How to fall in love with automated tests! 😍

Andrew Poole
Senior Backend Engineer @ FLAGSTONE

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

How to fall in love with automated tests! 😍

Andrew Poole
Senior Backend Engineer @ FLAGSTONE

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

EndToEnd Component Tests

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

Integration Tests

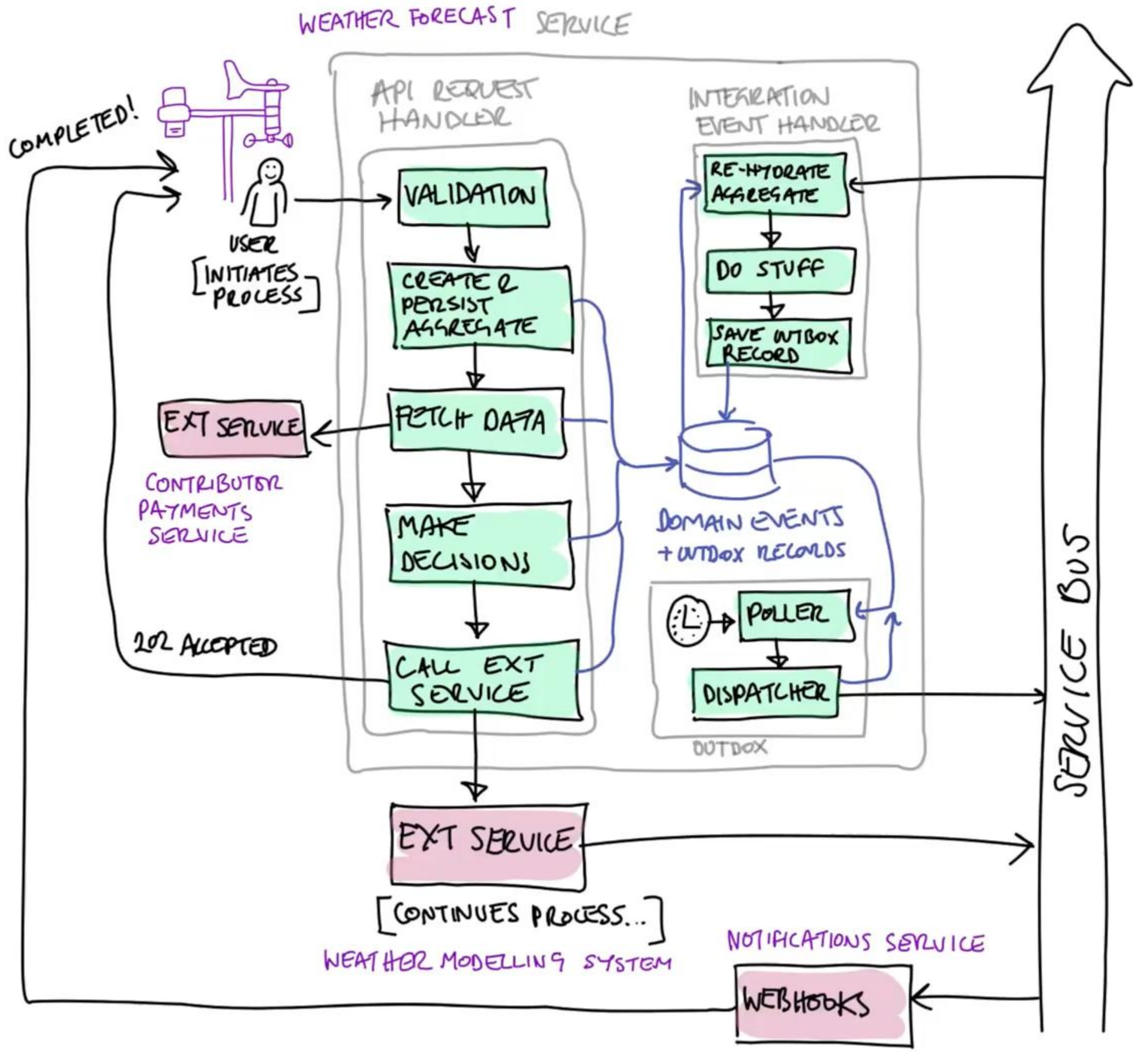
how *not* to hate them❤️

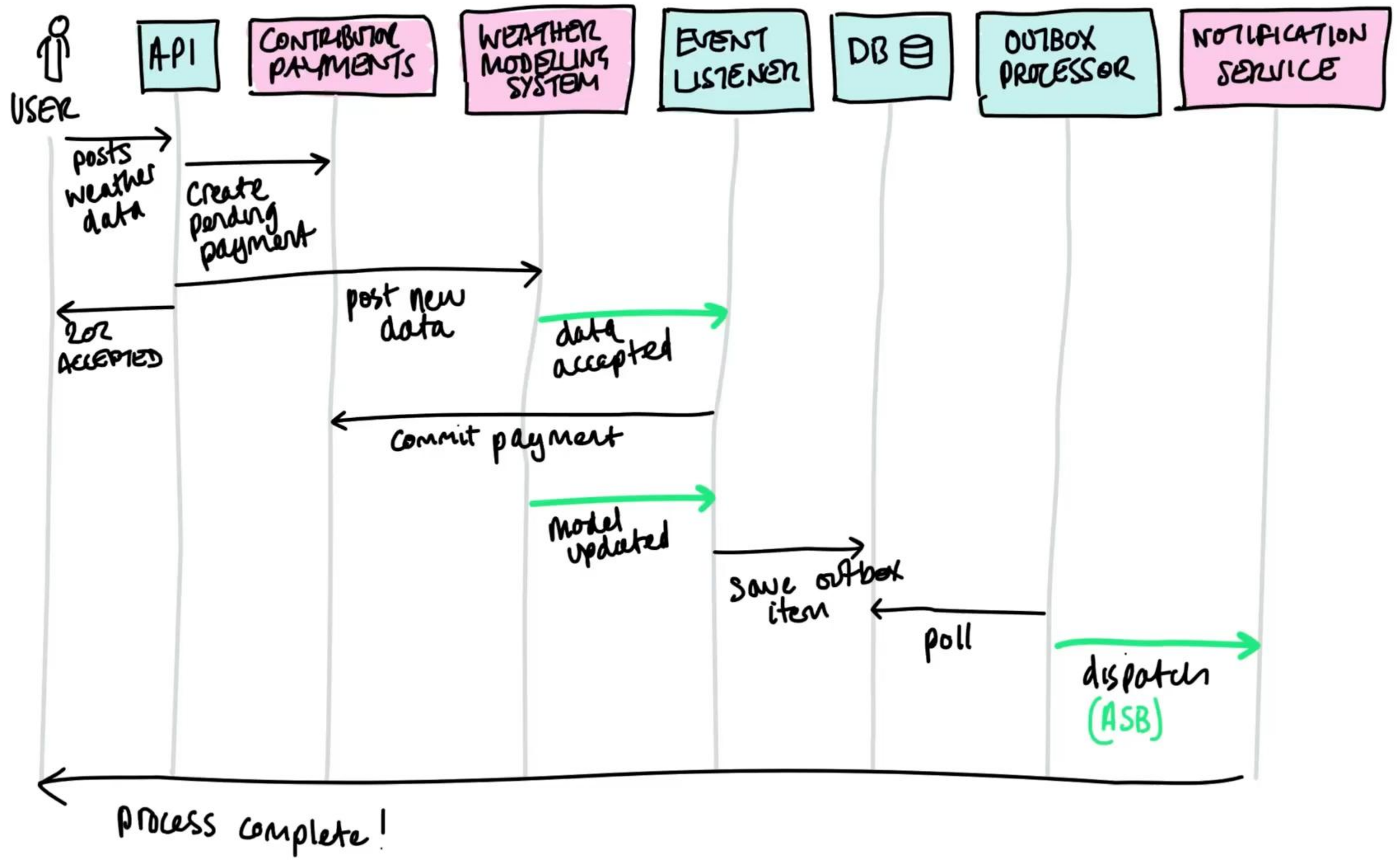
…also sort out your local dev ex!

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

Let’s start with a scenario…

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


WEATHER FORECAST SERVICE
API REQUEST HANDLER
INTEGRATION EVENT HANDLER
COMPLETED!
USER
[INITIATES PROCESS]
VALIDATION
CREATE & PERSIST AGGREGATE
FETCH DATA
MAKE DECISIONS
CALL EXT SERVICE
EXT SERVICE
CONTRIBUTOR PAYMENTS SERVICE
202 ACCEPTED
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!

There are lots of ways to test a piece of software

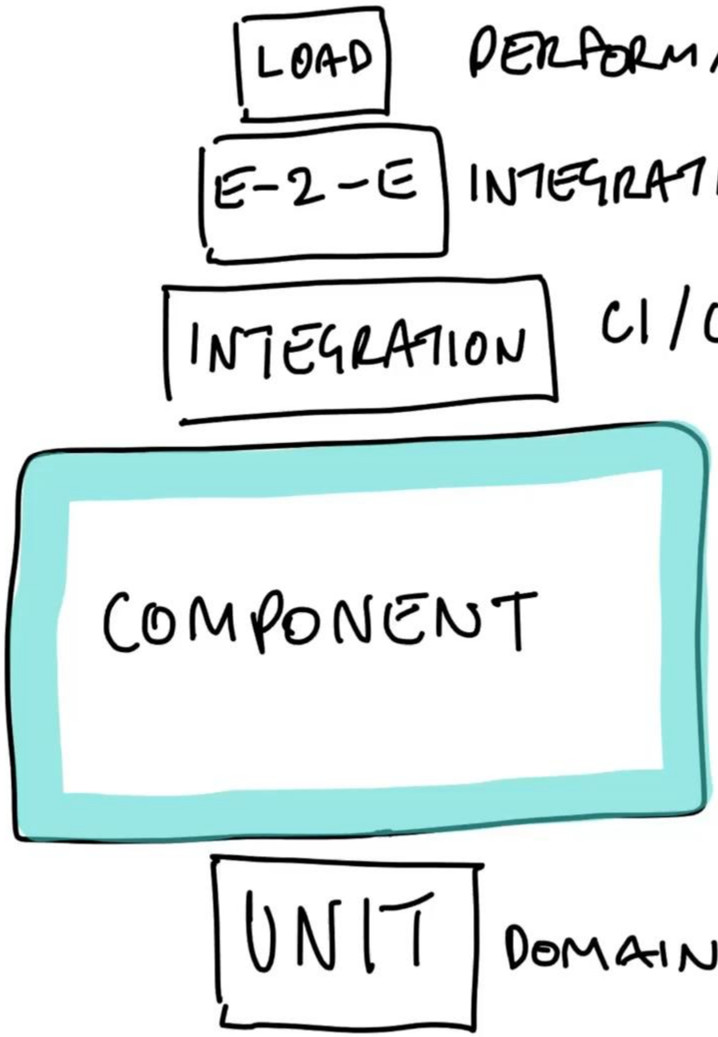
Every org|team|project|dev has a way

EndToEnd Component Tests

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

What?

- Unit is as large as possible - multiple executables!
- Test as much of the surface area as possible
- Test behaviour *not* impl



LOAD PERFORMANCE

E-2-E INTEGRATION WITH WHOLE SYSTEM

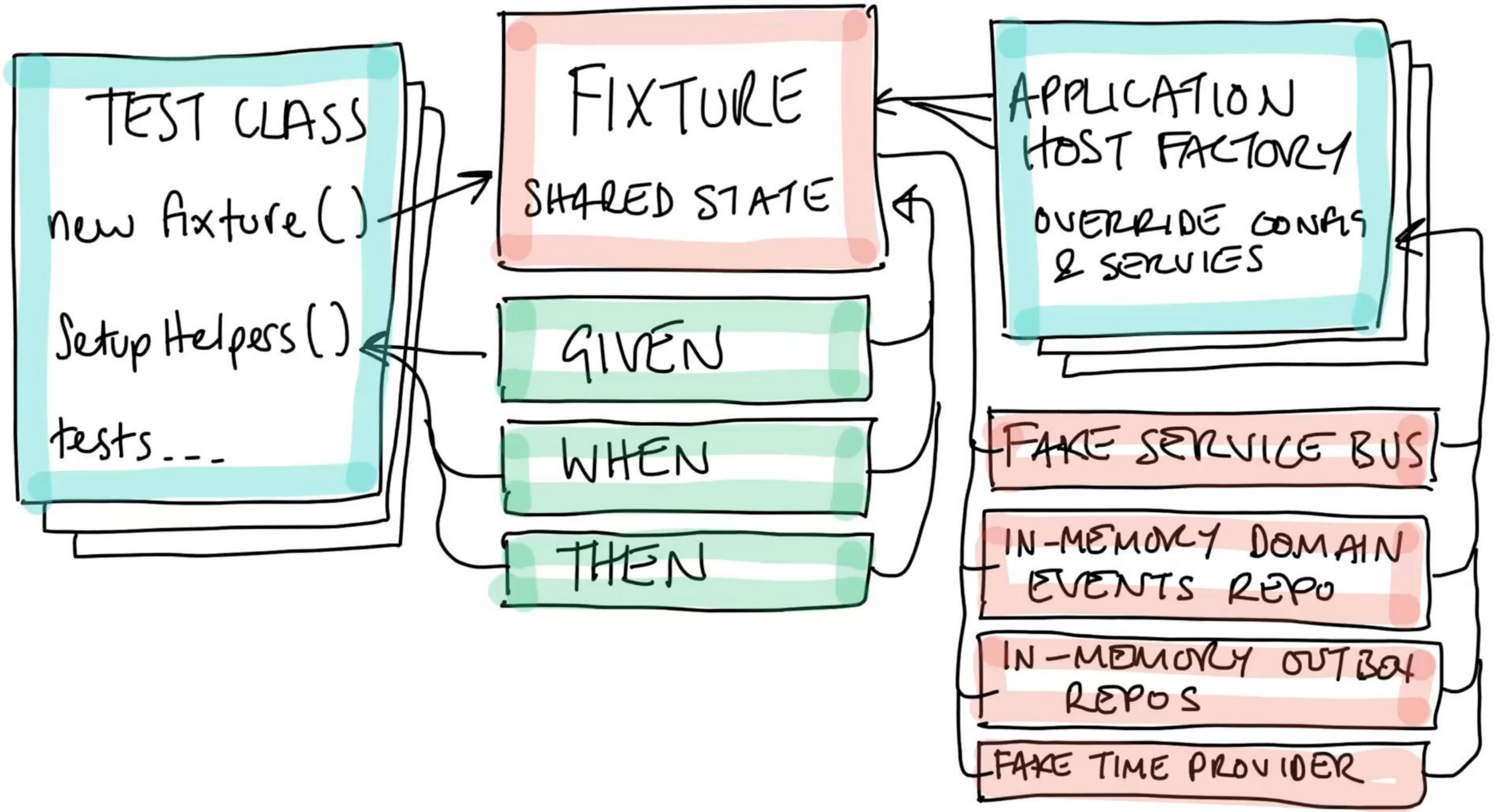
INTEGRATION CI/CD HAS WORKED

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

UNIT DOMAIN RULES

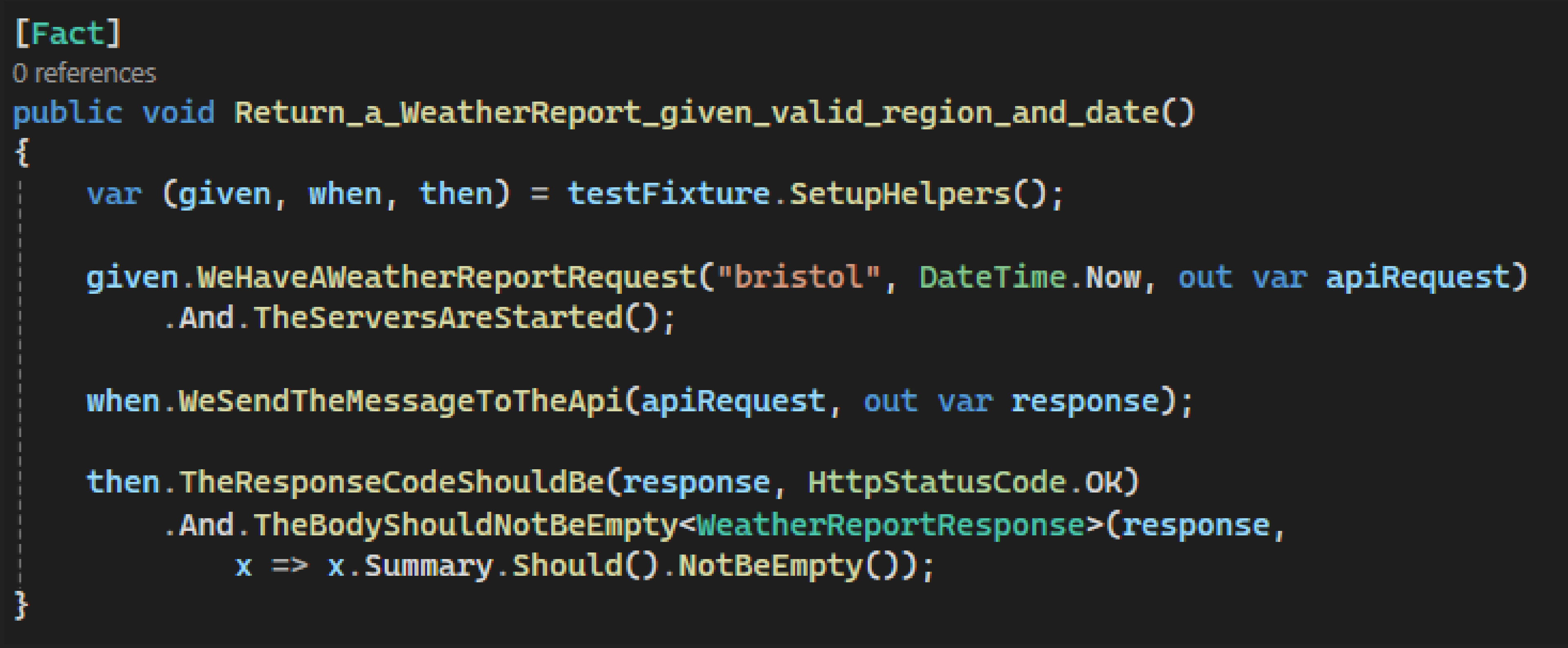
Why?

- Black box, refactor away!
- writing tests is not fun, building a test framework *can* be fun!
- Consolidation
- Probably less code overall
- Almost as good as running locally 😊


TEST CLASS
new fixture()
Setup Helpers()
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



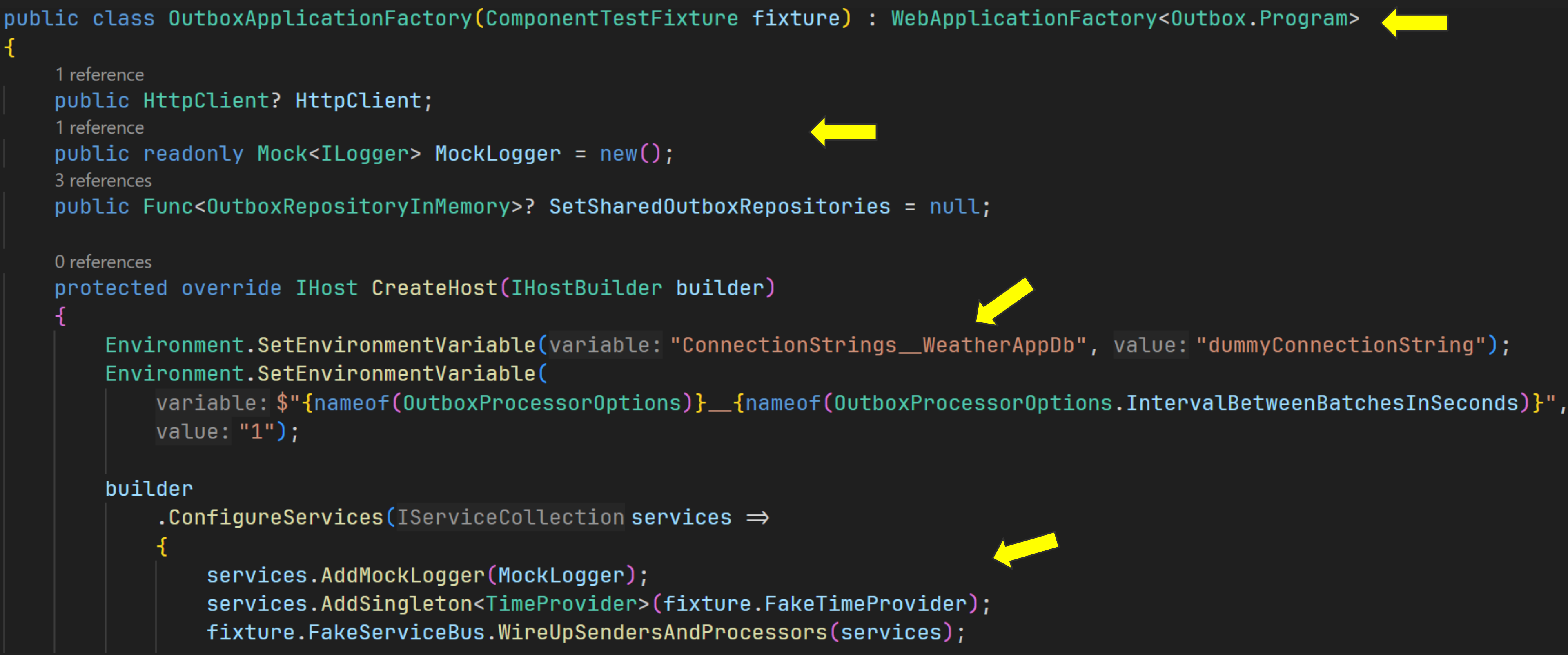
```
[Fact]
0 references
public void Return_a_WeatherReport_given_valid_region_and_date()
{
    var (given, when, then) = testFixture.SetupHelpers();

    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 Single test fixture

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

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

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

    FakeServiceBus.MessagesSentToSendersWillBeReceivedOnCorrespondingProcessors();

    FakeTimeProvider = new FakeTimeProvider();
```

How? #3 Create test host factory for each executable app

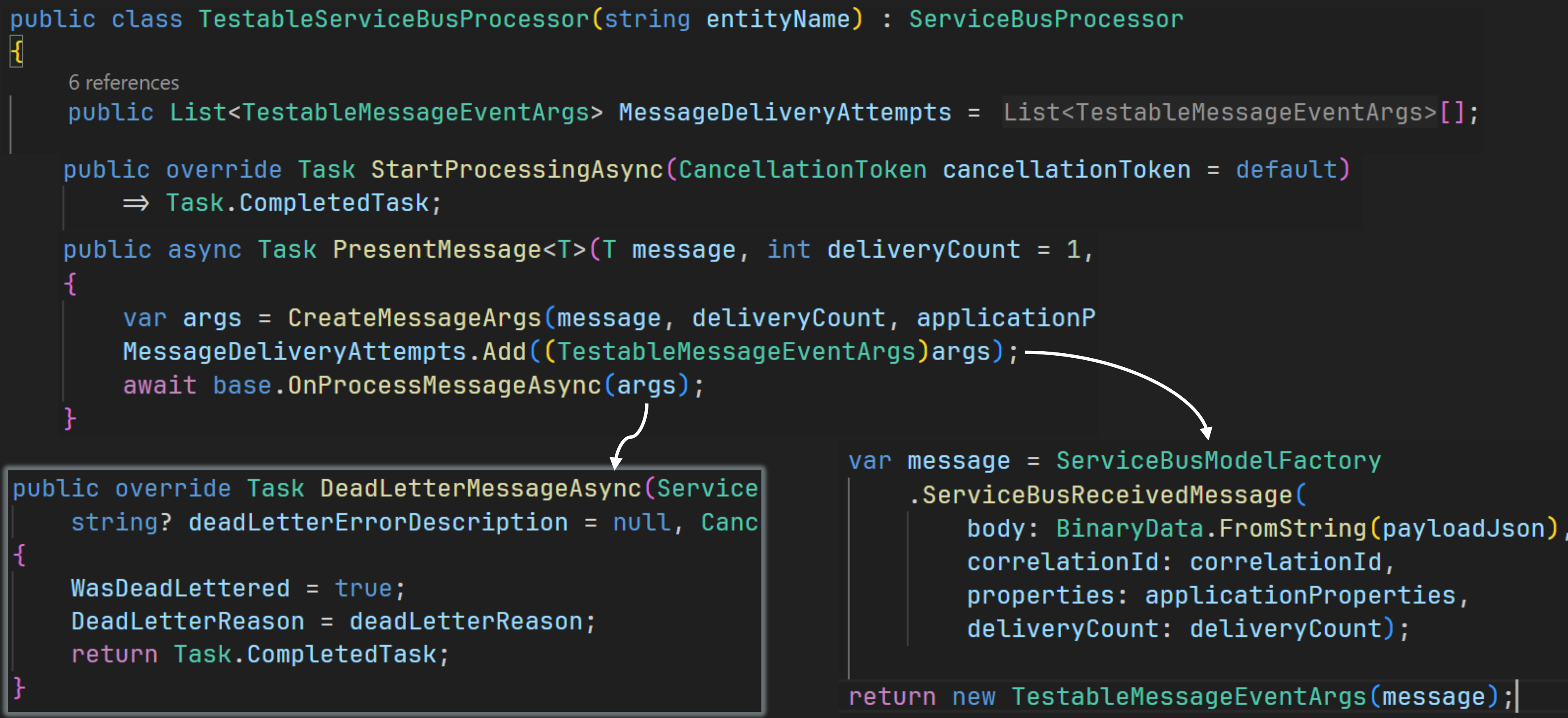
Microsoft.AspNetCore.Mvc.Testing



```
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.FakeServiceBus.WireUpSendersAndProcessors(services);
```

How? #4 Testable service bus processor😎

Azure.Messaging.ServiceBus package includes the ServiceBusModelFactory…



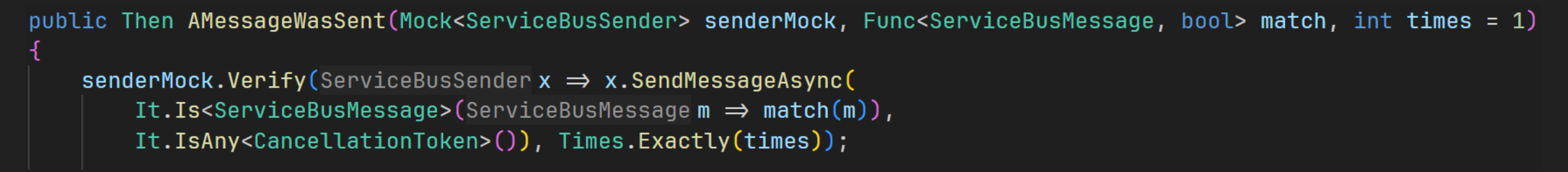
```
public class TestableServiceBusProcessor(string entityName) : ServiceBusProcessor
{
    6 references
    public List<TestableMessageEventArgs> MessageDeliveryAttempts = List<TestableMessageEventArgs>[];

    public override Task StartProcessingAsync(CancellationToken cancellationToken = default)
        ⇒ Task.CompletedTask;

    public async Task PresentMessage<T>(T message, int deliveryCount = 1,
    {
        var args = CreateMessageArgs(message, deliveryCount, applicationP
        MessageDeliveryAttempts.Add((TestableMessageEventArgs)args);
        await base.OnProcessMessageAsync(args);
    }
```

```
public override Task DeadLetterMessageAsync(Service
    string? deadLetterErrorDescription = null, Canc
{
    WasDeadLettered = true;
    DeadLetterReason = deadLetterReason;
    return Task.CompletedTask;
}
```

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

return new TestableMessageEventArgs(message);
```

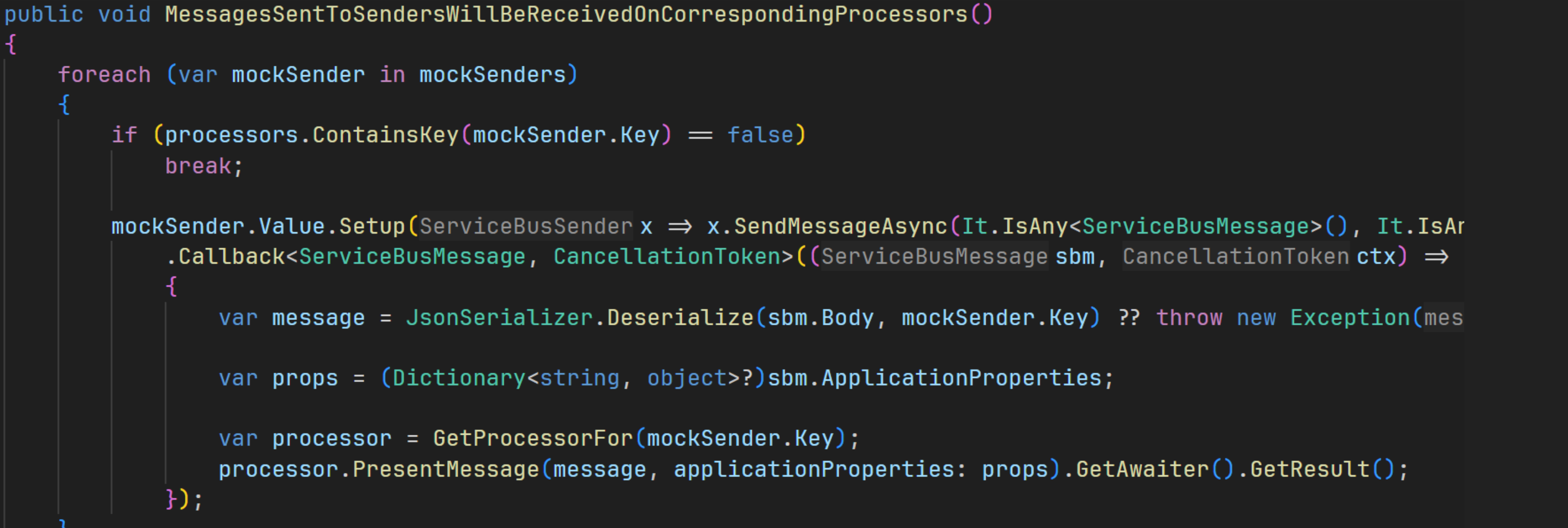
How? #5 Mock service bus sender 🚌

ServiceBusSender can be Mocked using your favourite mocking framework



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

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



```csharp
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.IsAr
            .Callback<ServiceBusMessage, CancellationToken>((ServiceBusMessage sbm, CancellationToken ctx) =>
            {
                var message = JsonSerializer.Deserialize(sbm.Body, mockSender.Key) ?? throw new Exception(mes

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

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

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 😅

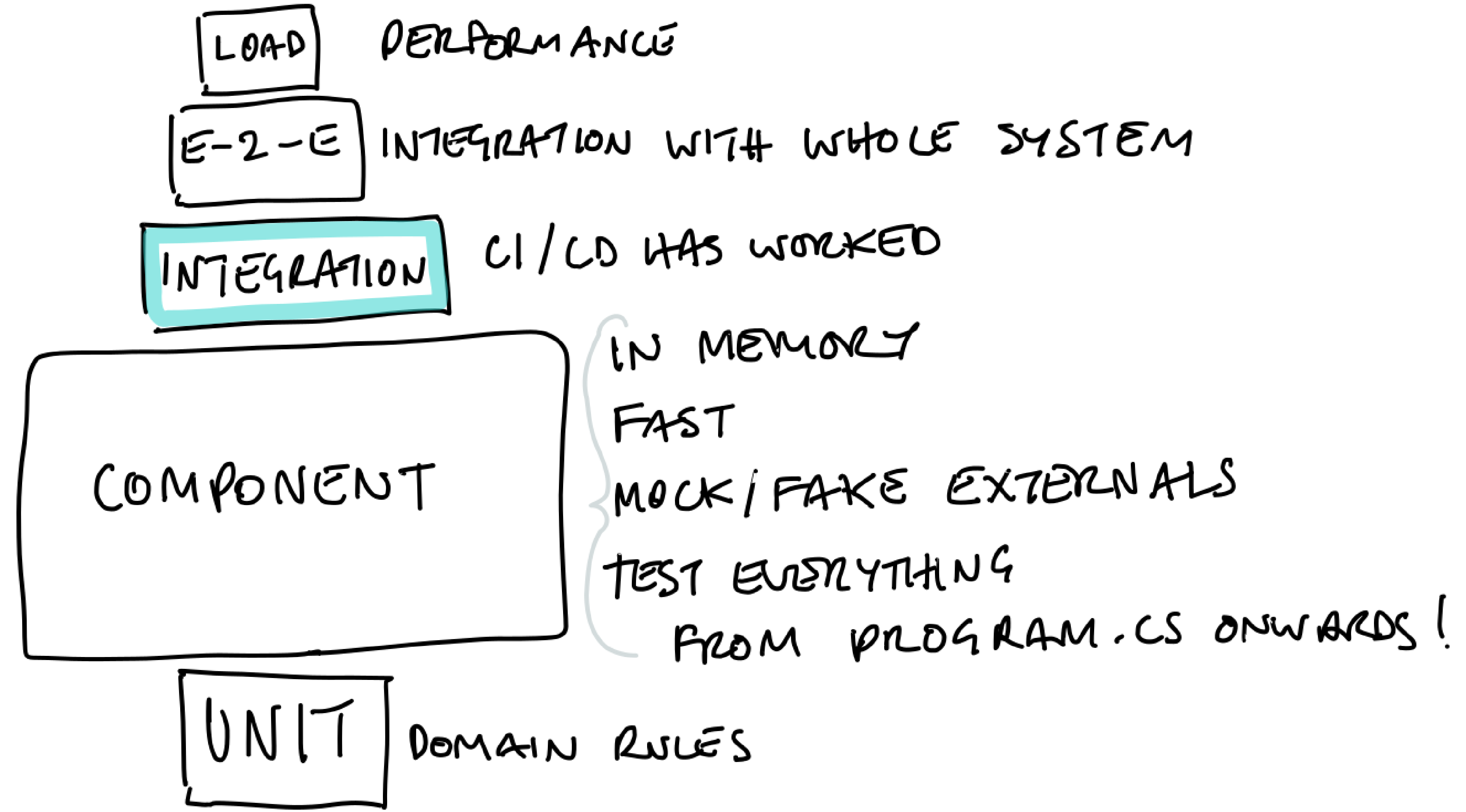
Pros and Cons ⚖️

- ✔️ 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 harder and especially at the start
- ✖️ Still need to run locally to prove integrations/mocked behaviours match real dependencies etc

Integration Tests

how *not* to hate them❤️

...also sort out your local dev experience!


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

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 🤯

Config, _so_ many options

Azure Service Bus topics, subscriptions, queues etc

ManagedIdentity from within containers

Auth

#IF DEBUG 😩

VS ≠ AKS | ACA | anything else!

Azure API Management

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😁

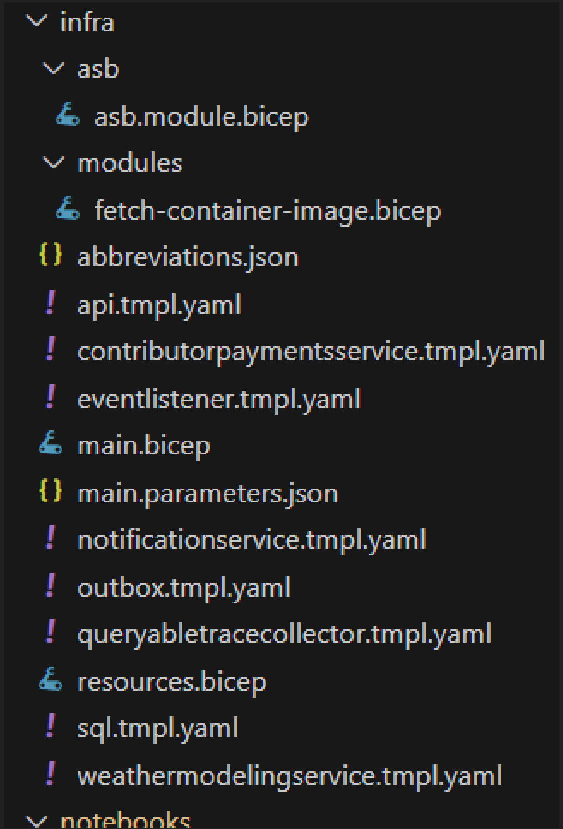
Change of direction alert!😅

Aspire🚀 exceptional local devex!

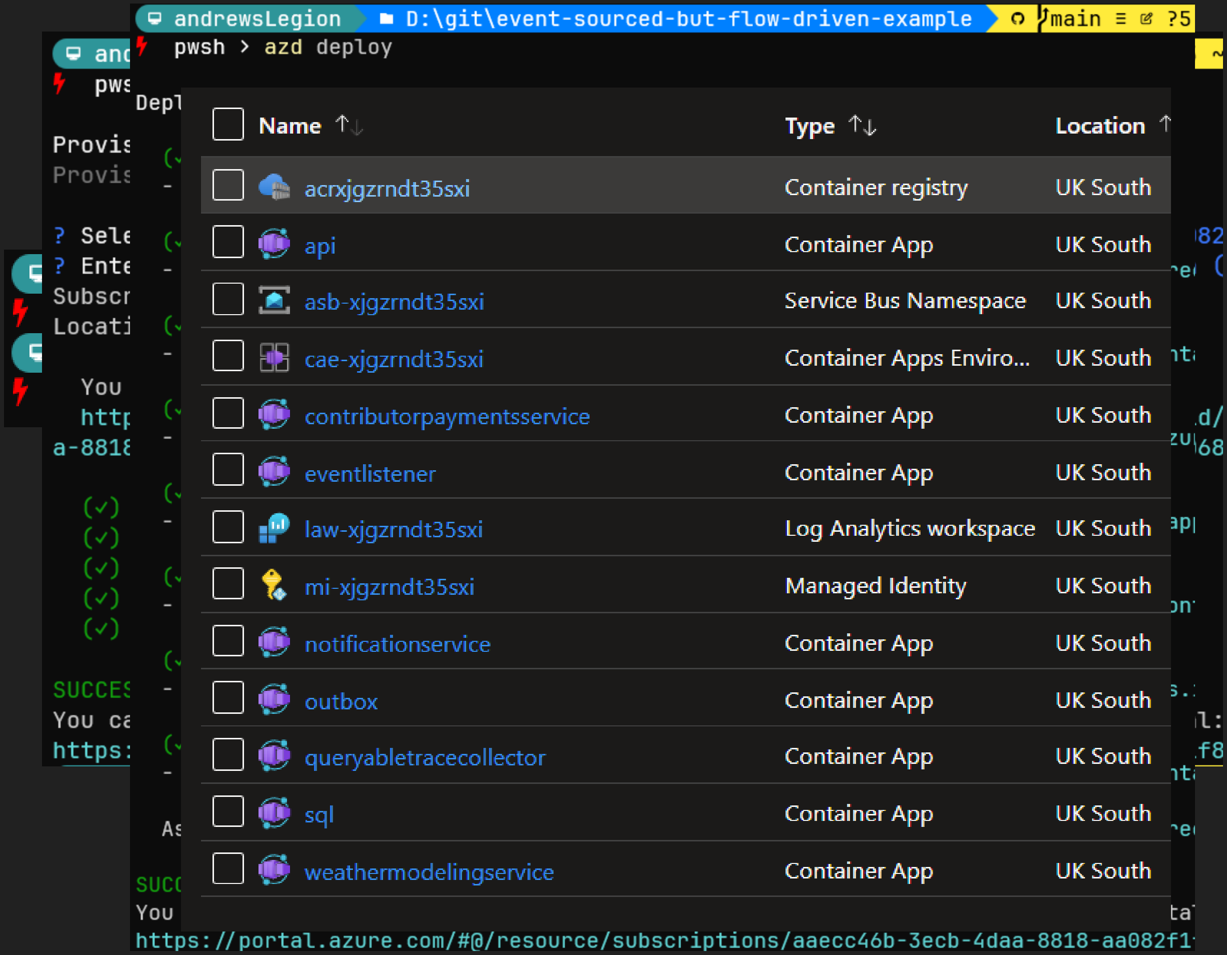
- ✓ Easy to add to an existing app
- ✓ Define everything in one file
- ✓ OTLP!
- ✓ Dashboard!
- ✓ Supports testing!
- 🤔 Deployment?

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
Depl
```

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

```
SUCC
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?🤔
- Aspire + GitHub Actions + Bicep + Azure 🫣
- Better solution for asserting against OTEL data? 🧐
- Do we still need those e2e component tests?😅

Any Questions?

Github repo: everything I showed + lots more cool stuff ⇒

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

Including these slides😊

Andrew Poole

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

LinkedIn | Github | forkInTheCode.net

Given when then blogpost | Testing service bus blogpost | CSharpSqlTests blogpost