# How to fall in love with automated tests!😍

Andrew Poole

Senior Backend Engineer @ FLAGSTONE

# How to fall in love with automated tests!😍

Andrew Poole

Senior Backend Engineer @ FLAGSTONE

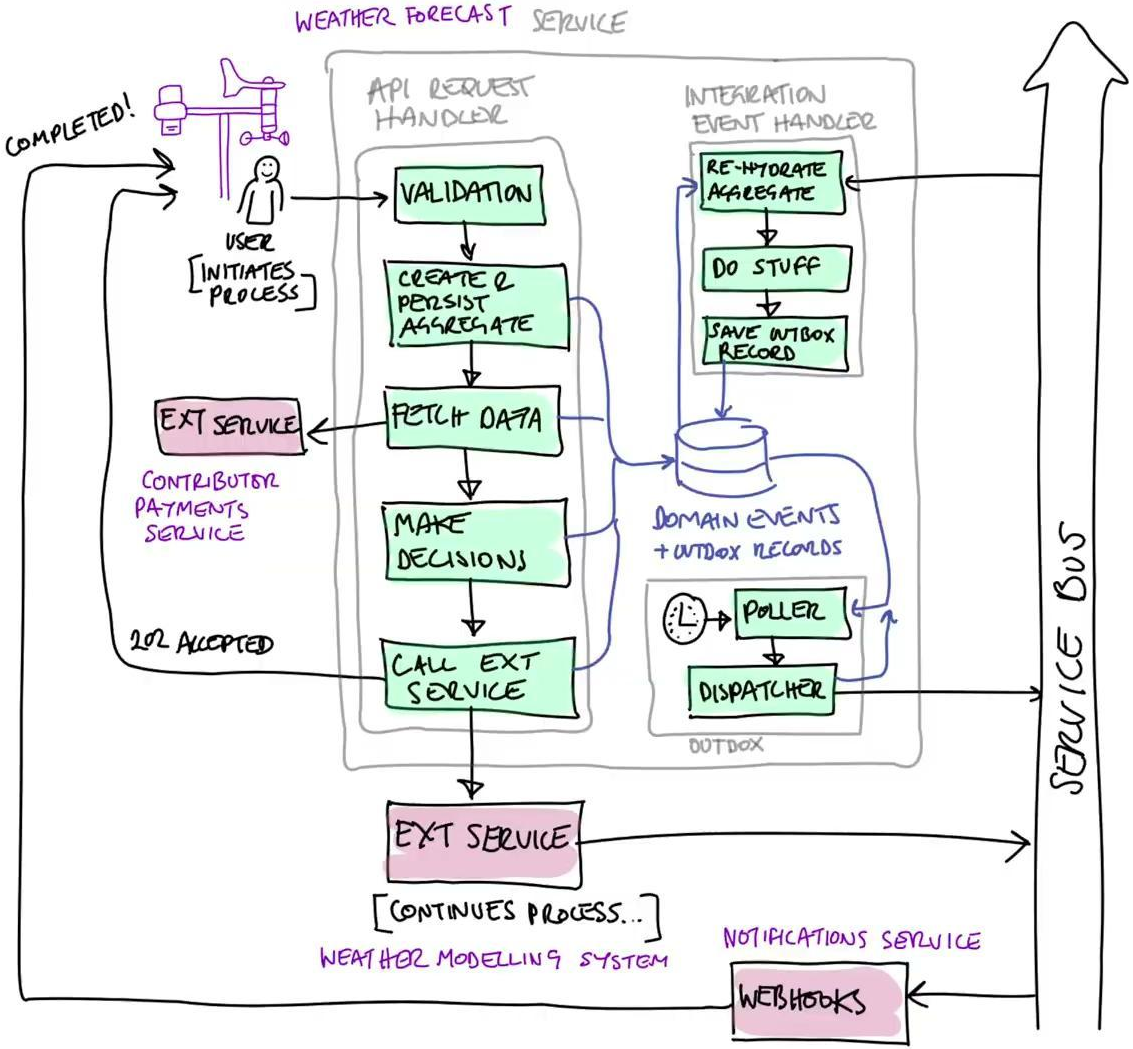
# EndToEnd Component Tests

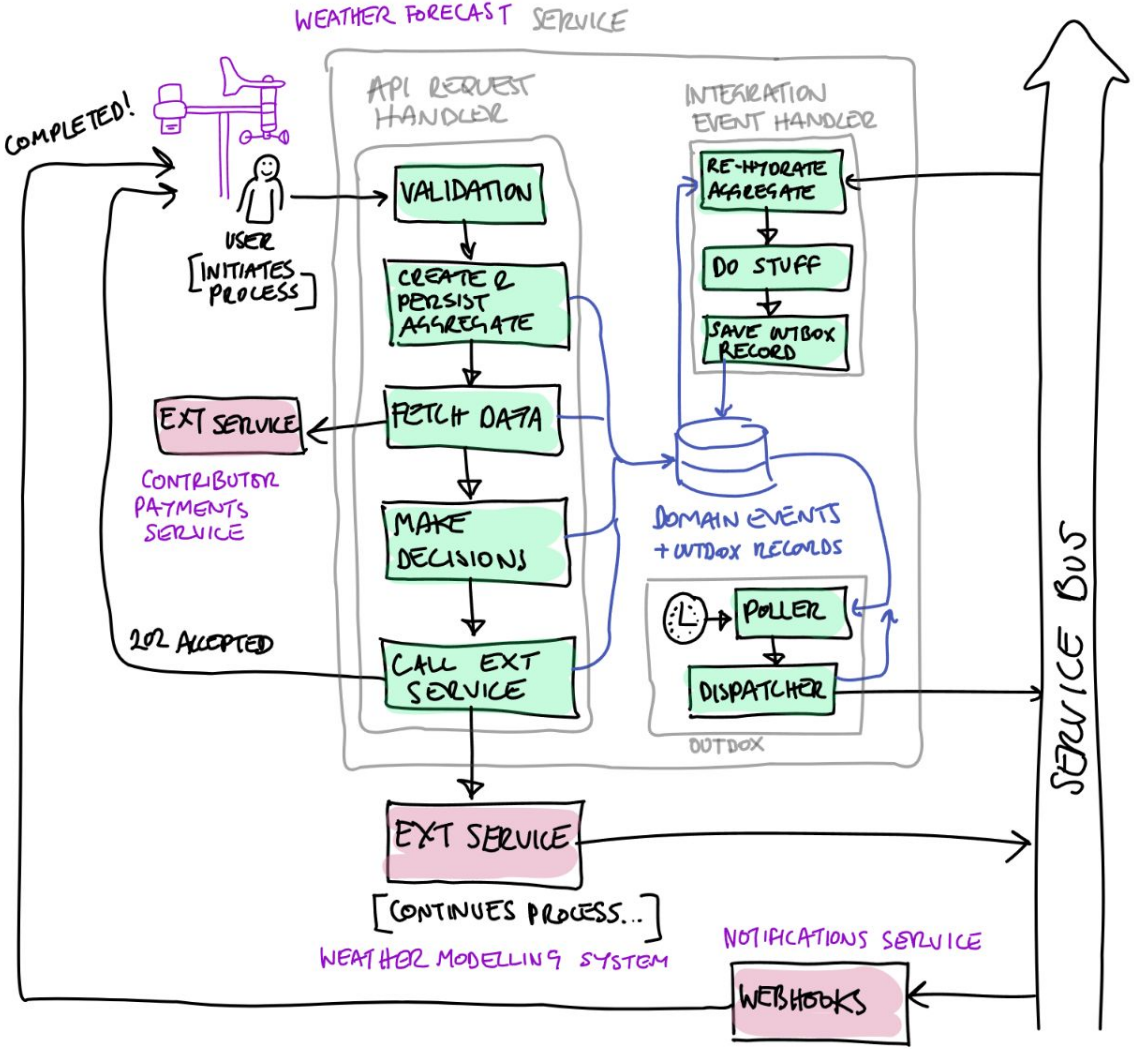
How to get the maximum mileage from a minimal number of tests🚀

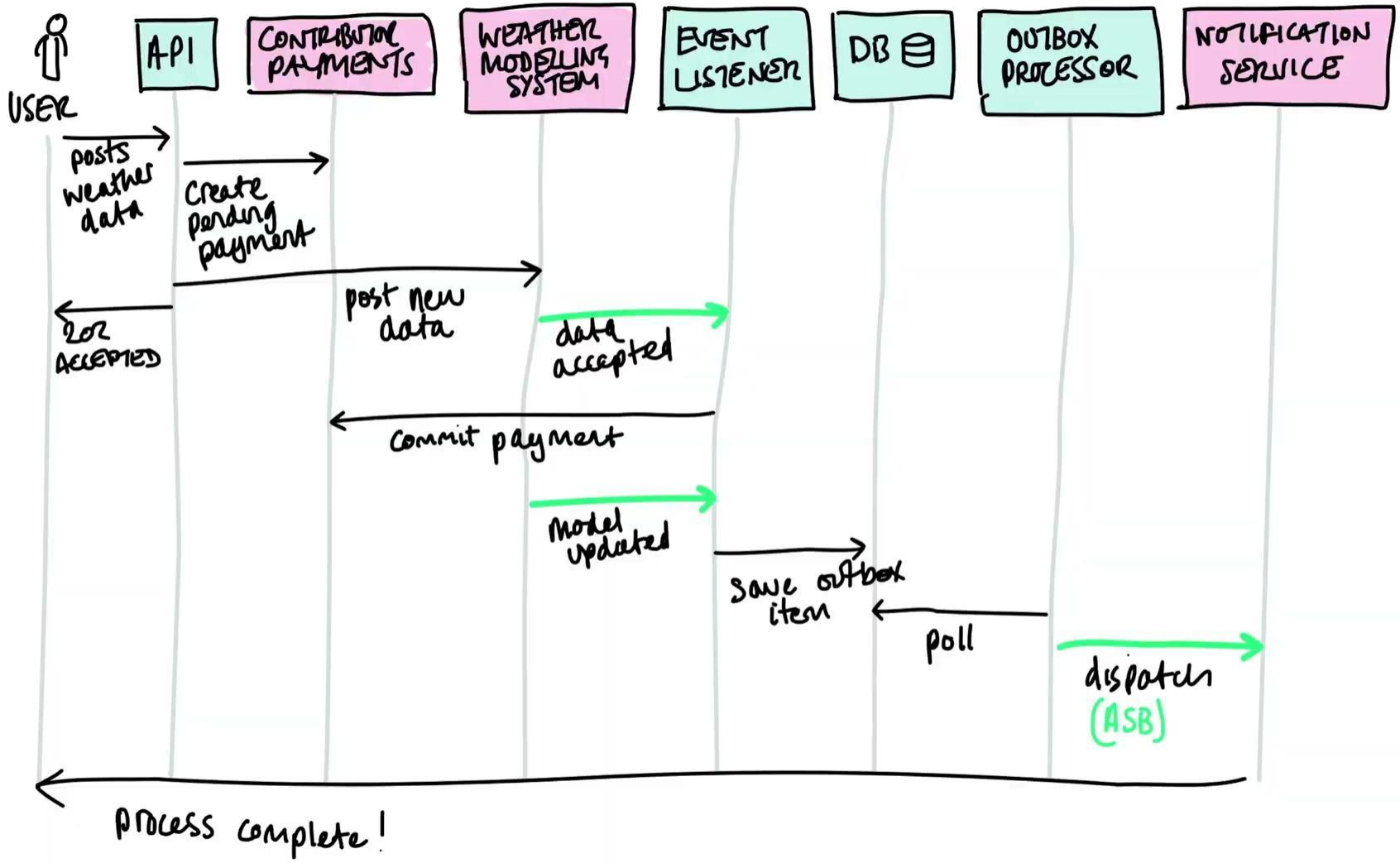
# Integration Tests

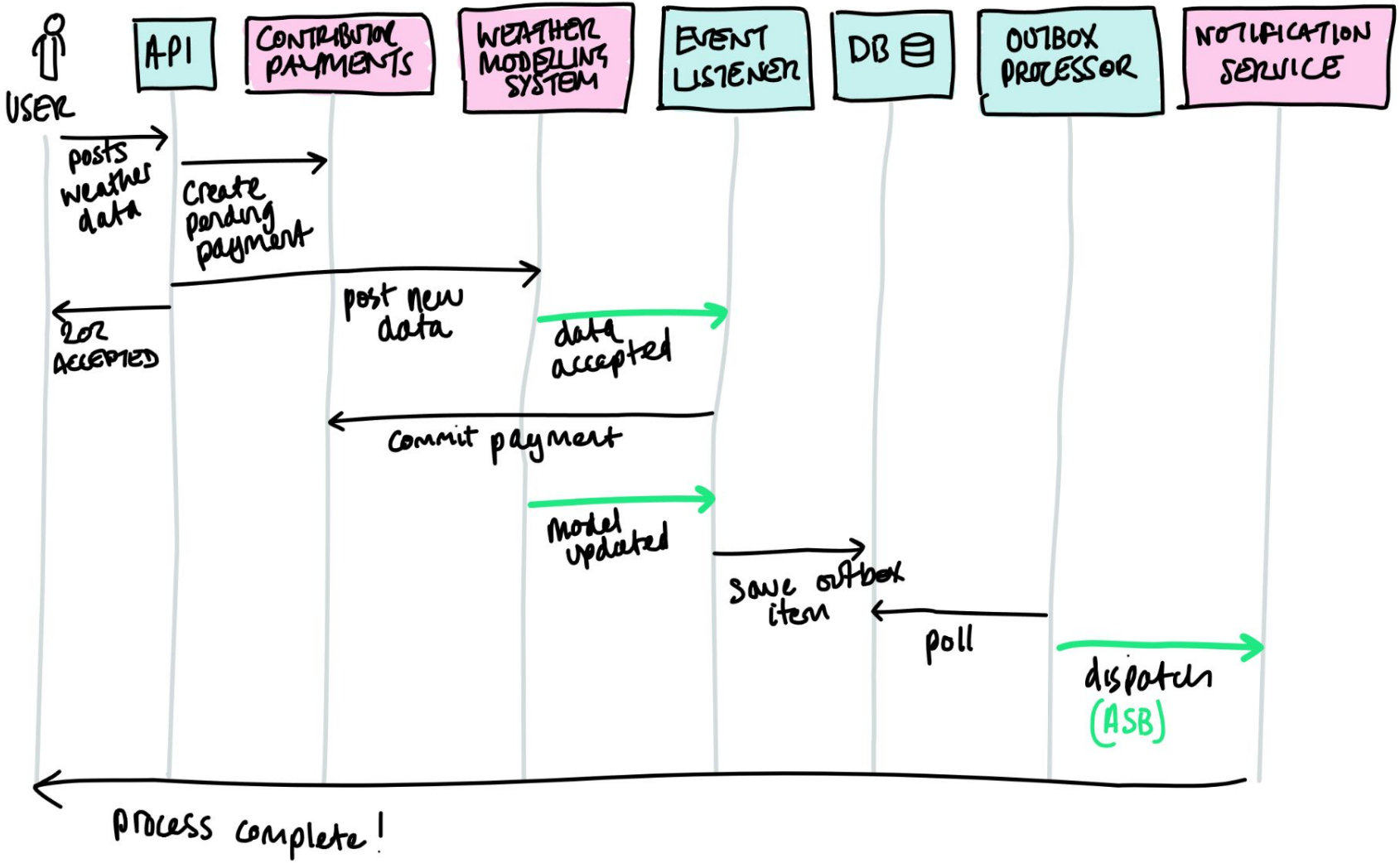
how *not* to hate them!❤️

Let’s start with a scenario…


WEATHER FORECAST SERVICE
COMPLETED!
USER
[INITIATES PROCESS]
API REQUEST HANDLER
VALIDATION
CREATE & PERSIST AGGREGATE
FETCH DATA
MAKE DECISIONS
CALL EXT SERVICE
EXT SERVICE
CONTRIBUTOR PAYMENTS SERVICE
202 ACCEPTED
INTEGRATION EVENT HANDLER
RE-HYDRATE AGGREGATE
DO STUFF
SAVE OUTBOX RECORD
DOMAIN EVENTS + OUTBOX RECORDS
POLLER
DISPATCHER
OUTBOX
SERVICE BUS
EXT SERVICE
[CONTINUES PROCESS...]
WEATHER MODELLING SYSTEM
NOTIFICATIONS SERVICE
WEBHOOKS


WEATHER FORECAST SERVICE
COMPLETED!
USER
[INITIATES PROCESS]
API REQUEST HANDLER
VALIDATION
CREATE & PERSIST AGGREGATE
FETCH DATA
MAKE DECISIONS
CALL EXT SERVICE
EXT SERVICE
CONTRIBUTOR PAYMENTS SERVICE
202 ACCEPTED
INTEGRATION EVENT HANDLER
RE-HYDRATE AGGREGATE
DO STUFF
SAVE OUTBOX RECORD
DOMAIN EVENTS + OUTBOX RECORDS
POLLER
DISPATCHER
OUTBOX
SERVICE BUS
EXT SERVICE
[CONTINUES PROCESS...]
WEATHER MODELLING SYSTEM
NOTIFICATIONS SERVICE
WEBHOOKS


USER
API
CONTRIBUTION PAYMENTS
WEATHER MODELLING SYSTEM
EVENT LISTENER
DB
OUTBOX PROCESSOR
NOTIFICATION SERVICE
posts weather data
create pending payment
post new data
202 ACCEPTED
data accepted
commit payment
model updated
save outbox item
poll
dispatch (ASB)
process complete!


USER
API
CONTRIBUTION PAYMENTS
WEATHER MODELLING SYSTEM
EVENT LISTENER
DB
OUTBOX PROCESSOR
NOTIFICATION SERVICE
posts weather data
create pending payment
post new data
202 ACCEPTED
data accepted
commit payment
model updated
save outbox item
poll
dispatch (ASB)
process complete!

# There are lots of ways to test a piece of software

Every org|team|project|dev has a way

# EndToEnd Component Tests

How to get the maximum mileage from a minimal number of tests🚀

# What?

- Unit is as large as possible - multiple executables!
- Consolidation (one project for multiple components)
- Test as much of the surface area as possible
- Test behaviour *not* impl

Caveat: **only opinions**!

LOAD — PERFORMANCE

E-2-E — INTEGRATION WITH WHOLE SYSTEM

INTEGRATION — CI/CD HAS WORKED

COMPONENT

- IN MEMORY
- FAST
- MOCK/FAKE EXTERNALS
- TEST EVERYTHING FROM PROGRAM.CS ONWARDS!

UNIT — DOMAIN RULES

LOAD — PERFORMANCE

E-2-E — INTEGRATION WITH WHOLE SYSTEM

INTEGRATION — CI/CD HAS WORKED

COMPONENT

- IN MEMORY
- FAST
- MOCK/FAKE EXTERNALS
- TEST EVERYTHING FROM PROGRAM.CS ONWARDS!

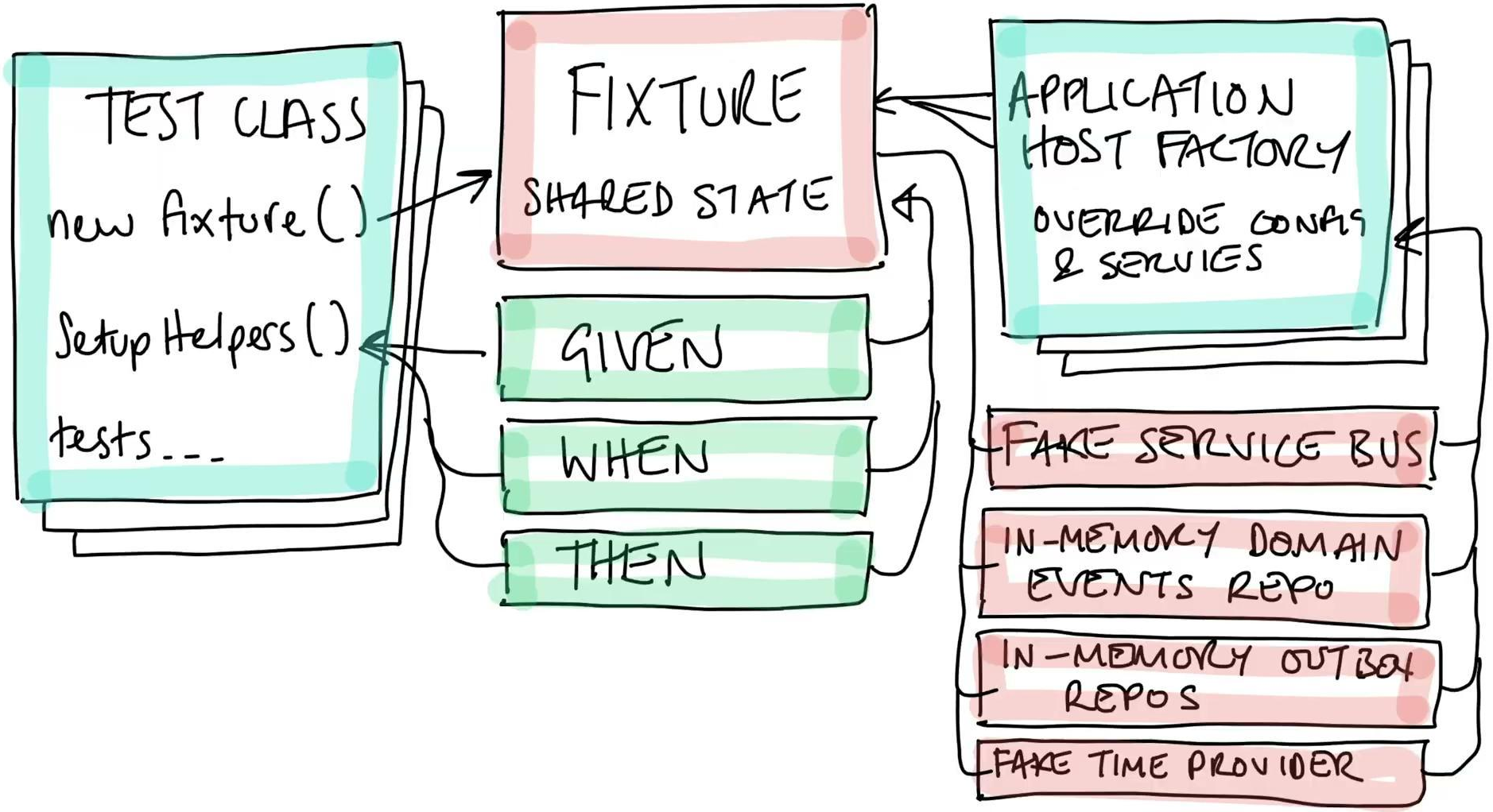
UNIT — DOMAIN RULES

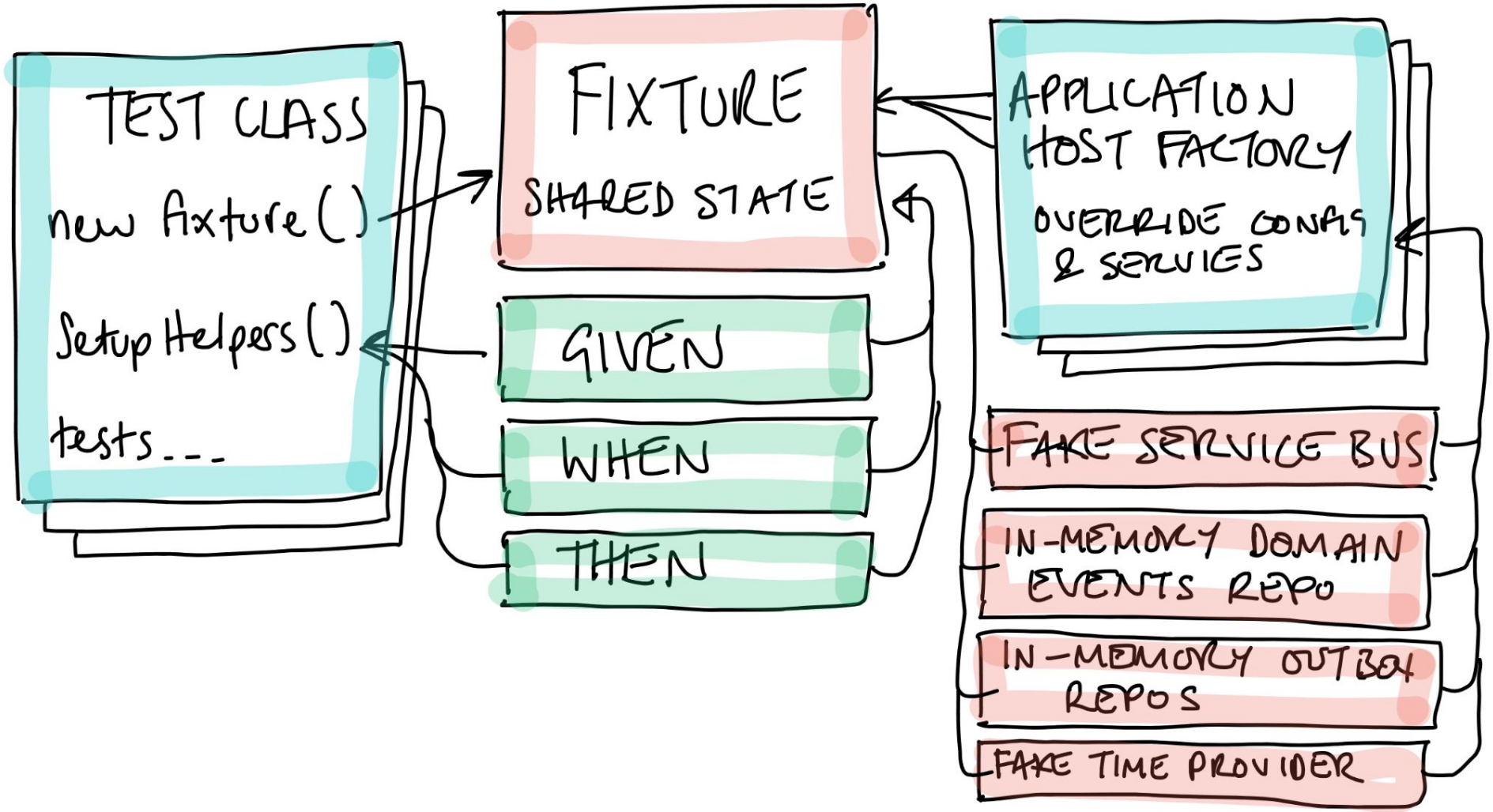
# Why?

- blackbox, refactor away!
- writing tests is not fun, building a test framework *can* be fun!
- Almost as good as running locally😊

# Show me!

```
[Test]
0 references
public void e2e_flow_notifications_sent_when_ModelingDataAccepted()
{
    var (given, when, then) = testFixture.SetupHelpers();

    var testLocation = $"testLocation{Guid.NewGuid()}"[..20];
    var testReference = $"testReference{Guid.NewGuid()}"[..5];

    given.WeHaveSomeCollectedWeatherData(out var weatherData)
        .And.TheContributorPaymentsServicePendingEndpointWillReturn(HttpStatusCode.Accepted)
        .And.TheModelingServiceSubmitEndpointWillReturn(HttpStatusCode.Accepted)
        .And.TheServersAreStarted();

    when.InPhase(newPhase: "1 (initial API request)")
        .And.WeWrapTheCollectedWeatherDataInAnHttpRequestMessage(weatherData, testLocation, testReference, out var httpRequest)
        .And.WeSendTheMessageToTheApi(httpRequest, out var response);

    then.And.TheModelingServiceSubmitEndpointShouldHaveBeenCalled(times: 1)
        .And.TheEventShouldHaveBeenPersisted<SubmittedToModeling>()
        .And.TheResponseCodeShouldBe(response, HttpStatusCode.OK)
        .And.TheBodyShouldNotBeEmpty<WeatherDataCollectionResponse>(response, out var responseBody);

    when.InPhase(newPhase: "2 (1st ASB message back from modeling service)")
        .AMessageAppears(message: new ModelingDataAcceptedIntegrationEvent(responseBody.RequestId));

    then.TheEventShouldHaveBeenPersisted<ModelingDataAccepted>();

    when.InPhase(newPhase: "3 (2nd ASB message back from modeling service)")
        .AMessageAppears(message: new ModelUpdatedIntegrationEvent(responseBody.RequestId));

    then.TheEventShouldHaveBeenPersisted<ModelUpdated>()
        .And.AnOutboxRecordWasInserted();

    then.InPhase(newPhase: "4 (Notification Service handles event dispached by outbox)")
        .AfterSomeTimeHasPassed(numberOfMsToAdvance: 2_000, numberOfMsToWait: 1_000)
        .And.TheMessageWasHandled<ModelUpdatedIntegrationEvent>()
        .And.TheNotificationServiceNotifiedTheUser(testLocation, testReference);
}
```


TEST CLASS
new Fixture()
SetupHelpers()
tests...
FIXTURE
SHARED STATE
GIVEN
WHEN
THEN
APPLICATION HOST FACTORY
OVERRIDE CONFIG & SERVICES
FAKE SERVICE BUS
IN-MEMORY DOMAIN EVENTS REPO
IN-MEMORY OUTBOX REPOS
FAKE TIME PROVIDER


TEST CLASS
new Fixture()
SetupHelpers()
tests...
FIXTURE
SHARED STATE
GIVEN
WHEN
THEN
APPLICATION HOST FACTORY
OVERRIDE CONFIG & SERVICES
FAKE SERVICE BUS
IN-MEMORY DOMAIN EVENTS REPO
IN-MEMORY OUTBOX REPOS
FAKE TIME PROVIDER

# How? #1 Make a nice framework

Given, When & Then or Arrange, Act & Assert etc

```
[Fact]
0 references
public void Return_a_WeatherReport_given_valid_region_and_date()
{
    var (given, when, then) = testFixture.SetupHelpers();

    given.WeHaveAWeatherReportRequest("bristol", DateTime.Now, out var apiRequest)
        .And.TheServersAreStarted();

    when.WeSendTheMessageToTheApi(apiRequest, out var response);

    then.TheResponseCodeShouldBe(response, HttpStatusCode.OK)
        .And.TheBodyShouldNotBeEmpty<WeatherReportResponse>(response,
            x => x.Summary.Should().NotBeEmpty());
}
```

# How? #2 Create test host factory for each executable app

Microsoft.AspNetCore.Mvc.Testing package

```
public class OutboxApplicationFactory(ComponentTestFixture fixture) : WebApplicationFactory<Outbox.Program>
{
    1 reference
    public HttpClient? HttpClient;
    1 reference
    public readonly Mock<ILogger> MockLogger = new();
    3 references
    public Func<OutboxRepositoryInMemory>? SetSharedOutboxRepositories = null;

    0 references
    protected override IHost CreateHost(IHostBuilder builder)
    {
        Environment.SetEnvironmentVariable(variable: "ConnectionStrings__WeatherAppDb", value: "dummyConnectionString");
        Environment.SetEnvironmentVariable(
            variable: $"{nameof(OutboxProcessorOptions)}__{nameof(OutboxProcessorOptions.IntervalBetweenBatchesInSeconds)}", value: "1");

        builder
            .ConfigureServices(IServiceCollection services =>
            {
                services.AddMockLogger(MockLogger);

                services.AddSingleton<TimeProvider>(fixture.FakeTimeProvider);

                fixture.MockServiceBus.WireUpSendersAndProcessors(services);
```

# How? #3 Single test fixture

```
public ComponentTestFixture()
{
    ApiFactory = new(this) { SetSharedEventRepository = () => EventRepositoryInMemory };
    EventListenerFactory = new(this)
    {
        SetSharedEventRepository = () => EventRepositoryInMemory,
        SetSharedOutboxRepositories = () => OutboxRepositoryInMemory
    };
    OutboxApplicationFactory = new(this) { SetSharedOutboxRepositories = () => OutboxRepositoryInMemory };
    NotificationServiceFactory = new(this);

    MockServiceBus = new MockServiceBus(
        string entityName => EntityNames.GetTypeNameFromEntityName(entityName),
        Type type => EntityNames.GetEntityNameFromTypeName(type));

    MockServiceBus.AddSenderFor<UserNotificationEvent>();
    MockServiceBus.AddProcessorFor<ModelingDataAcceptedIntegrationEvent>();
    MockServiceBus.AddProcessorFor<ModelingDataRejectedIntegrationEvent>();
    MockServiceBus.AddProcessorFor<ModelUpdatedIntegrationEvent>();
    MockServiceBus.AddProcessorFor<UserNotificationEvent>();

    MockServiceBus.MessagesSentToSendersWillBeReceivedOnCorrespondingProcessors();

    FakeTimeProvider = new FakeTimeProvider();
    FakeTimeProvider.SetUtcNow(TimeProvider.System.GetUtcNow());
```

# How? #4 Testable service bus processor😎

Azure.Messaging.ServiceBus has a few test hooks included…

```
public class TestableServiceBusProcessor<T> : ServiceBusProcessor where T : class
{
    public List<TestableProcessMessageEventArgs> MessageDeliveryAttempts = [];
```

```
public override Task StartProcessingAsync(CancellationToke
{
    return Task.CompletedTask;
}
```

```
public async Task SendMessage(T request, int deliveryCount = 1)
{
    var args = CreateMessageArgs(request, deliveryCount);
    MessageDeliveryAttempts.Add((TestableProcessMessageEventArgs)args);
    await base.OnProcessMessageAsync(args);
}
```

```
public static ProcessMessageEventArgs CreateMessageArgs(T payload, int deliveryCount = 1)
{
    var payloadJson = JsonSerializer.Serialize(payload, GlobalJsonSerialiserSettings.Defa

    var correlationId = Guid.NewGuid().ToString();
    var applicationProperties = new Dictionary<string, object>
    {
        {"origin", "ComponentTests"}
    };

    var message = ServiceBusModelFactory.ServiceBusReceivedMessage(
        body: BinaryData.FromString(payloadJson),
        correlationId: correlationId,
        properties: applicationProperties,
        deliveryCount: deliveryCount);

    var args = new TestableProcessMessageEventArgs(message);

    return args;
}
```

```
public class TestableProcessMessageEventArgs(ServiceBusReceivedMessage message)
    : ProcessMessageEventArgs(message, null, CancellationToken.None)
{
    public bool WasCompleted;
    public bool WasDeadLettered;
    public bool WasAbandoned;
    public DateTime Created = DateTime.UtcNow;
    public string DeadLetterReason = string.Empty;

    0 references
    public override Task CompleteMessageAsync(ServiceBusReceivedMessage message,
        CancellationToken cancellationToken = new())
    {
        WasCompleted = true;
        return Task.CompletedTask;
    }
```

# How? #5 Mock service bus sender🚌

ServiceBusSender can be Mocked using your favourite mocking framework

```
public Then AMessageWasSent(Mock<ServiceBusSender> senderMock, Func<ServiceBusMessage, bool> match, int times = 1)
{
    senderMock.Verify(x => x.SendMessageAsync(It.Is<ServiceBusMessage>(m => match(m)), It.IsAny<CancellationToken>()), Times.Exactly(times));

    return this;
}
```

If one service sends a message to another, use Callback() or equivalent

```
public void MessagesSentToSendersWillBeReceivedOnCorrespondingProcessors()
{
    foreach (var mockSender in mockSenders)
    {
        if (processors.ContainsKey(mockSender.Key) == false)
            break;

        mockSender.Value.Setup(ServiceBusSender x => x.SendMessageAsync(It.IsAny<ServiceBusMessage>(), It.IsAny<Cancellat
            .Callback<ServiceBusMessage, CancellationToken>((ServiceBusMessage sbm, CancellationToken ctx) =>
            {
                var message = JsonSerializer.Deserialize(sbm.Body, mockSender.Key) ??
                              throw new Exception(message: "unable to deserialise service bus message body");

                var applicationProperties = (Dictionary<string, object>?)sbm.ApplicationProperties;

                var processor = GetProcessorFor(mockSender.Key);
                processor.SendMessage(message, applicationProperties: applicationProperties).GetAwaiter().GetResult();
            });
    }
```

# How? #6 Re-wire Http clients if needed

```
public class EventListenerWebApplicationFactory : WebApplicationFactory<EventListener.Program>
{
    private readonly CustomHttpClientFactory httpClientFactory = new();

    protected override void ConfigureWebHost(IWebHostBuilder builder)
    {
        base.ConfigureWebHost(builder);
        builder.ConfigureServices(services =>
        {
            // Replace standard IHttpClientFactory impl with custom one with any added HTTP clients.
            services.AddSingleton<IHttpClientFactory>(httpClientFactory);
        });
    }

    1 reference
    public void ClearHttpClients() => httpClientFactory.HttpClients.Clear();

    1 reference
    public void AddHttpClient(string clientName, HttpClient client)
    {
        if (httpClientFactory.HttpClients.TryAdd(clientName, client) == false)
        {
            throw new InvalidOperationException($"HttpClient with name {clientName} is already added");
        }
    }
```

```
public class CustomHttpClientFactory() : IHttpClientFactory
{
    public Dictionary<string, HttpClient> HttpClients = [];

    0 references
    public HttpClient CreateClient(string name) =>
        HttpClients.GetValueOrDefault(name)
        ?? throw new InvalidOperationException($"HTTP client is not found for client with name {name}");
}
```

# How? #7 Database Connections

- Replace db connection in IoC container with Mock/Fake backed by in-memory collections
- EFCore in-memory database*
- Or use a real database with something like CSharpSqlTests

# Bending time⏱️

TimeProvider

FakeTimeProvider

```
FakeTimeProvider = new FakeTimeProvider();
FakeTimeProvider.SetUtcNow(TimeProvider.System.GetUtcNow());
FakeTimeProvider.AutoAdvanceAmount = TimeSpan.FromMilliseconds(100);
```

```
services.AddSingleton<TimeProvider>(fixture.FakeTimeProvider);
```

```
await Task.Delay(TimeSpan.FromSeconds(options.IntervalBetweenBatchesInSeconds),
    timeProvider, cancellationToken);
```

```
// Advance the time so the outbox processor wakes up to check for messages...
fixture.FakeTimeProvider.Advance(TimeSpan.FromMilliseconds(numberOfMsToAdvance));
// So cool!😁
```

# Demo😅

# Pros and Cons⚖️

✔️The more reusable the code, the more care/love is justified

✔️Test entire e2e flows in addition to individual phases

✔️Detect config and IoC registration issues

✖️TDD is possible but *is* definitely a bit harder and especially at the start

✖️Still need to run locally to prove integrations/mocked behaviours work as expected etc

# Integration Tests

how *not* to hate them❤️

...also sort out you local dev experience!

# Change of direction alert!⚠

# The problem with integration tests…

They tend to be the last task on a story…😒

change → CI → release → tests, loop is *way* too long🤔

They expose how difficult it is to run things locally😬

# The problem with local dev ex…

Heavily dependant on Compute platform😕

Too many options!🥴

Large differences between local and real running in an environment😖

# Specific issues…

Bacs! 🤯

1) Config
- Too many options!
- Managing git changes - don’t check in secrets!

2) Service Bus topics, subscriptions, queues etc
- Sharing a team env namespace means prefixing etc
- Manual queue creation in Azure portal😒

# Differences between local and real env/pipeline

- ManagedIdentity from within containers

  #IF DEBUG 😒

- VS ≠ AKS | ACA | anything else!

  I.e. Communication with pods without Ingress etc

- No local APIM

  Different base urls per service rather than one

- Auth
- Windows vs Linux

# In an ideal world…

We would:

- Clone a repo
- Run non-integration tests, all pass *first time*
- Run local deploy script/command?
- Run integration tests, all pass *first time*
- From there on, hit [F5] and everything runs nicely😁

## Closing gaps between local and env running…

- Use same tooling as release where possible?
- Use environment variables for config?
- ~~Manage via Powershell~~
- Use external services from a team/test env?
- ~~script/fixture to create ASB entities?~~

.Net Aspire ftw! 😍

# Aspire🚀 exceptional local devex!

✔️Easy to add to an existing app

✔️Define everything in one file

✔️OTLP!

✔️Dashboard!

✔️Supports testing!

🤔Deployment?

Demo needed…

# Deployment…

- infra
  - asb
    - asb.module.bicep
  - modules
    - fetch-container-image.bicep
  - abbreviations.json
  - api.tmpl.yaml
  - contributorpaymentsservice.tmpl.yaml
  - eventlistener.tmpl.yaml
  - main.bicep
  - main.parameters.json
  - notificationservice.tmpl.yaml
  - outbox.tmpl.yaml
  - queryabletracecollector.tmpl.yaml
  - resources.bicep
  - sql.tmpl.yaml
  - weathermodelingservice.tmpl.yaml
- notebooks

```
andrewsLegion  D:\git\event-sourced-but-flow-driven-example  main ≡ ?5
pwsh > azd deploy
```

| Name | Type | Location |
|---|---|---|
| acrxjgzrndt35sxi | Container registry | UK South |
| api | Container App | UK South |
| asb-xjgzrndt35sxi | Service Bus Namespace | UK South |
| cae-xjgzrndt35sxi | Container Apps Enviro... | UK South |
| contributorpaymentsservice | Container App | UK South |
| eventlistener | Container App | UK South |
| law-xjgzrndt35sxi | Log Analytics workspace | UK South |
| mi-xjgzrndt35sxi | Managed Identity | UK South |
| notificationservice | Container App | UK South |
| outbox | Container App | UK South |
| queryabletracecollector | Container App | UK South |
| sql | Container App | UK South |
| weathermodelingservice | Container App | UK South |

```
You
https://portal.azure.com/#@/resource/subscriptions/aaecc46b-3ecb-4daa-8818-aa082f1
```

# Next stage in this journey…

- How does the opinionated nature of Aspire fit into enterprise CI/CD pipeline and existing environments?
- Better solution for asserting against OTEL data?

Andrew Poole

**Constructive feedback wanted!!**

andrew.poole@flagstoneim.com | andrewjpoole@gmail.com

LinkedIn | Github | forkInTheCode.net

Github repo containing the code I showed ⇒

github.com/andrewjpoole/event-sourced-but-flow-driven-example

Including these slides😊

Given when then blogpost

Testing service bus blogpost

CSharpSqlTests blogpost