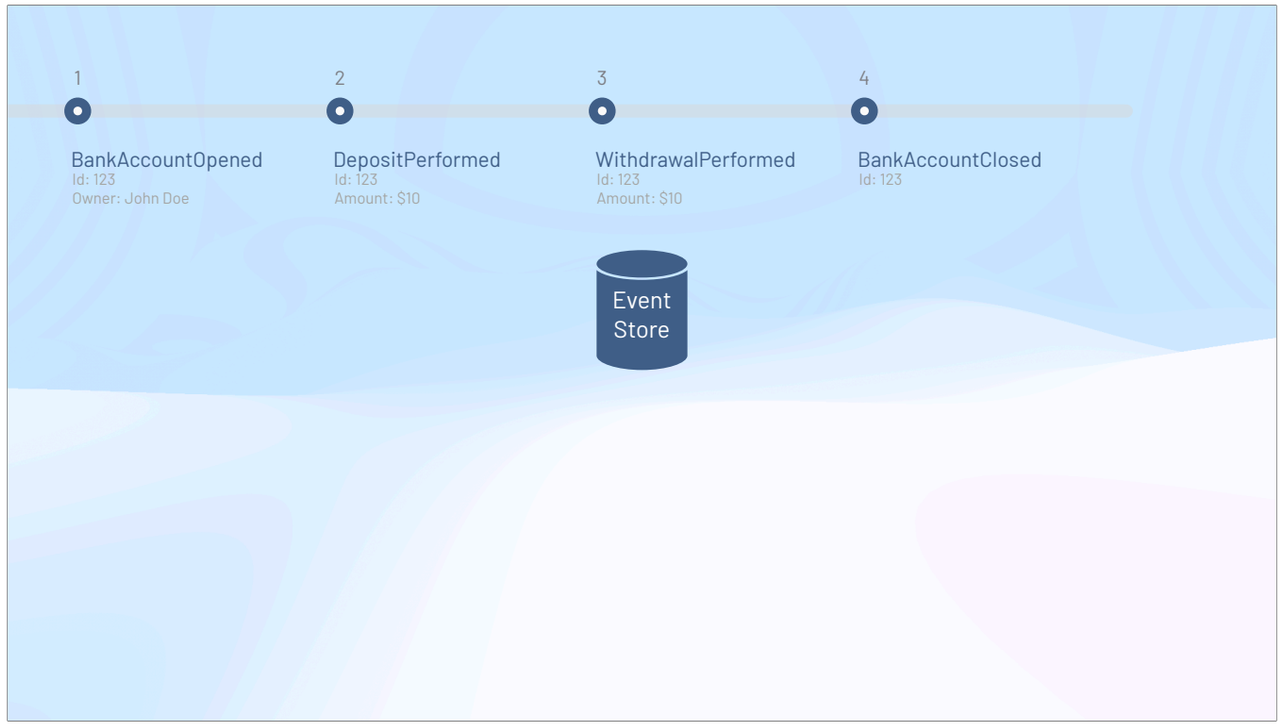

1
BankAccountOpened
Id: 123
Owner: John Doe
2
DepositPerformed
Id: 123
Amount: $10
3
WithdrawalPerformed
Id: 123
Amount: $10
4
BankAccountClosed
Id: 123

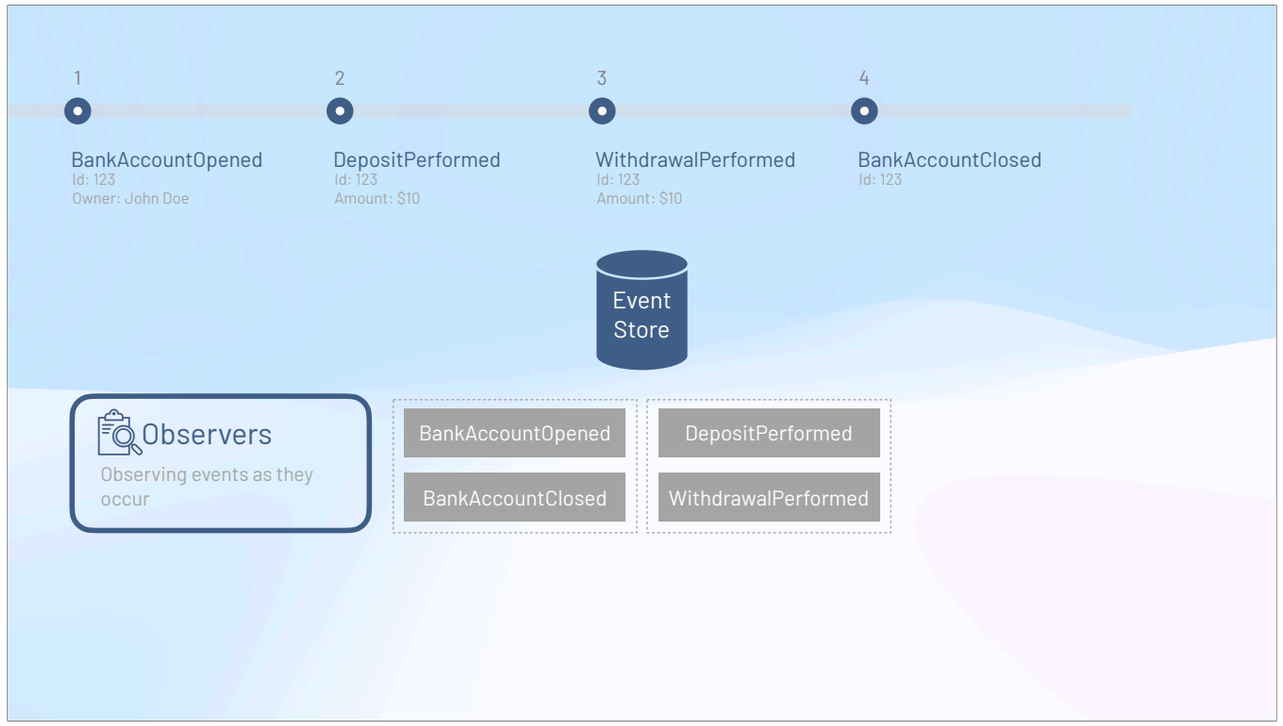

1
BankAccountOpened
Id: 123
Owner: John Doe
2
DepositPerformed
Id: 123
Amount: $10
3
WithdrawalPerformed
Id: 123
Amount: $10
4
BankAccountClosed
Id: 123

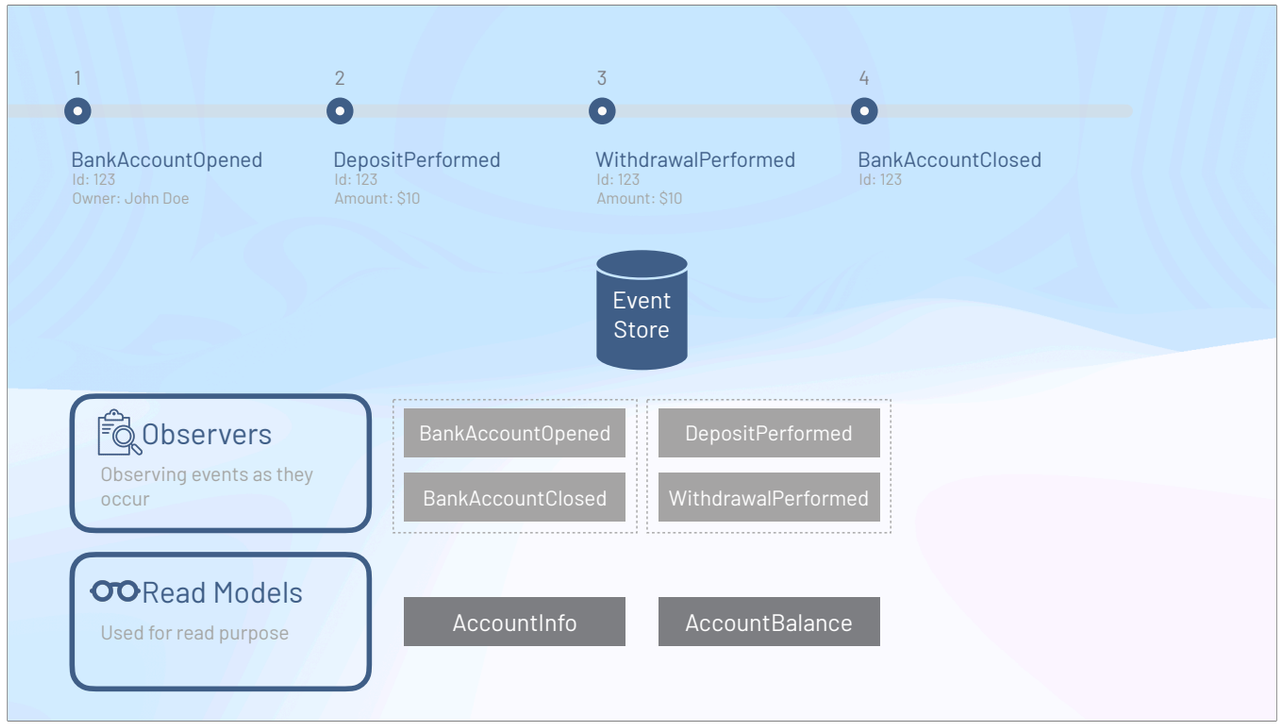

1
BankAccountOpened
Id: 123
Owner: John Doe
2
DepositPerformed
Id: 123
Amount: $10
3
WithdrawalPerformed
Id: 123
Amount: $10
4
BankAccountClosed
Id: 123

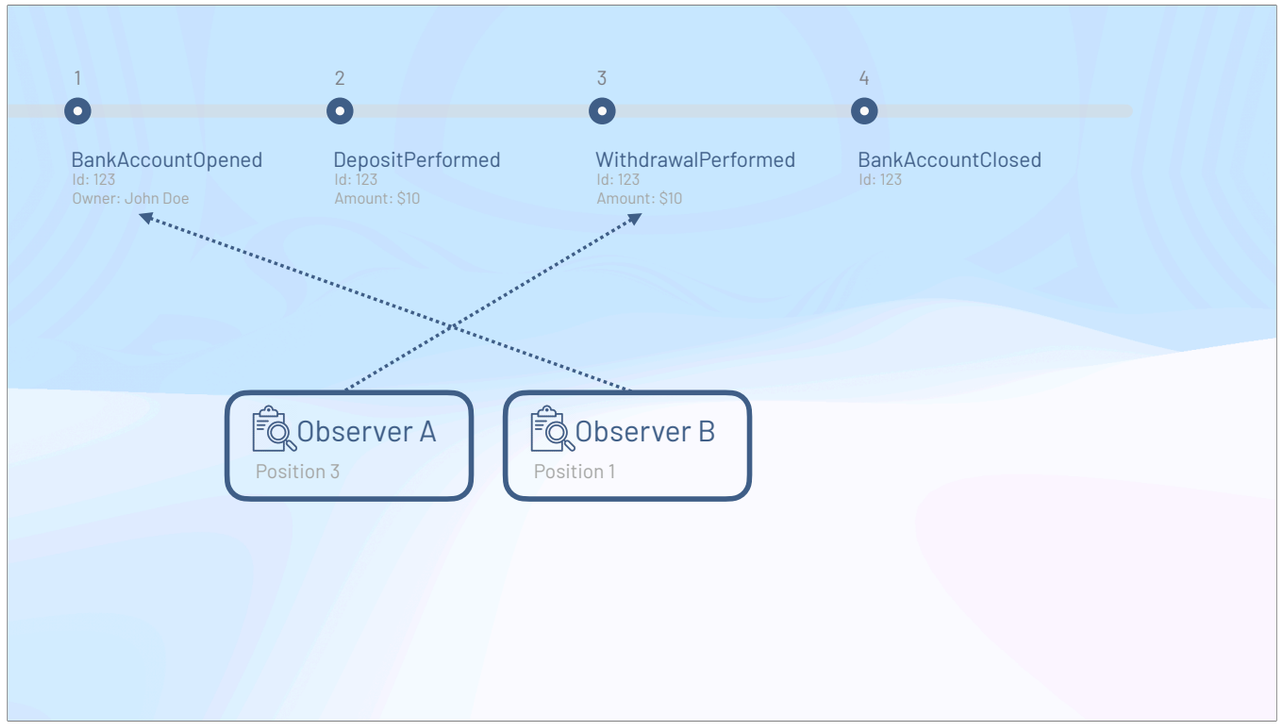


If an observer changes what it is interested in, it needs to replay (only if the change involves looking at event types that has events already registered)

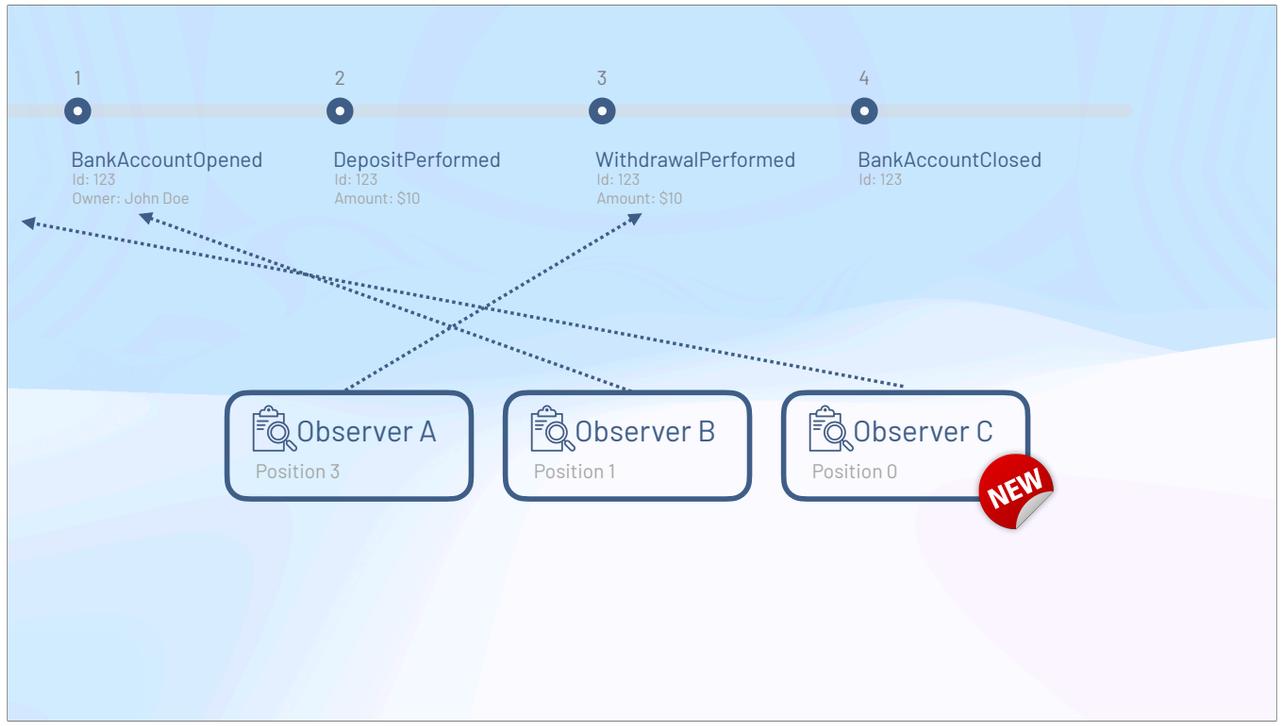

1
BankAccountOpened
Id: 123
Owner: John Doe
2
DepositPerformed
Id: 123
Amount: $10
3
WithdrawalPerformed
Id: 123
Amount: $10
4
BankAccountClosed
Id: 123
Event
Store


1
2
3
4
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10
WithdrawalPerformed
Id: 123
Amount: $10
BankAccountClosed
Id: 123
Event Store
Observers
Observing events as they occur
BankAccountOpened
DepositPerformed
BankAccountClosed
WithdrawalPerformed


1
2
3
4
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10
WithdrawalPerformed
Id: 123
Amount: $10
BankAccountClosed
Id: 123
Event Store
Observers
Observing events as they occur
BankAccountOpened
DepositPerformed
BankAccountClosed
WithdrawalPerformed
Read Models
Used for read purpose
AccountInfo
AccountBalance



Competing consumer pattern


1
2
3
4
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10
WithdrawalPerformed
Id: 123
Amount: $10
BankAccountClosed
Id: 123
Observer A
Position 3
Observer B
Position 1
Observer C
Position 0
NEW

MapReduce

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(item => item * 2);
console.log(doubled); // [2, 4, 6, 8]
```

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce(function (result, item) {
  return result + item;
}, 0);
console.log(sum); // 10
```

```
array = [1, 2, 3]
new_array = array.map { |element| element * 2 }
p array
p new_array
```

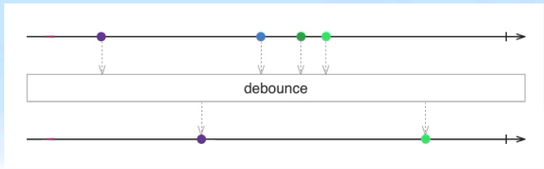
```
array = [1, 2, 3, 4, 5]
array.reduce { |sum, n| sum + n }
```

```
var result = numbers.Select(x => x + 3);
```

```
var sum = number.Aggregate((x, y) => x + y);
```

```
List<Integer> list = Arrays.asList(1, 4, 9, 16);
Stream<Integer> s = list.stream();
Stream<Double> t = s.map(x -> Math.sqrt(x));
```

```
Arrays.asList(1,2,3).stream()
   .reduce(0, (x,y) -> x+y)
```




```
import { fromEvent, takeUntil, interval, map, reduce } from 'rxjs';

const clicksInFiveSeconds = fromEvent(document, 'click')
  .pipe(takeUntil(interval(5000)));

const ones = clicksInFiveSeconds.pipe(map(() => 1));
const seed = 0;
const count = ones.pipe(reduce((acc, one) => acc + one, seed));

count.subscribe(x => console.log(x));
```

MapReduce

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(item => item * 2);
console.log(doubled); // [2, 4, 6, 8]
```

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce(function (result, item) {
  return result + item;
}, 0);
console.log(sum); // 10
```

```
array = [1, 2, 3]
new_array = array.map { |element| element * 2 }
p array
p new_array
```

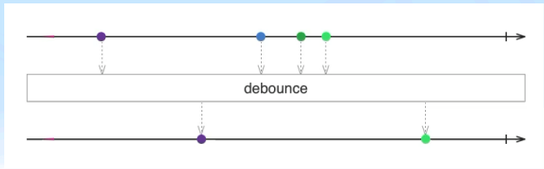
```
array = [1, 2, 3, 4, 5]
array.reduce { |sum, n| sum + n }
```

```
var result = numbers.Select(x => x + 3);
```

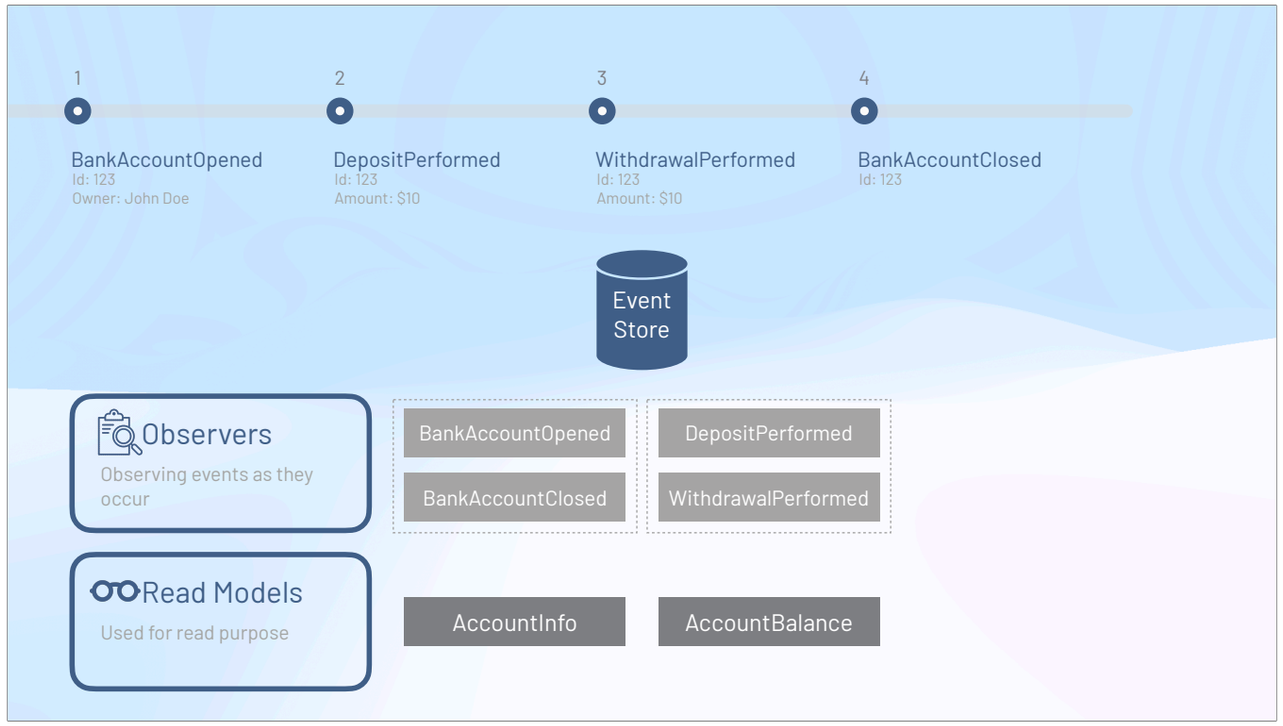
```
var sum = number.Aggregate((x, y) => x + y);
```

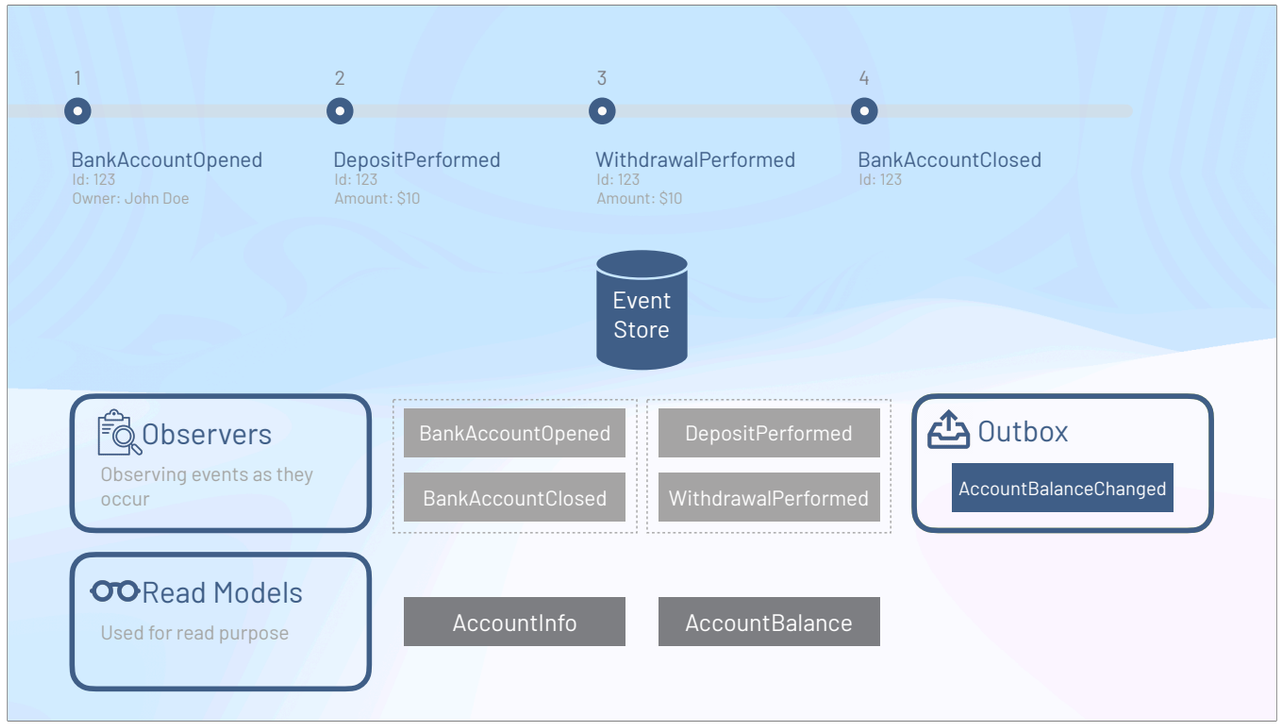
```
List<Integer> list = Arrays.asList(1, 4, 9, 16);
Stream<Integer> s = list.stream();
Stream<Double> t = s.map(x -> Math.sqrt(x));
```

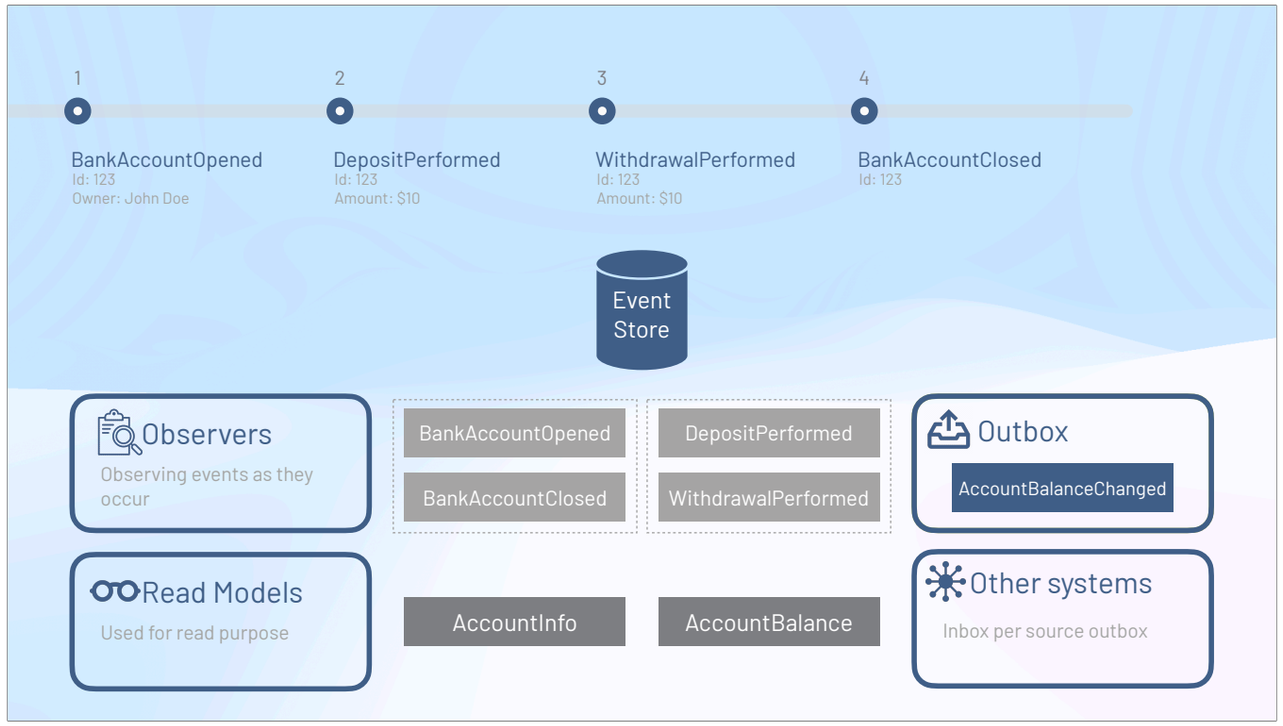
```
Arrays.asList(1,2,3).stream()
   .reduce(0, (x,y) -> x+y)
```



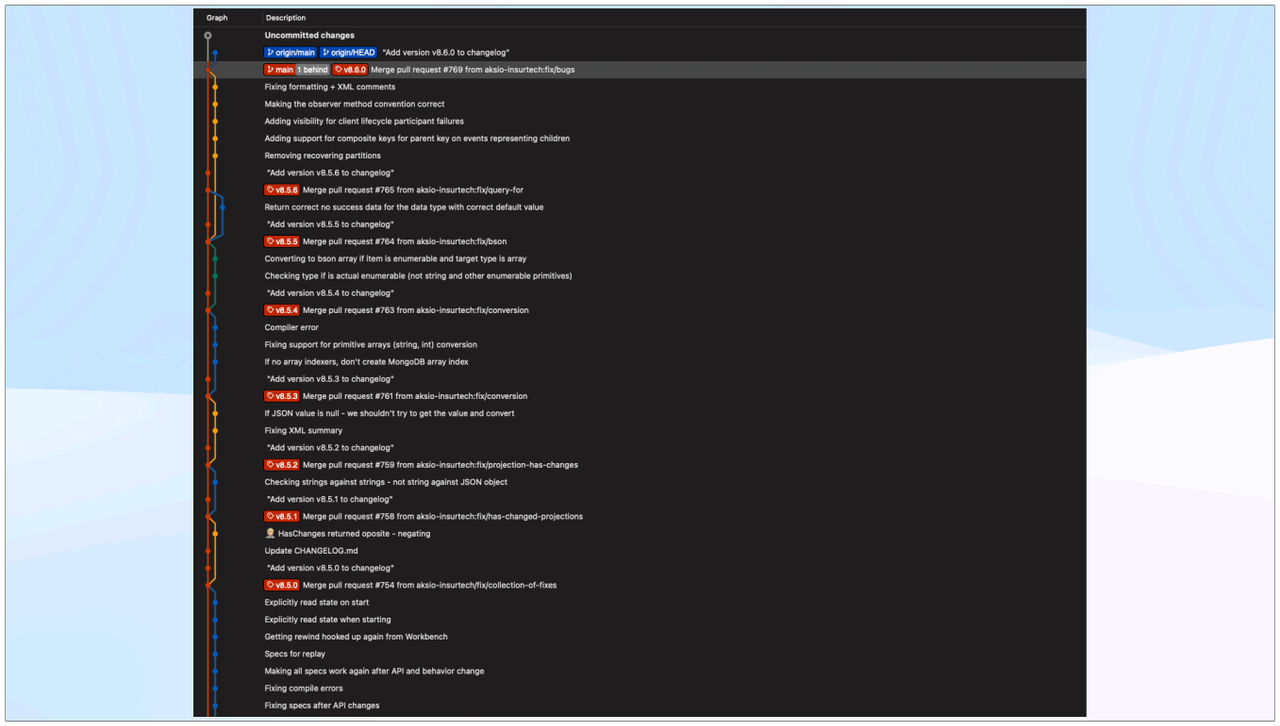

```
import { fromEvent, takeUntil, interval, map, reduce } from 'rxjs';

const clicksInFiveSeconds = fromEvent(document, 'click')
  .pipe(takeUntil(interval(5000)));

const ones = clicksInFiveSeconds.pipe(map(() => 1));
const seed = 0;
const count = ones.pipe(reduce((acc, one) => acc + one, seed));

count.subscribe(x => console.log(x));
```

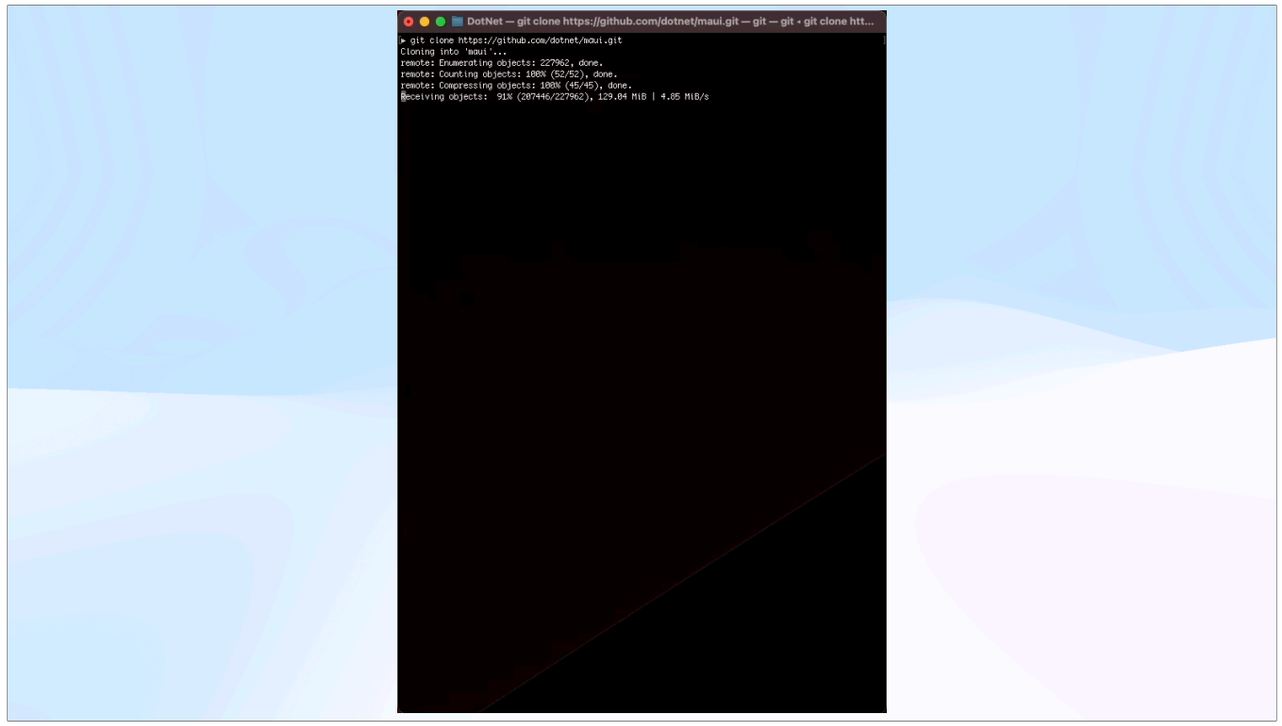

1
2
3
4
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10
WithdrawalPerformed
Id: 123
Amount: $10
BankAccountClosed
Id: 123
Event Store
Observers
Observing events as they occur
BankAccountOpened
DepositPerformed
BankAccountClosed
WithdrawalPerformed
Read Models
Used for read purpose
AccountInfo
AccountBalance

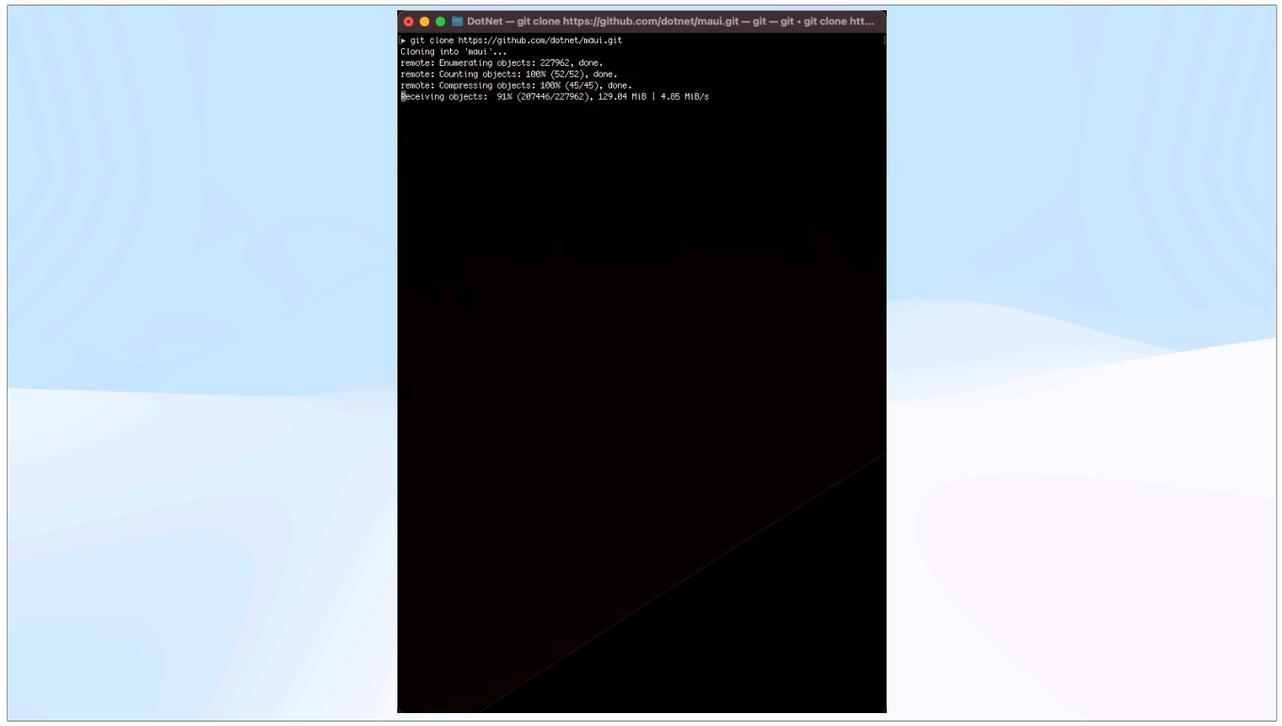

1
2
3
4
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10
WithdrawalPerformed
Id: 123
Amount: $10
BankAccountClosed
Id: 123
Event Store
Observers
Observing events as they occur
BankAccountOpened
DepositPerformed
BankAccountClosed
WithdrawalPerformed
Outbox
AccountBalanceChanged
Read Models
Used for read purpose
AccountInfo
AccountBalance


1
2
3
4
BankAccountOpened
Id: 123
Owner: John Doe
DepositPerformed
Id: 123
Amount: $10
WithdrawalPerformed
Id: 123
Amount: $10
BankAccountClosed
Id: 123
Event Store
Observers
Observing events as they occur
BankAccountOpened
DepositPerformed
BankAccountClosed
WithdrawalPerformed
Outbox
AccountBalanceChanged
Read Models
Used for read purpose
AccountInfo
AccountBalance
Other systems
Inbox per source outbox

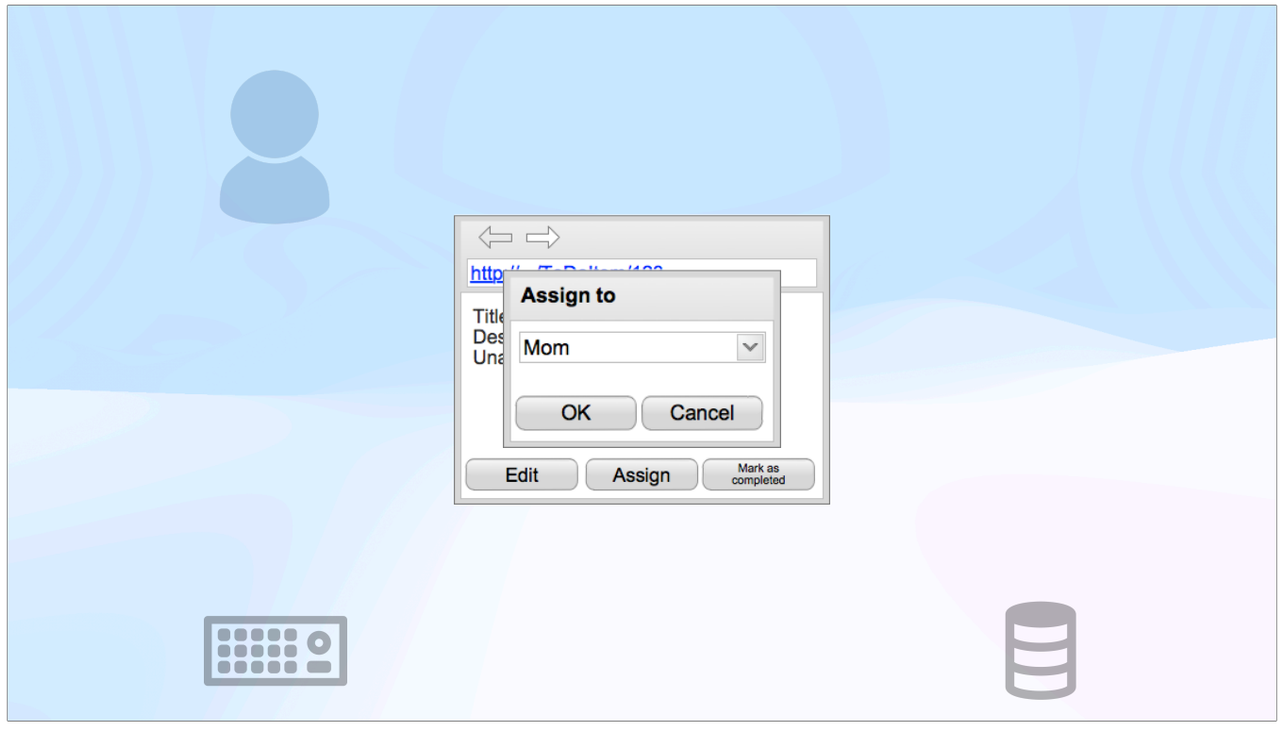

KEEP
CALM
YOU'RE
PROBABLY ALREADY
EVENT SOURCING

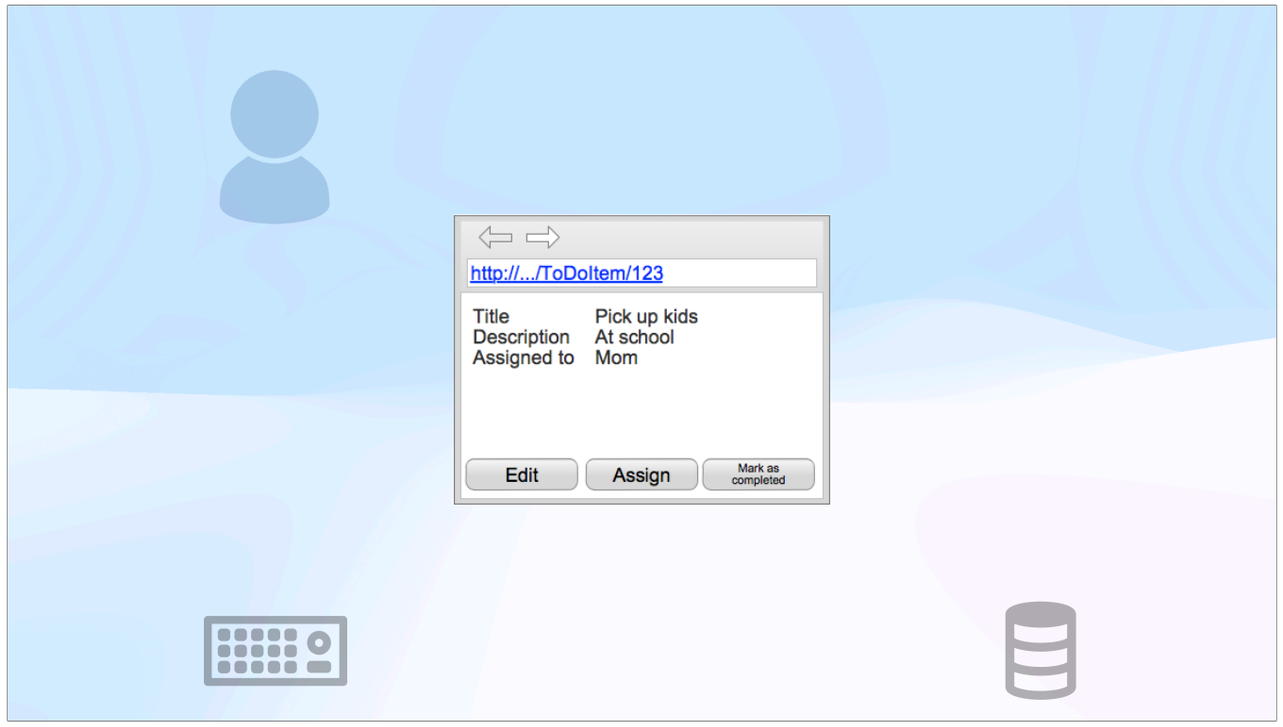

Graph
Description
Uncommitted changes
origin/main
origin/HEAD
"Add version v8.6.0 to changelog"
main
1 behind
v8.6.0
Merge pull request #769 from aksio-insurtech:fix/bugs
Fixing formatting + XML comments
Making the observer method convention correct
Adding visibility for client lifecycle participant failures
Adding support for composite keys for parent key on events representing children
Removing recovering partitions
"Add version v8.5.6 to changelog"
v8.5.6
Merge pull request #765 from aksio-insurtech:fix/query-for
Return correct no success data for the data type with correct default value
"Add version v8.5.5 to changelog"
v8.5.5
Merge pull request #764 from aksio-insurtech:fix/bson
Converting to bson array if item is enumerable and target type is array
Checking type if is actual enumerable (not string and other enumerable primitives)
"Add version v8.5.4 to changelog"
v8.5.4
Merge pull request #763 from aksio-insurtech:fix/conversion
Compiler error
Fixing support for primitive arrays (string, int) conversion
If no array indexers, don't create MongoDB array index
"Add version v8.5.3 to changelog"
v8.5.3
Merge pull request #761 from aksio-insurtech:fix/conversion
If JSON value is null - we shouldn't try to get the value and convert
Fixing XML summary
"Add version v8.5.2 to changelog"
v8.5.2
Merge pull request #759 from aksio-insurtech:fix/projection-has-changes
Checking strings against strings - not string against JSON object
"Add version v8.5.1 to changelog"
v8.5.1
Merge pull request #758 from aksio-insurtech:fix/has-changed-projections
HasChanges returned opposite - negating
Update CHANGELOG.md
"Add version v8.5.0 to changelog"
v8.5.0
Merge pull request #754 from aksio-insurtech/fix/collection-of-fixes
Explicitly read state on start
Explicitly read state when starting
Getting rewind hooked up again from Workbench
Specs for replay
Making all specs work again after API and behavior change
Fixing compile errors
Fixing specs after API changes

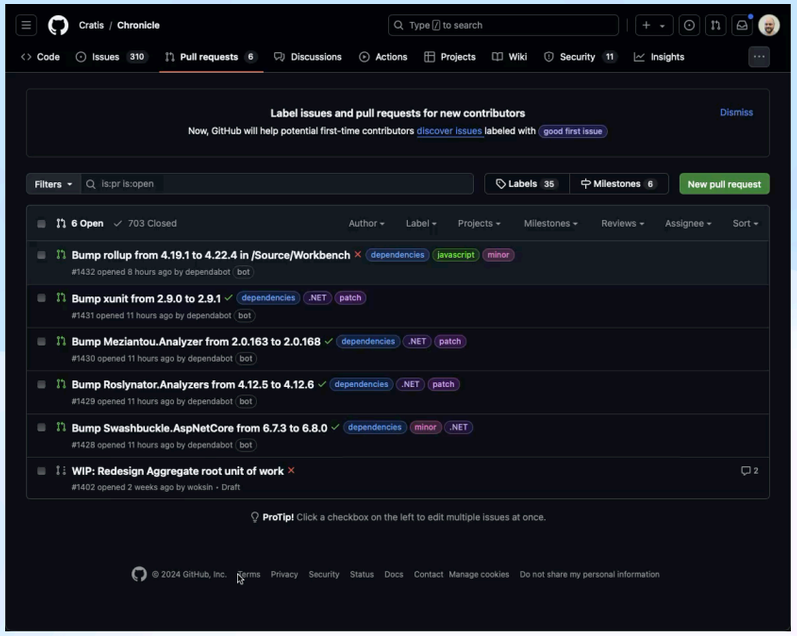

DotNet — git clone https://github.com/dotnet/maui.git — git — git ◂ git clone htt...
► git clone https://github.com/dotnet/maui.git
Cloning into 'maui'...
remote: Enumerating objects: 227962, done.
remote: Counting objects: 100% (32/62), done.
remote: Compressing objects: 100% (45/45), done.
Receiving objects: 91% (207446/227962), 129.04 MiB | 4.85 MiB/s

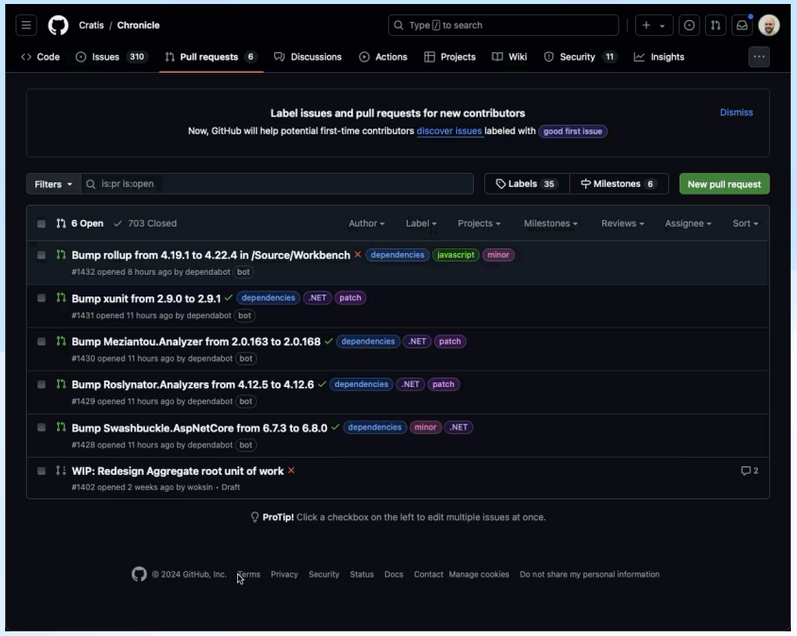

DotNet — git clone https://github.com/dotnet/maui.git — git — git ◂ git clone htt...
► git clone https://github.com/dotnet/maui.git
Cloning into 'maui'...
remote: Enumerating objects: 227962, done.
remote: Counting objects: 100% (32/62), done.
remote: Compressing objects: 100% (45/45), done.
Receiving objects: 91% (207446/227962), 129.04 MiB | 4.85 MiB/s

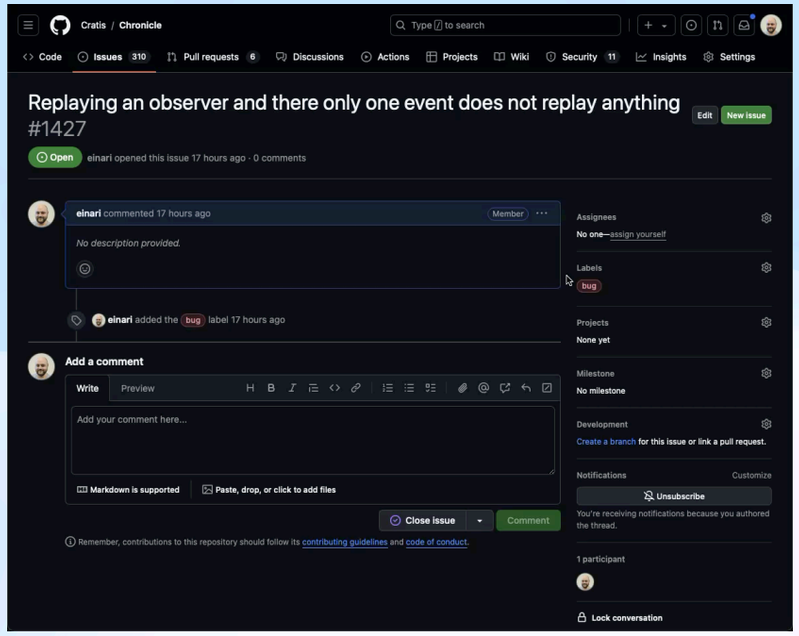

http://.../ToDoItem/123
Title
Description
Unassigned
Pick up kids
At school
Edit
Assign
Mark as completed

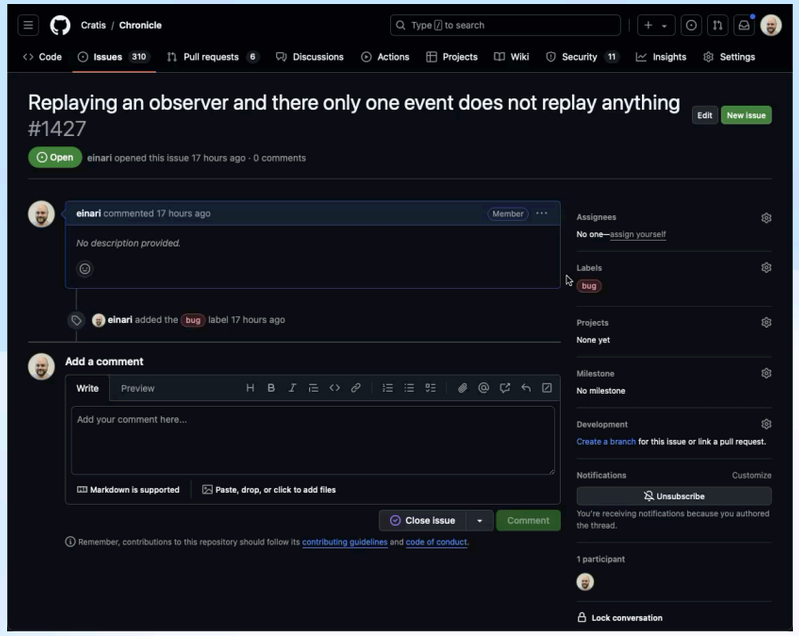

Assign to
Mom
OK
Cancel
Edit
Assign
Mark as completed


http://.../ToDoItem/123
Title
Description
Assigned to
Pick up kids
At school
Mom
Edit
Assign
Mark as completed


Cratis / Chronicle
Type / to search
Code
Issues 310
Pull requests 6
Discussions
Actions
Projects
Wiki
Security 11
Insights
Label issues and pull requests for new contributors
Now, GitHub will help potential first-time contributors discover issues labeled with good first issue
Dismiss
Filters
is:pr is:open
Labels 35
Milestones 6
New pull request
6 Open
703 Closed
Author
Label
Projects
Milestones
Reviews
Assignee
Sort
Bump rollup from 4.19.1 to 4.22.4 in /Source/Workbench
dependencies javascript minor
#1432 opened 8 hours ago by dependabot bot
Bump xunit from 2.9.0 to 2.9.1
dependencies .NET patch
#1431 opened 11 hours ago by dependabot bot
Bump Meziantou.Analyzer from 2.0.163 to 2.0.168
dependencies .NET patch
#1430 opened 11 hours ago by dependabot bot
Bump Roslynator.Analyzers from 4.12.5 to 4.12.6
dependencies .NET patch
#1429 opened 11 hours ago by dependabot bot
Bump Swashbuckle.AspNetCore from 6.7.3 to 6.8.0
dependencies minor .NET
#1428 opened 11 hours ago by dependabot bot
WIP: Redesign Aggregate root unit of work
2
#1402 opened 2 weeks ago by woksin • Draft
ProTip! Click a checkbox on the left to edit multiple issues at once.
© 2024 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information


Cratis / Chronicle
Type / to search
Code
Issues 310
Pull requests 6
Discussions
Actions
Projects
Wiki
Security 11
Insights
Label issues and pull requests for new contributors
Now, GitHub will help potential first-time contributors discover issues labeled with good first issue
Dismiss
Filters
is:pr is:open
Labels 35
Milestones 6
New pull request
6 Open
703 Closed
Author
Label
Projects
Milestones
Reviews
Assignee
Sort
Bump rollup from 4.19.1 to 4.22.4 in /Source/Workbench
dependencies javascript minor
#1432 opened 8 hours ago by dependabot bot
Bump xunit from 2.9.0 to 2.9.1
dependencies .NET patch
#1431 opened 11 hours ago by dependabot bot
Bump Meziantou.Analyzer from 2.0.163 to 2.0.168
dependencies .NET patch
#1430 opened 11 hours ago by dependabot bot
Bump Roslynator.Analyzers from 4.12.5 to 4.12.6
dependencies .NET patch
#1429 opened 11 hours ago by dependabot bot
Bump Swashbuckle.AspNetCore from 6.7.3 to 6.8.0
dependencies minor .NET
#1428 opened 11 hours ago by dependabot bot
WIP: Redesign Aggregate root unit of work
2
#1402 opened 2 weeks ago by woksin • Draft
ProTip! Click a checkbox on the left to edit multiple issues at once.
© 2024 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information


Cratis / Chronicle
Type / to search
Code
Issues 310
Pull requests 6
Discussions
Actions
Projects
Wiki
Security 11
Insights
Settings
Replaying an observer and there only one event does not replay anything #1427
Edit
New issue
Open
einari opened this issue 17 hours ago · 0 comments
einari commented 17 hours ago
Member
No description provided.
einari added the bug label 17 hours ago
Add a comment
Write
Preview
Add your comment here...
Markdown is supported
Paste, drop, or click to add files
Close issue
Comment
Remember, contributions to this repository should follow its contributing guidelines and code of conduct.
Assignees
No one—assign yourself
Labels
bug
Projects
None yet
Milestone
No milestone
Development
Create a branch for this issue or link a pull request.
Notifications
Customize
Unsubscribe
You're receiving notifications because you authored the thread.
1 participant
Lock conversation


Cratis / Chronicle
Type / to search
Code
Issues 310
Pull requests 6
Discussions
Actions
Projects
Wiki
Security 11
Insights
Settings
Replaying an observer and there only one event does not replay anything #1427
Edit
New issue
Open
einari opened this issue 17 hours ago · 0 comments
einari commented 17 hours ago
Member
No description provided.
einari added the bug label 17 hours ago
Add a comment
Write
Preview
Add your comment here...
Markdown is supported
Paste, drop, or click to add files
Close issue
Comment
Remember, contributions to this repository should follow its contributing guidelines and code of conduct.
Assignees
No one—assign yourself
Labels
bug
Projects
None yet
Milestone
No milestone
Development
Create a branch for this issue or link a pull request.
Notifications
Customize
Unsubscribe
You're receiving notifications because you authored the thread.
1 participant
Lock conversation

Benefits of event sourcing

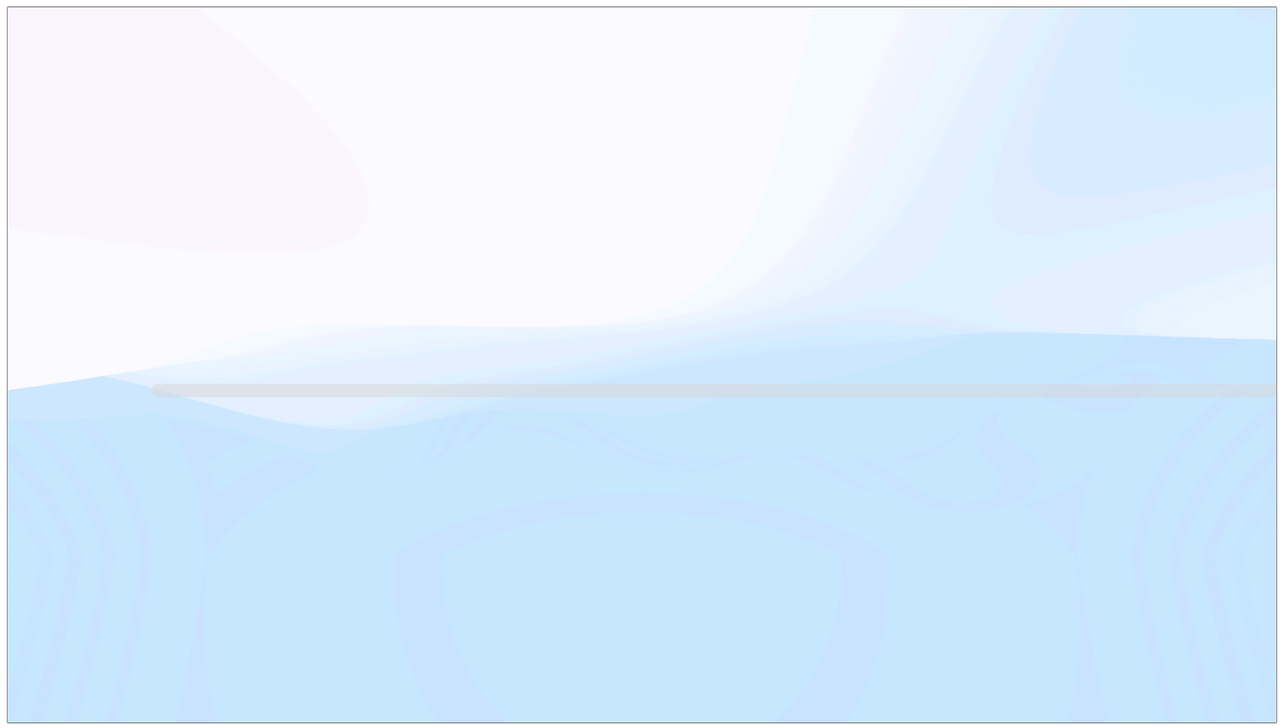
- We can replay our system to any point in time - great for debugging problems
- State changes can be utilized for any purpose - in a reactive manner
- Easier to reason about state changes in the system with domain experts than “data”
- Perfect audit trail, who / what, when, why did something happen

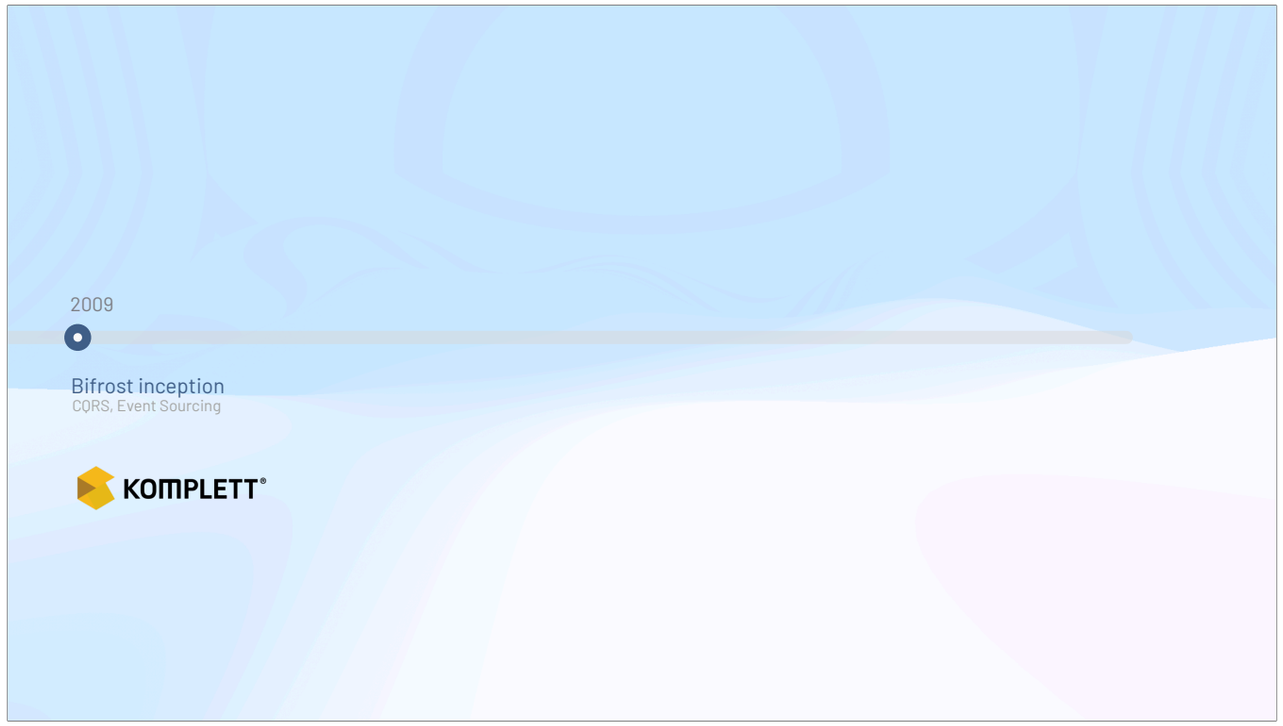
Polyglot persistence

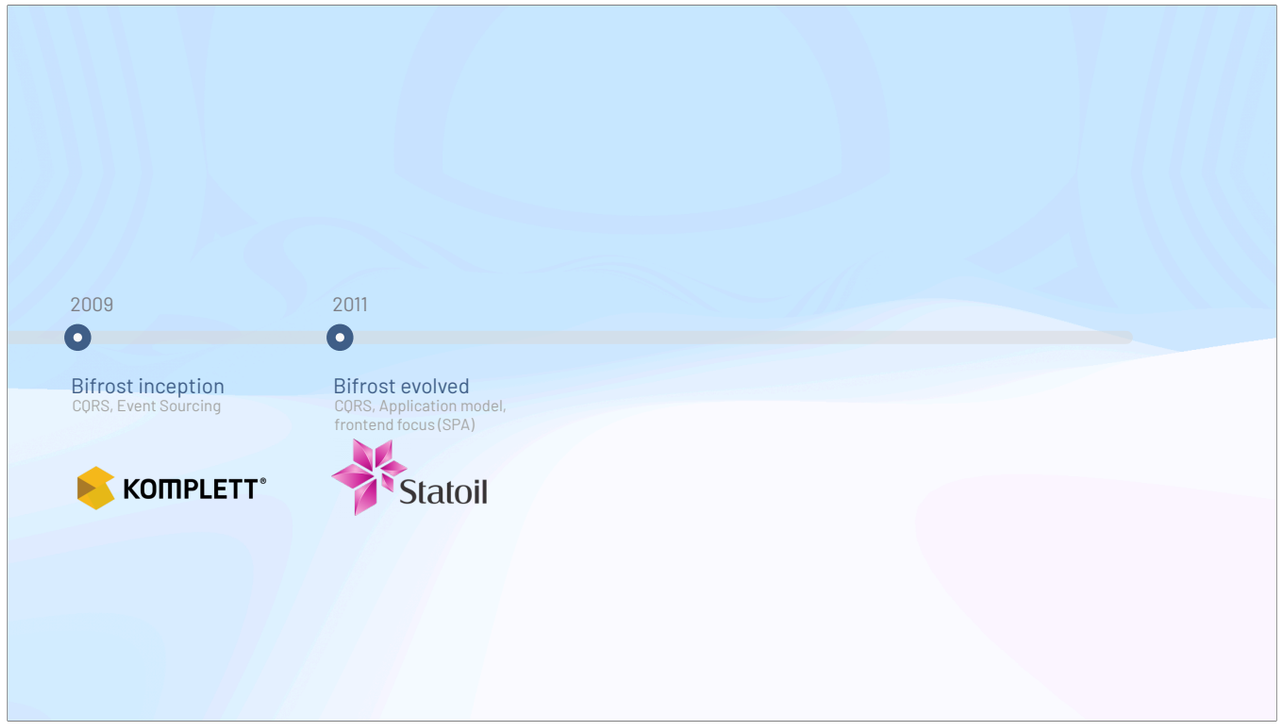
- The truth in the system is in the event store. Current state are projections of this.
- Optimize for each use-case
- Use the right technology for the job
- Replay is your friend

02

Cratis




2009
Bifrost inception
CQRS, Event Sourcing
KOMPLETT®


2009
2011
Bifrost inception
CQRS, Event Sourcing
Bifrost evolved
CQRS, Application model,
frontend focus (SPA)
KOMPLETT®
Statoil


2009
Bifrost inception
CQRS, Event Sourcing
KOMPLETT®
2011
Bifrost evolved
CQRS, Application model, frontend focus (SPA)
Statoil
2017
Bifrost repurposed
CQRS, AppModel, Event Sourcing
DOLITTLE


2009
Bifrost inception
CQRS, Event Sourcing
KOMPLETT®
2011
Bifrost evolved
CQRS, Application model, frontend focus (SPA)
Statoil
2017
Bifrost repurposed
CQRS, AppModel, Event Sourcing
DOLITTLE
2021
Cratis inception
Event Sourcing double down (Some CQRS, AppModel++)
Aksio

Cratis

Chronicle

Developer centric event sourcing database and client libraries

Application Model

Productivity, opinionated, CQRS oriented, bridging frontend and backend

Fundamentals

Generic building blocks, helpers

Cratis

Chronicle

Developer centric event sourcing database and client libraries

Application Model

Productivity, opinionated, CQRS oriented, bridging frontend and backend

Fundamentals

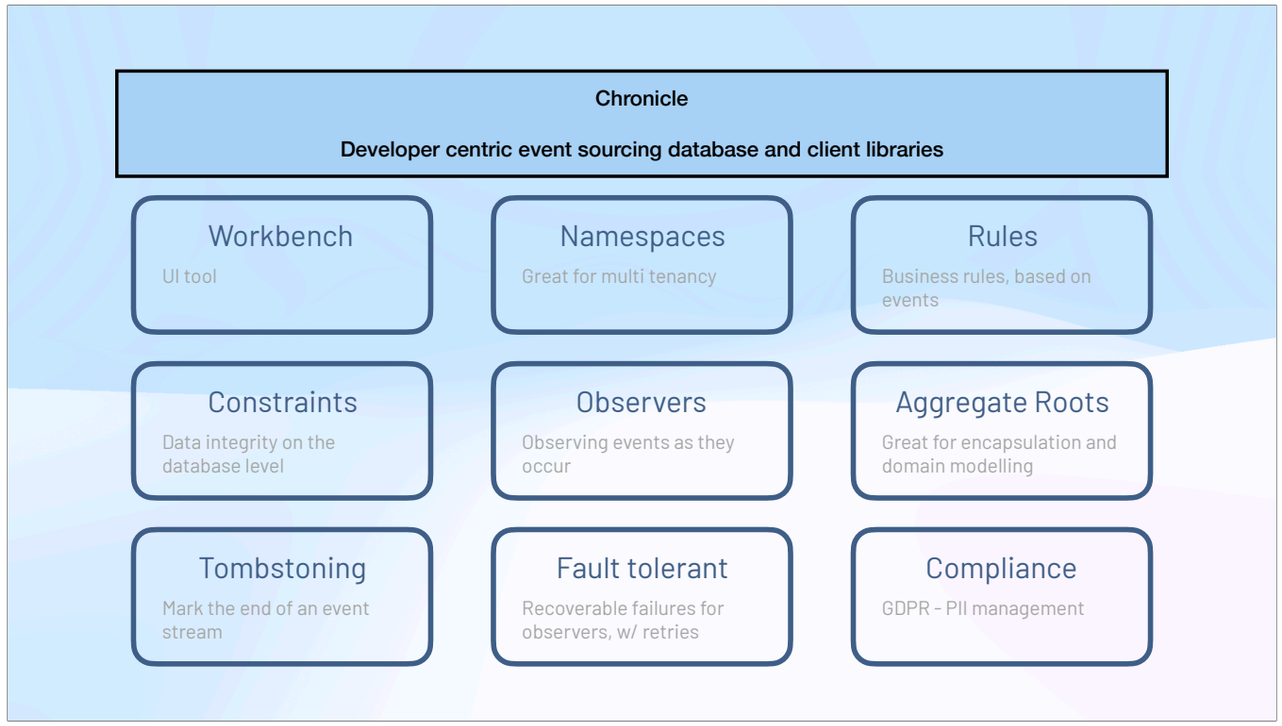
Generic building blocks, helpers

Chronicle

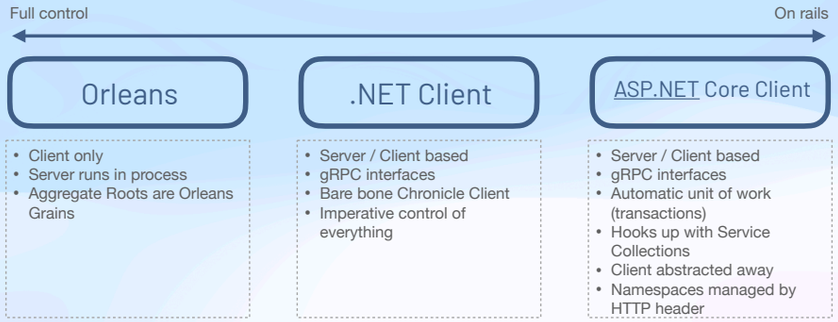
Developer centric event sourcing database and client libraries

Chronicle

Developer centric event sourcing database and client libraries


Chronicle
Developer centric event sourcing database and client libraries
Workbench
UI tool
Namespaces
Great for multi tenancy
Rules
Business rules, based on events
Constraints
Data integrity on the database level
Observers
Observing events as they occur
Aggregate Roots
Great for encapsulation and domain modelling
Tombstoning
Mark the end of an event stream
Fault tolerant
Recoverable failures for observers, w/ retries
Compliance
GDPR - PII management

Usage

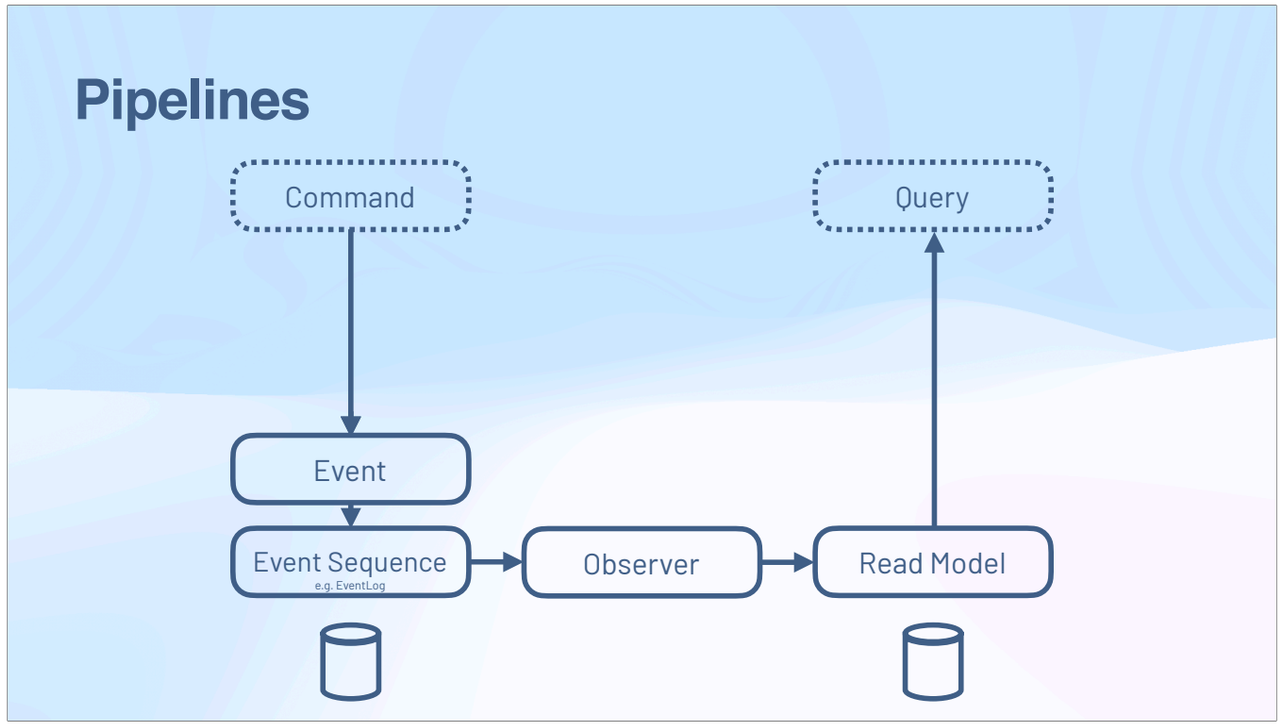

Full control
On rails
Orleans
.NET Client
ASP.NET Core Client
• Client only
• Server runs in process
• Aggregate Roots are Orleans Grains
• Server / Client based
• gRPC interfaces
• Bare bone Chronicle Client
• Imperative control of everything
• Server / Client based
• gRPC interfaces
• Automatic unit of work (transactions)
• Hooks up with Service Collections
• Client abstracted away
• Namespaces managed by HTTP header

Observer types

- "If-this-then-that" scenarios
- Unmanaged idempotency
- Might not be replayable
- Can produce more events
- Candidate for machine learning

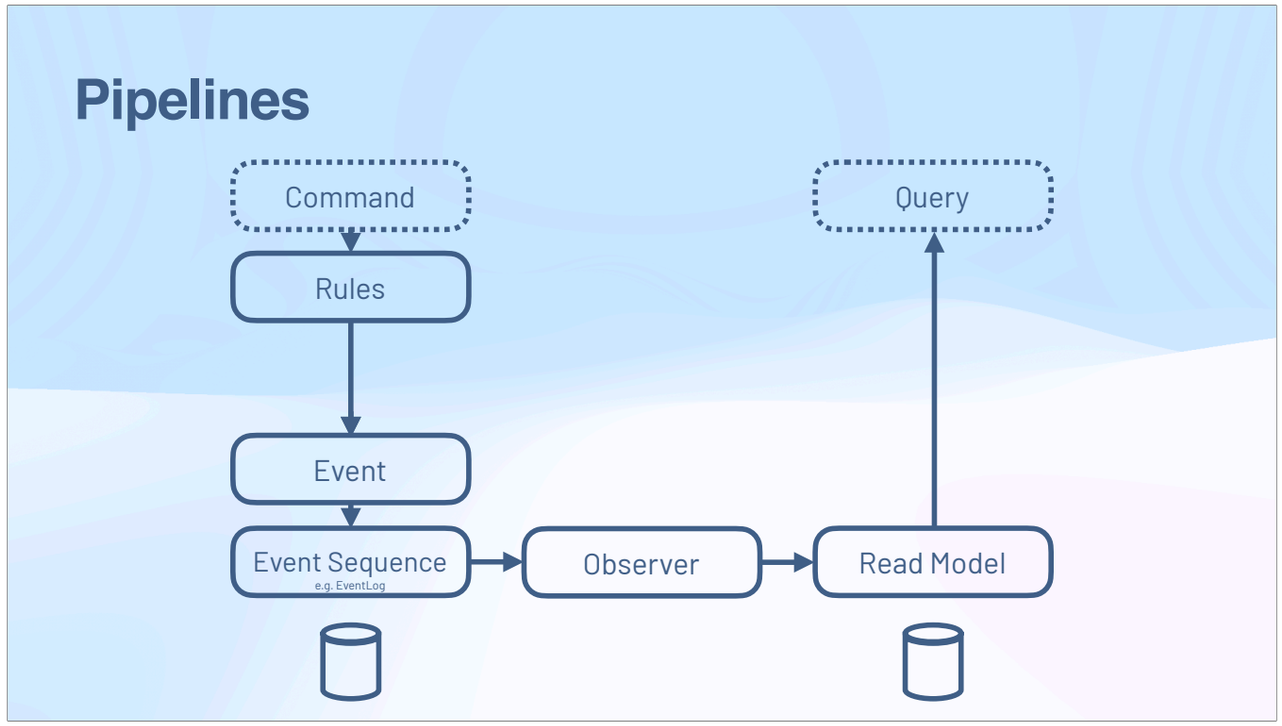
- Maps events and reduces to read models
- Managed read model state
- Can produce more optimal database operations
- Idempotency is easier to guarantee
- Replayable by nature

- Declarative model defining map / reduce
- Produces read models
- Managed read model state
- Guaranteed idempotency
- Complex relationships can be easier
- Replayable by nature

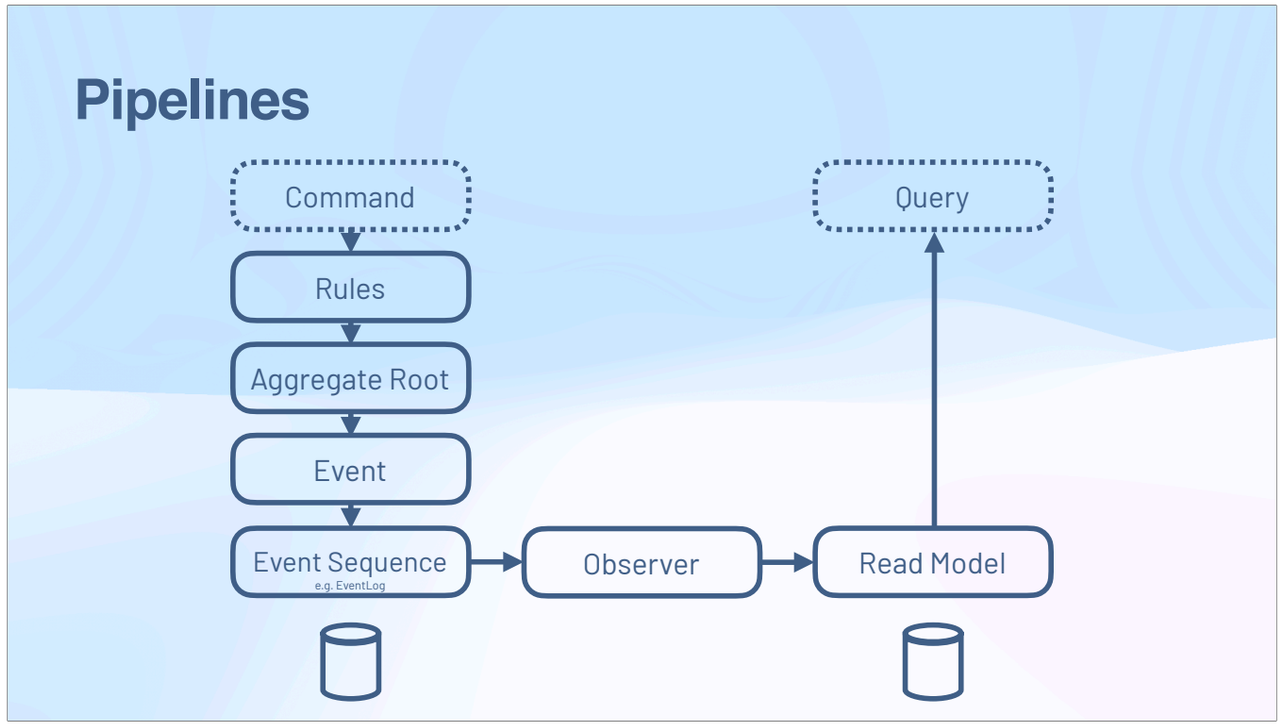


Talk about the different consistency models; strong consistency on the write side and eventual consistency on the read side.

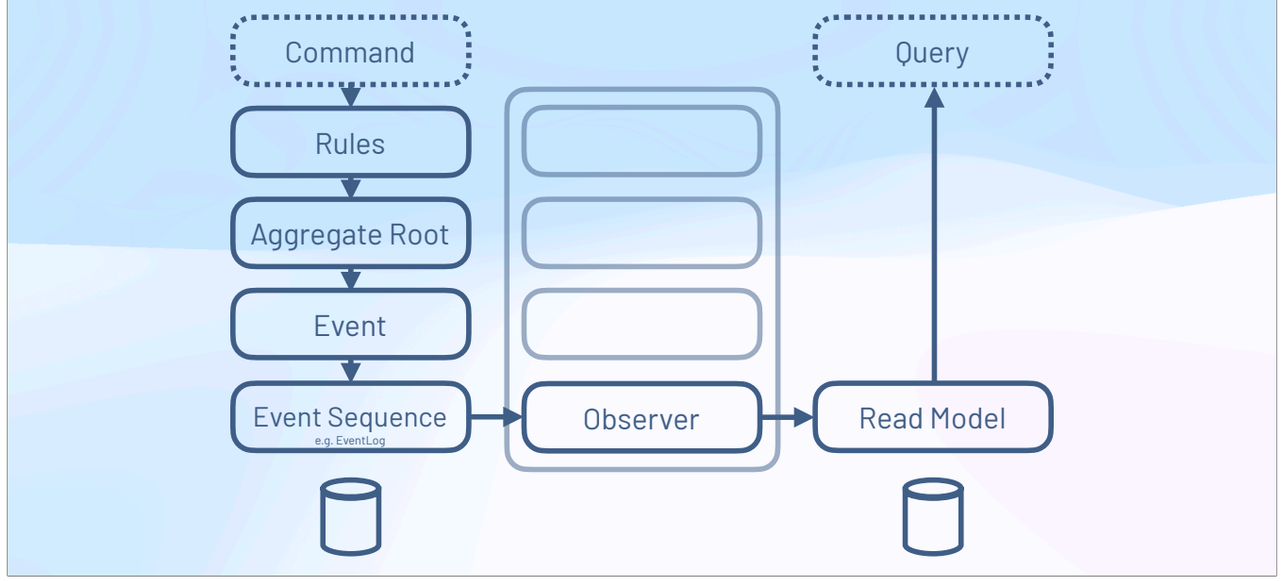
Pipelines

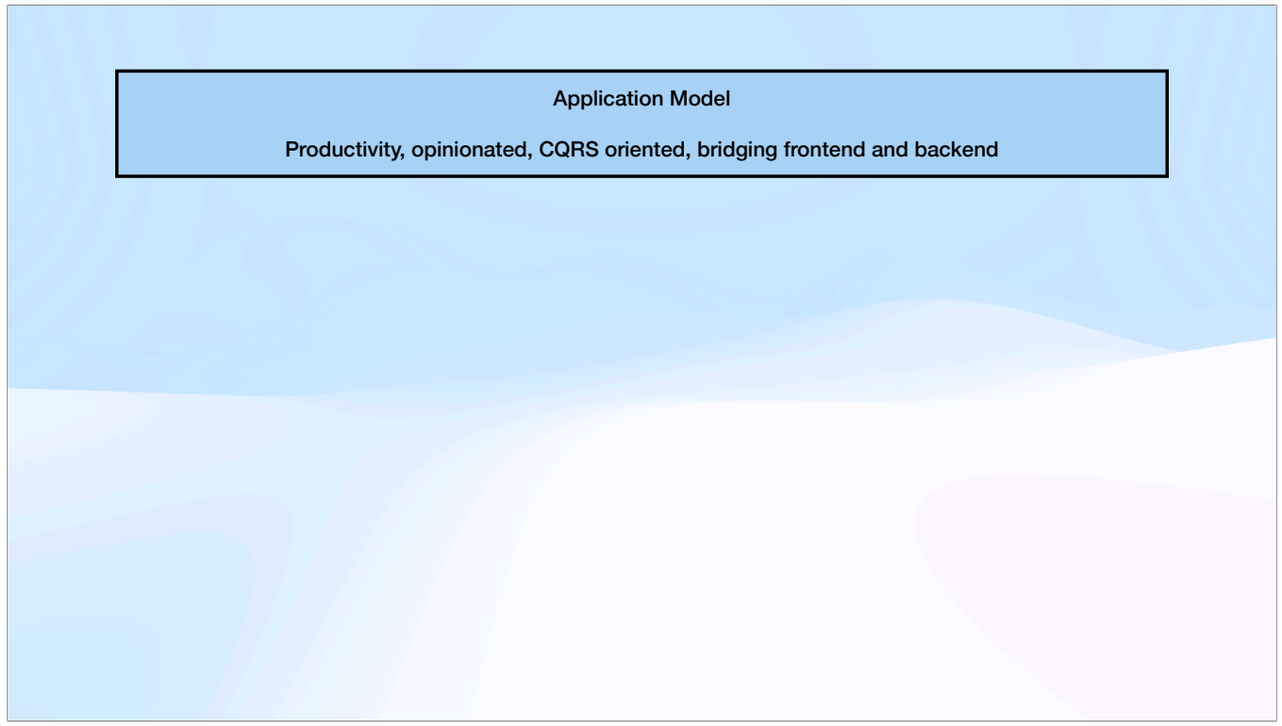

Command
Rules
Event
Event Sequence
e.g. EventLog
Observer
Query
Read Model

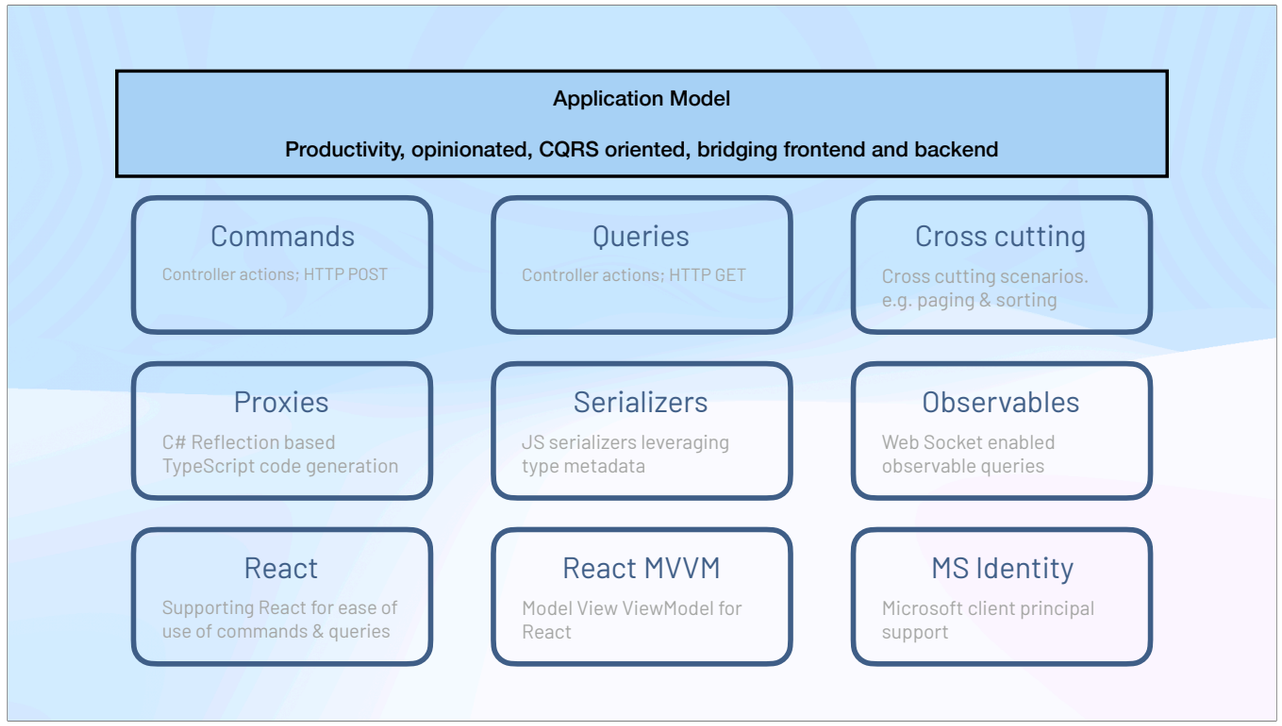
Pipelines


Command
Rules
Aggregate Root
Event
Event Sequence
e.g. EventLog
Observer
Read Model
Query

Pipelines


Command
Rules
Aggregate Root
Event
Event Sequence
e.g. EventLog
Observer
Query
Read Model


Application Model
Productivity, opinionated, CQRS oriented, bridging frontend and backend


Application Model
Productivity, opinionated, CQRS oriented, bridging frontend and backend
Commands
Controller actions; HTTP POST
Queries
Controller actions; HTTP GET
Cross cutting
Cross cutting scenarios.
e.g. paging & sorting
Proxies
C# Reflection based
TypeScript code generation
Serializers
JS serializers leveraging
type metadata
Observables
Web Socket enabled
observable queries
React
Supporting React for ease of
use of commands & queries
React MVVM
Model View ViewModel for
React
MS Identity
Microsoft client principal
support

React MVVM

```tsx
export const Observers = withViewModel(ObserversViewModel, ({ viewModel }) => {
    const params = useParams<EventStoreAndNamespaceParams>();
    const queryArgs: AllObserversArguments = {
        eventStore: params.eventStore!,
        namespace: viewModel.currentNamespace.name
    };

    return (
        <DataPage
            title={strings.eventStore.namespaces.observers.title}
            query={AllObservers}
            queryArguments={queryArgs}
            emptyMessage={strings.eventStore.namespaces.observers.empty}
            defaultFilters={defaultFilters}
            globalFilterFields={['runningState']}
            dataKey='observerId'
            onSelectionChange={(e) => (viewModel.selectedObserver = e.value as ObserverInformation)}>
            <DataPage.MenuItems>
                <MenuItem
                    id="replay"
                    label={strings.eventStore.namespaces.observers.actions.replay} icon={faIcons.FaArrowsRotate}
                    disableOnUnselected
                    command={() => viewModel.replay()} />
            </DataPage.MenuItems>
            <DataPage.Columns>
                <Column field='observerId' header={strings.eventStore.namespaces.observers.columns.id} sortable />
                <Column
                    field='type'
                    header={strings.eventStore.namespaces.observers.columns.observerType}
                    sortable
                    body={observerType} />

                <Column
                    field='nextEventSequenceNumber'
                    dataType='numeric'
                    header={strings.eventStore.namespaces.observers.columns.nextEventSequenceNumber}
                    sortable />

                <Column
                    field='runningState'
                    dataType='numeric'
                    header={strings.eventStore.namespaces.observers.columns.state}
                    sortable
                    body={runningState}/>
```

```tsx
You, 6 days ago | 1 author (You)
@injectable()
export class ObserversViewModel {

    constructor(
        namespaces: INamespaces,
        private readonly _replay: Replay,
        private readonly _dialogs: IDialogs,
        @inject('params') private readonly _params: EventStoreAndNamespaceParams) {
        this.currentNamespace = { id: Guid.empty, name: '', description: '' };

        namespaces.currentNamespace.subscribe(namespace => {
            this.currentNamespace = namespace;       You, 3 weeks ago • Introducing common DataTable for queries and ob…
        });
    }

    currentNamespace: Namespace;
    selectedObserver: ObserverInformation | undefined;

    async replay() {
        if (this.selectedObserver) {
            const observerId = this.selectedObserver.observerId;
            const result = await this._dialogs.showConfirmation('Replay?', `Are you sure you want to replay ${observerId}?`, DialogButtons.YesNo);
            if (result == DialogResult.Yes) {
                this._replay.eventStore = this._params.eventStore!;
                this._replay.namespace = this.currentNamespace.name;
                this._replay.observerId = observerId;
                const commandResult = await this._replay.execute();
                commandResult
                    .onException((error) => {
                        this._dialogs.showConfirmation('Replay', `Replay ${observerId} failed: ${error}`, DialogButtons.Ok);
                    });
            }
        }
    }
}
```

```tsx
describe('when replaying and there is a selected observer', given(a_view_model, (context) => {
    beforeEach(() => {
        context.viewModel.selectedObserver = new ObserverInformation();
        context.dialogs.showConfirmation.resolves(DialogResult.Yes);
        context.viewModel.replay();
    });

    it('should display dialog', () => context.dialogs.showConfirmation.should.be.called);
    it('should perform replay', () => context.replay.execute.should.be.called);
}));
```

ProxyGeneration

Queries

```
/// <summary>
/// Get and observe all observers for an event store and namespace.
/// </summary>
/// <param name="eventStore">The event store the observers are for.</param>
/// <param name="namespace">The namespace within the event store the observers are for.</param>
/// <returns>An observable of a collection of <see cref="ObserverInformation"/>.</returns>
[HttpGet("all-observers/observe")]
0 references
public ISubject<IEnumerable<ObserverInformation>> AllObservers(
    [FromRoute] EventStoreName eventStore,
    [FromRoute] EventStoreNamespaceName @namespace)
{
    var namespaceStorage = storage.GetEventStore(eventStore).GetNamespace(@namespace
    return namespaceStorage.Observers.ObserveAll();
}
```

```
export class AllObservers extends ObservableQueryFor<ObserverInformation[], AllObserversArguments> {
    readonly route: string = '/api/event-store/{eventStore}/{namespace}/observers/all-observers/observe';
    readonly routeTemplate: Handlebars.TemplateDelegate = routeTemplate;
    readonly defaultValue: ObserverInformation[] = [];
    private readonly _sortBy: AllObserversSortBy;
    private static readonly _sortBy: AllObserversSortByWithoutQuery = new AllObserversSortByWithoutQuery();

    constructor() {
        super(ObserverInformation, true);
        this._sortBy = new AllObserversSortBy(this);
    }

    get requiredRequestArguments(): string[] {
        return [
            'eventStore',
            'namespace',
        ];
    }

    get sortBy(): AllObserversSortBy {
        return this._sortBy;
    }

    static get sortBy(): AllObserversSortByWithoutQuery {
        return this._sortBy;
    }

    static use(args?: AllObserversArguments, sorting?: Sorting): [QueryResultWithState<ObserverInformation[]>, SetSorting] {
        return useObservableQuery<ObserverInformation[], AllObservers, AllObserversArguments>(AllObservers, args, sorting);
    }

    static useWithPaging(pageSize: number, args?: AllObserversArguments, sorting?: Sorting): [QueryResultWithState<ObserverInformation[]>, SetSorting, SetPage, SetPageSize] {
        return useObservableQueryWithPaging<ObserverInformation[], AllObservers>(AllObservers, new Paging(0, pageSize), args, sorting);
    }
}
```

ProxyGeneration

Commands

```
/// <summary>
/// Rewind a specific observer in an event store and specific namespace.
/// </summary>
/// <param name="eventStore">Name of the event store the observer is for.</param>
/// <param name="namespace">Namespace within the event store the observer is for.</param>
/// <param name="observerId">Identifier of the observer to rewind.</param>
/// <returns>Awaitable task.</returns>
[HttpPost("{namespace}/replay/{observerId}")]
0 references
public async Task Replay(
    [FromRoute] EventStoreName eventStore,
    [FromRoute] EventStoreNamespaceName @namespace,
    [FromRoute] ObserverId observerId)
{
    await grainFactory.GetGrain<IObserver>(new ObserverKey(observerId, eventStore, @namespace, EventSequenceId.Log)).Replay();
}
```

```
export class Replay extends Command<IReplay> implements IReplay {
    readonly route: string = '/api/event-store/{eventStore}/observers/{namespace}/replay/{observerId}';
    readonly routeTemplate: Handlebars.TemplateDelegate = routeTemplate;
    readonly validation: CommandValidator = new ReplayValidator();

    private _eventStore!: string;
    private _namespace!: string;
    private _observerId!: string;

    constructor() {
        super(Object, false);
    }

    get requestArguments(): string[] {
        return [
            'eventStore',
            'namespace',
            'observerId',
        ];
    }

    get properties(): string[] {
        return [
            'eventStore',
            'namespace',
            'observerId',
        ];
    }

    get eventStore(): string {
        return this._eventStore;
    }

    set eventStore(value: string) {
        this._eventStore = value;
        this.propertyChanged('eventStore');
    }
    get namespace(): string {
        return this._namespace;
    }

    set namespace(value: string) {
        this._namespace = value;
        this.propertyChanged('namespace');
    }
    get observerId(): string {
        return this._observerId;
    }

    set observerId(value: string) {
        this._observerId = value;
        this.propertyChanged('observerId');
    }

    static use(initialValues?: IReplay): [Replay, SetCommandValues<IReplay>, ClearCommandValues] {
        return useCommand<Replay, IReplay>(Replay, initialValues);
    }
}
```


BACK
TO THE FUTURE

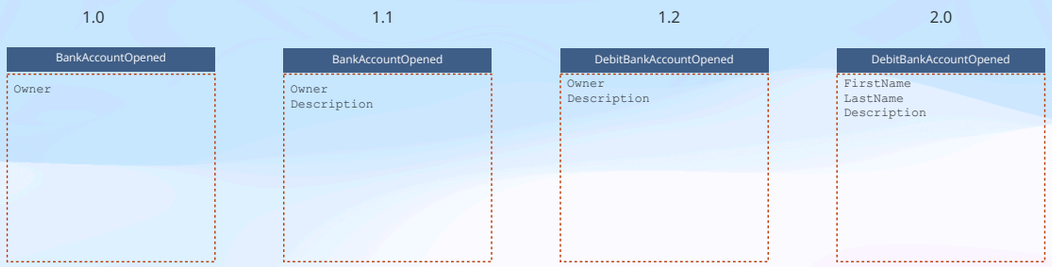
Concurrency

- Be able to not allow something based on concurrency rules
  - e.g. expected event sequence number (Aggregate Root)
- Extensible
- Governed by the event sequence - concurrency scope part of *Append* methods

Event evolution

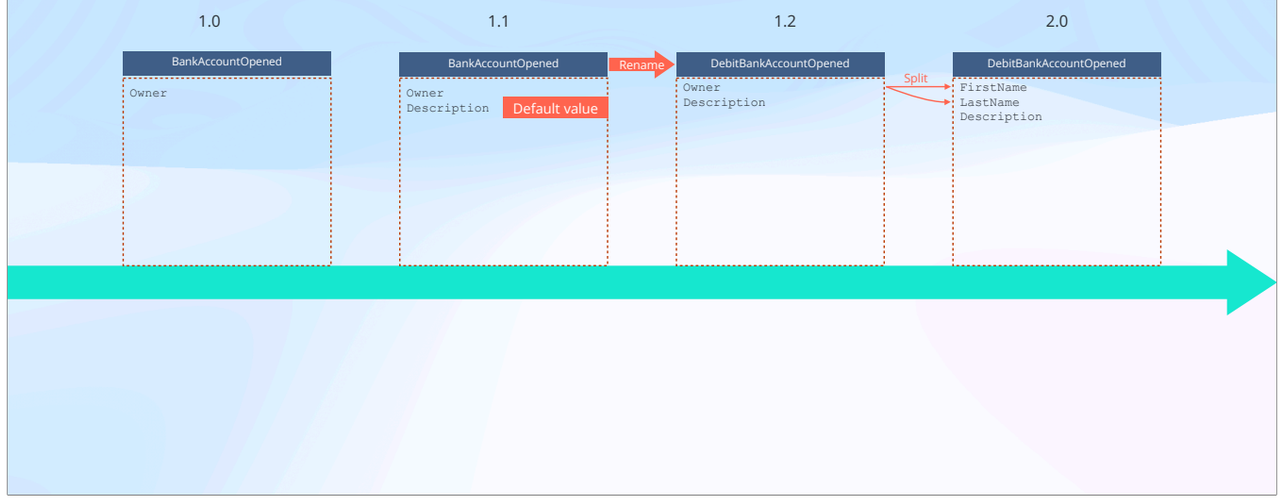
- Systems evolve, so do events
- Event types come and go
- New meaning to events
- Properties are added or removed
- Needs to support replaying your own system
- You might want to be able to roll back systems
- Needs to be able to run without code (e.g. JMESPath)
  - Possibility: C# declarative definition

Event evolution



Event evolution

Upcasting



Event evolution

Downcasting

And then some…

- Projections
  - Attribute based projections on models (similar to EntityFramework, Dapper)
- Sinks
  - Relational databases (SQL, PostgreSQL)
  - Event Sequences
- Event Sequences
  - Retention policies
- Workbench
  - Query editor for sequences

GitHelp

Expert help at your fingertips

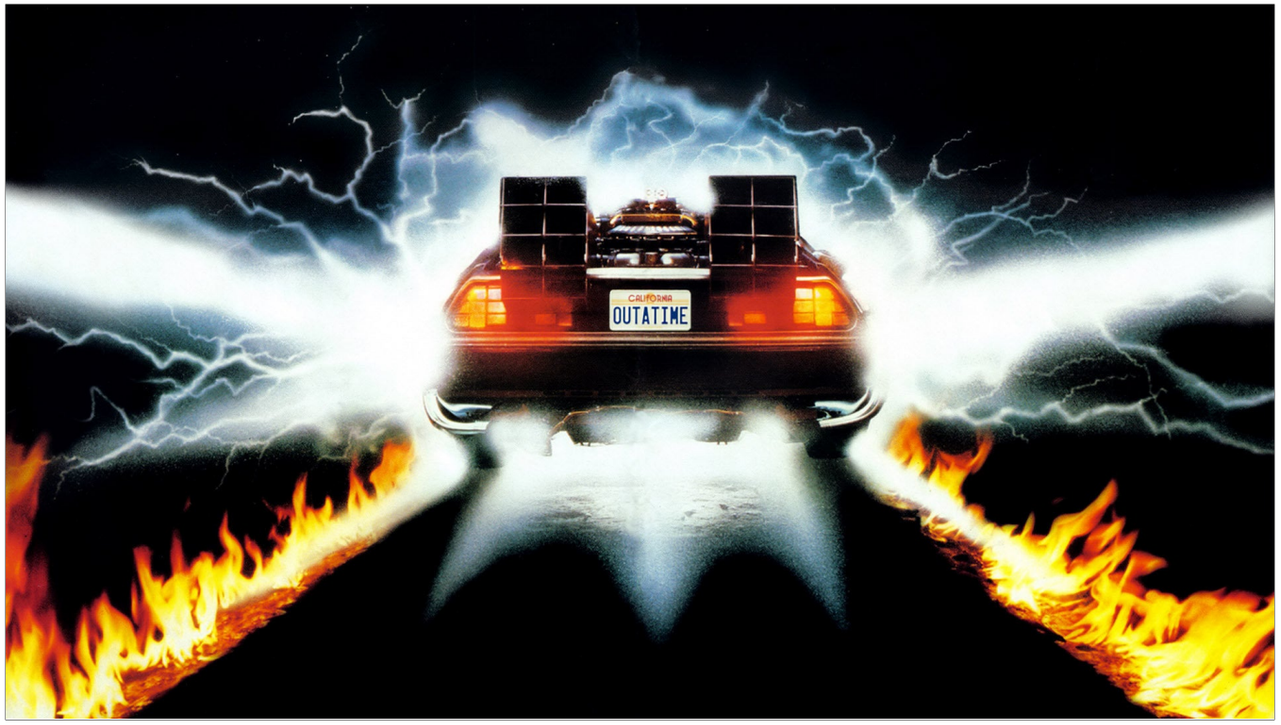
GitHelp

Expert help at your fingertips


packt
1ST EDITION
Metaprogramming in C#
Automate your .NET development
and simplify overcomplicated code
EINAR INGEBRIGTSEN

“Event sourcing made **easy** - no complexity, just **powerful tools** and **flexibility** for everyone, from beginners to pros.”

https://cratis.io
https://github.com/cratis/chronicle


CALIFORNIA
OUTATIME