

# Testing av http $\mu$ -tjenester

Olav Nybø

# $\mu$ -tjenester

**Consumer**

**Provider**



# $\mu$ -tjenester

Consumer



Provider



# $\mu$ -tjenester

**Consumer**

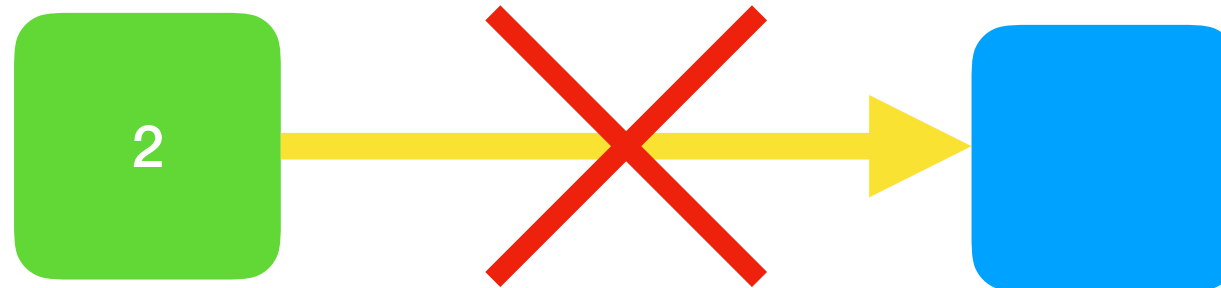
**Provider**



# $\mu$ -tjenester

Consumer

Provider

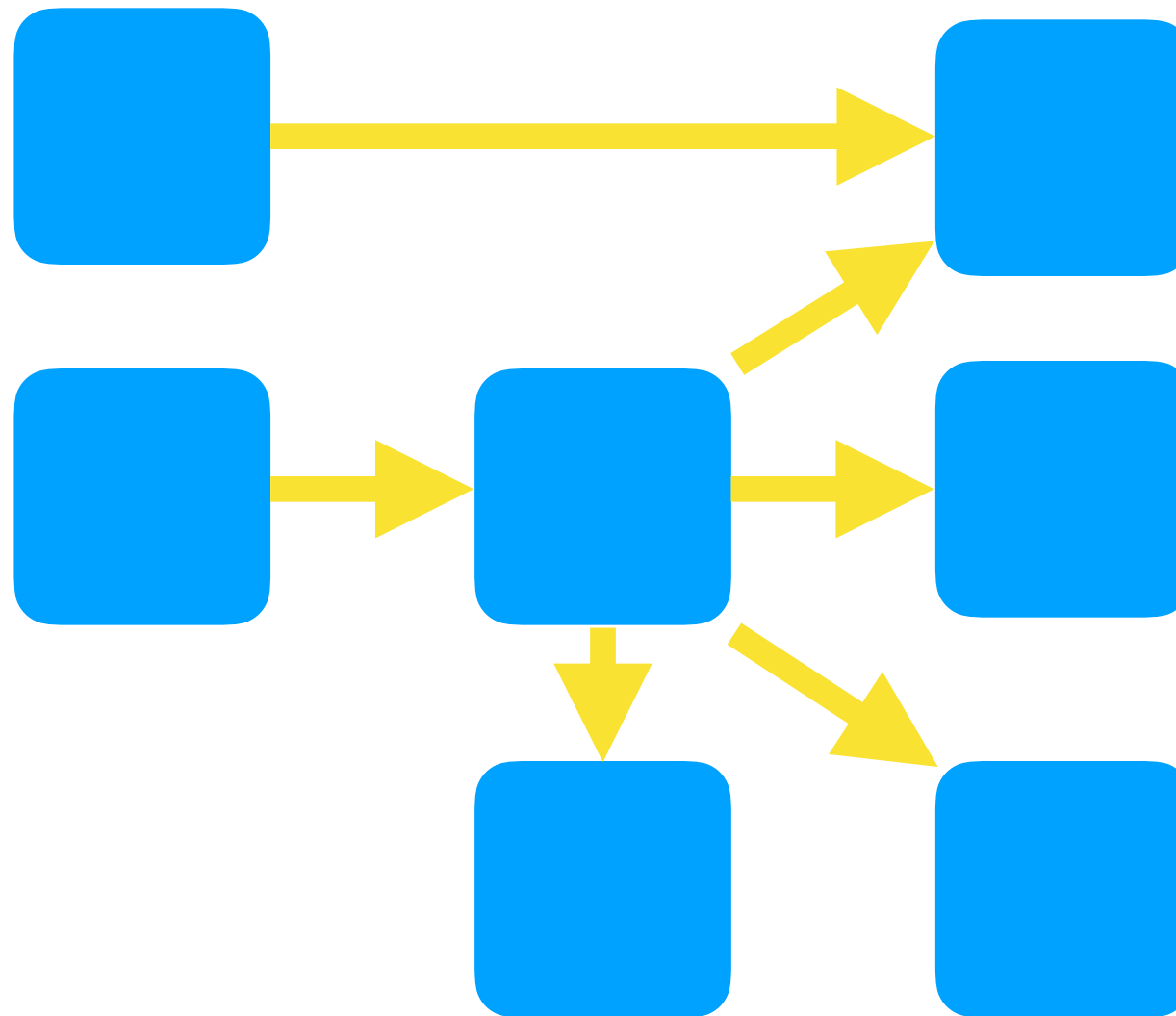




shutterstock.com • 84002389



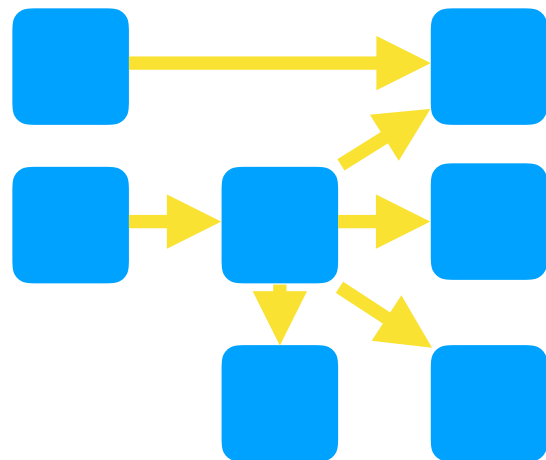
# Microservices



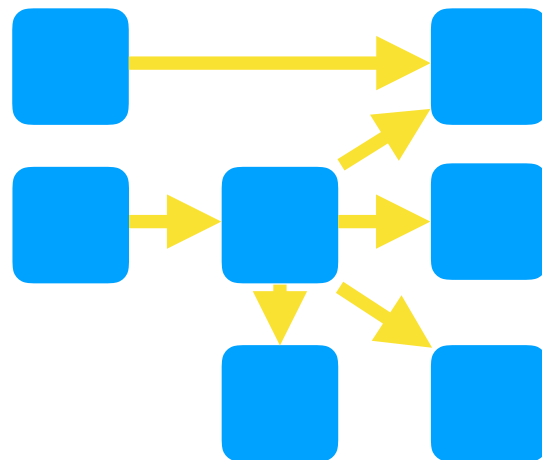


# Multiple environments

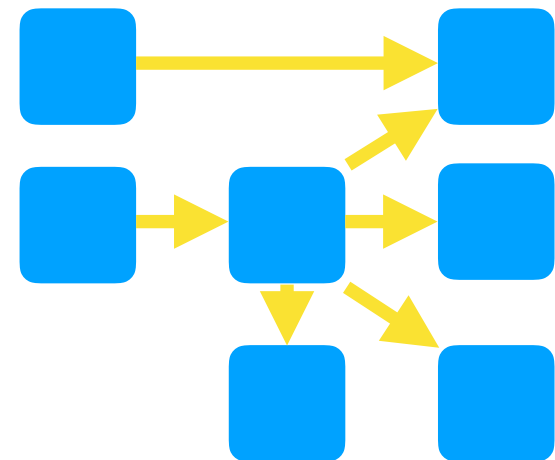
**Dev**



**Test**

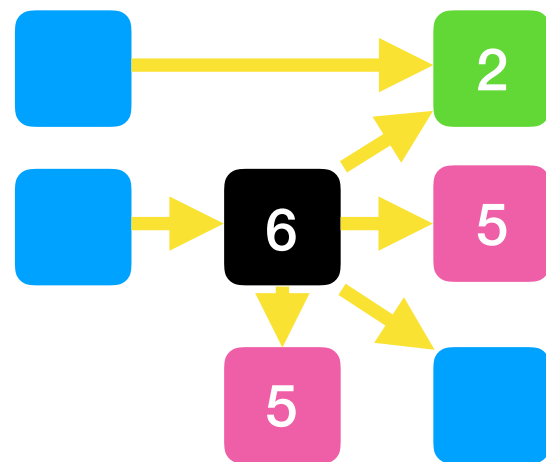


**Prod**

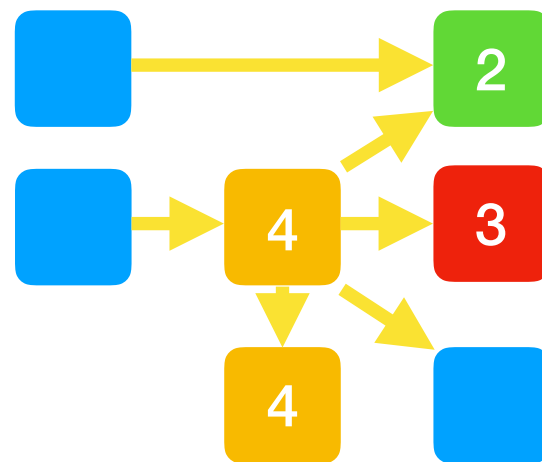


# Multiple versions

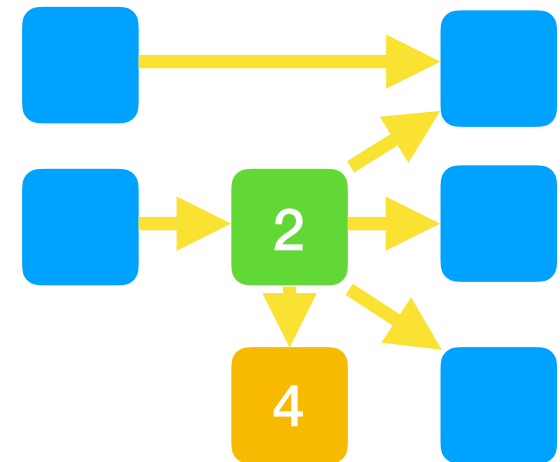
**Dev**



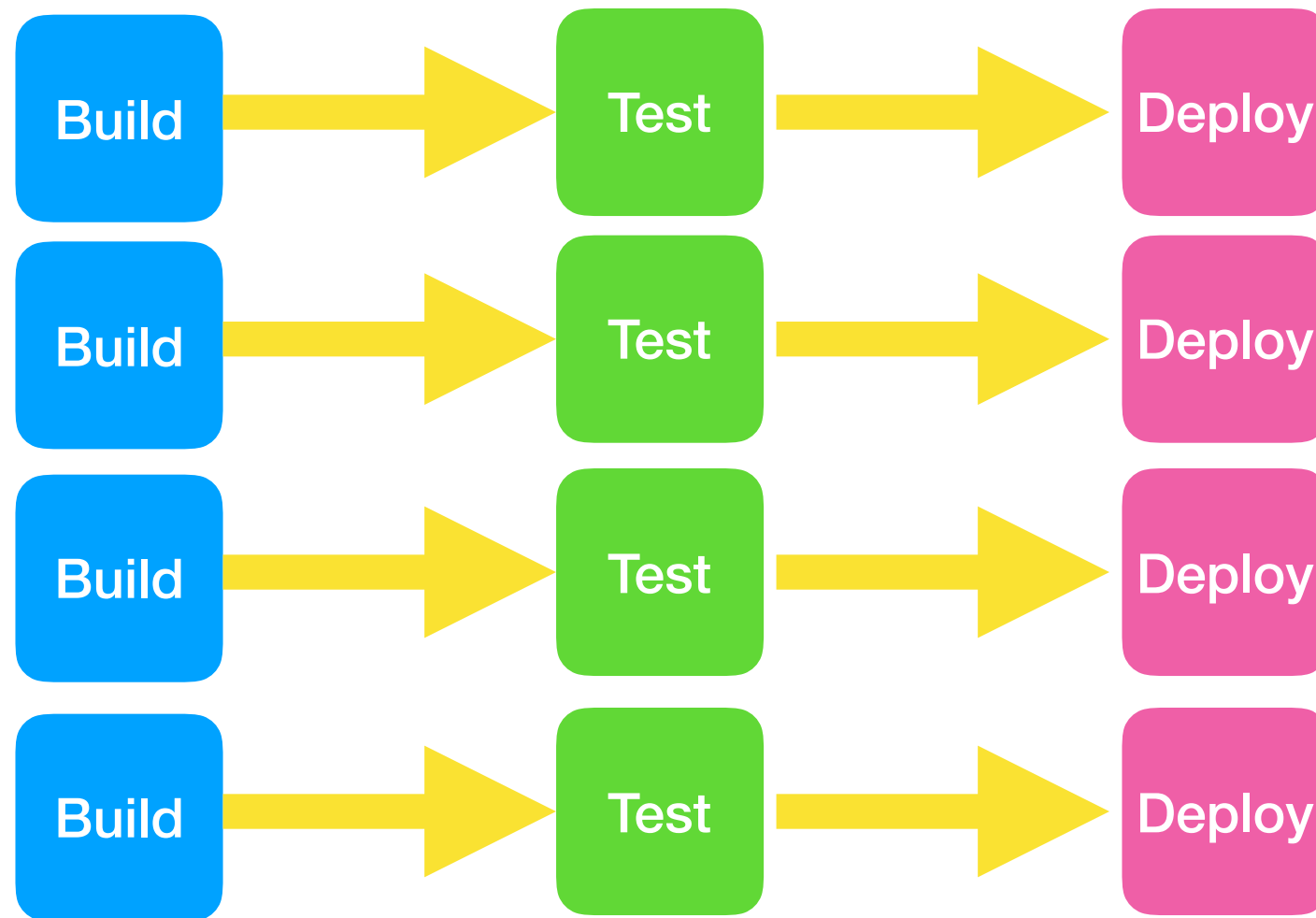
**Test**



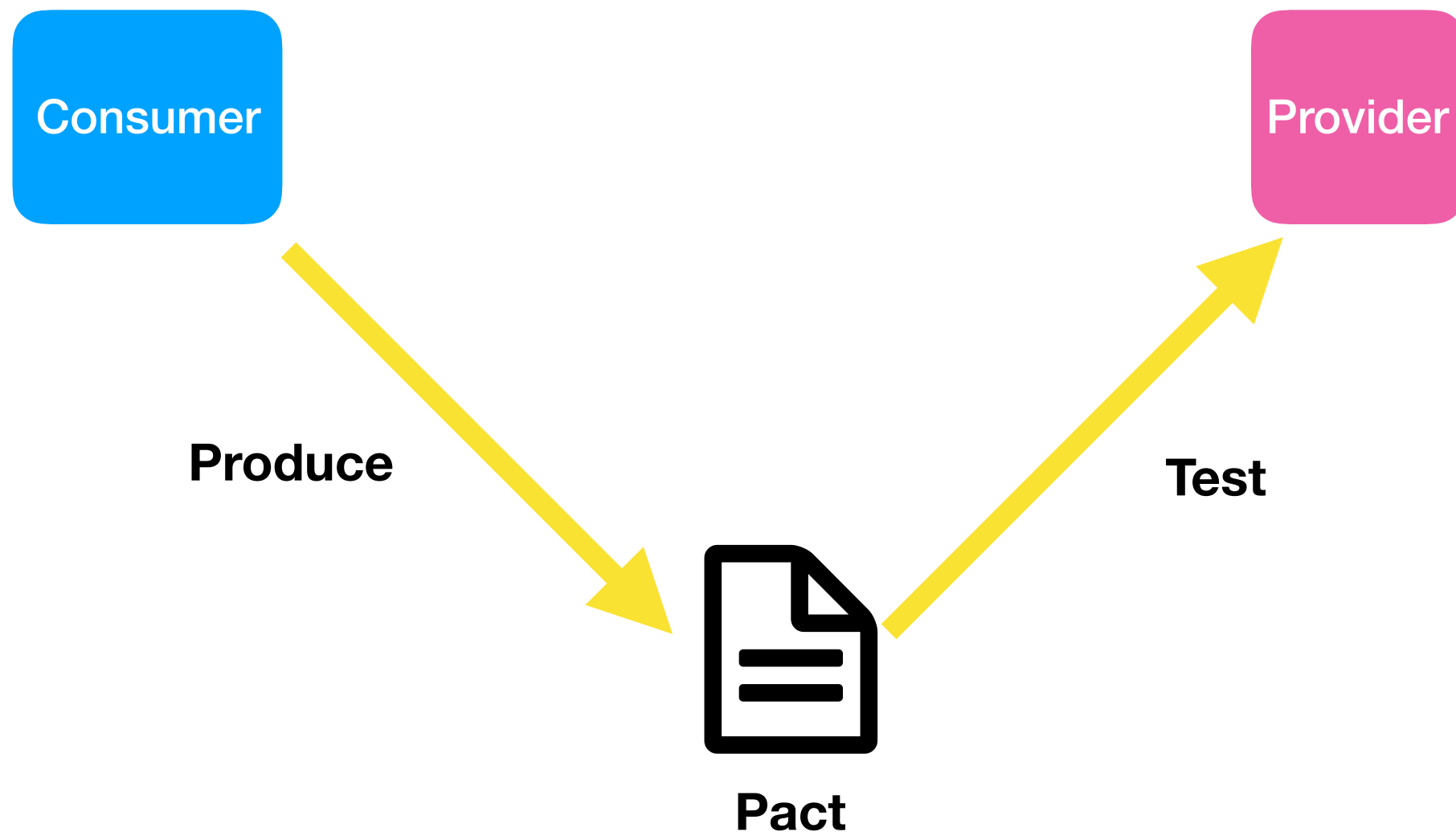
**Prod**



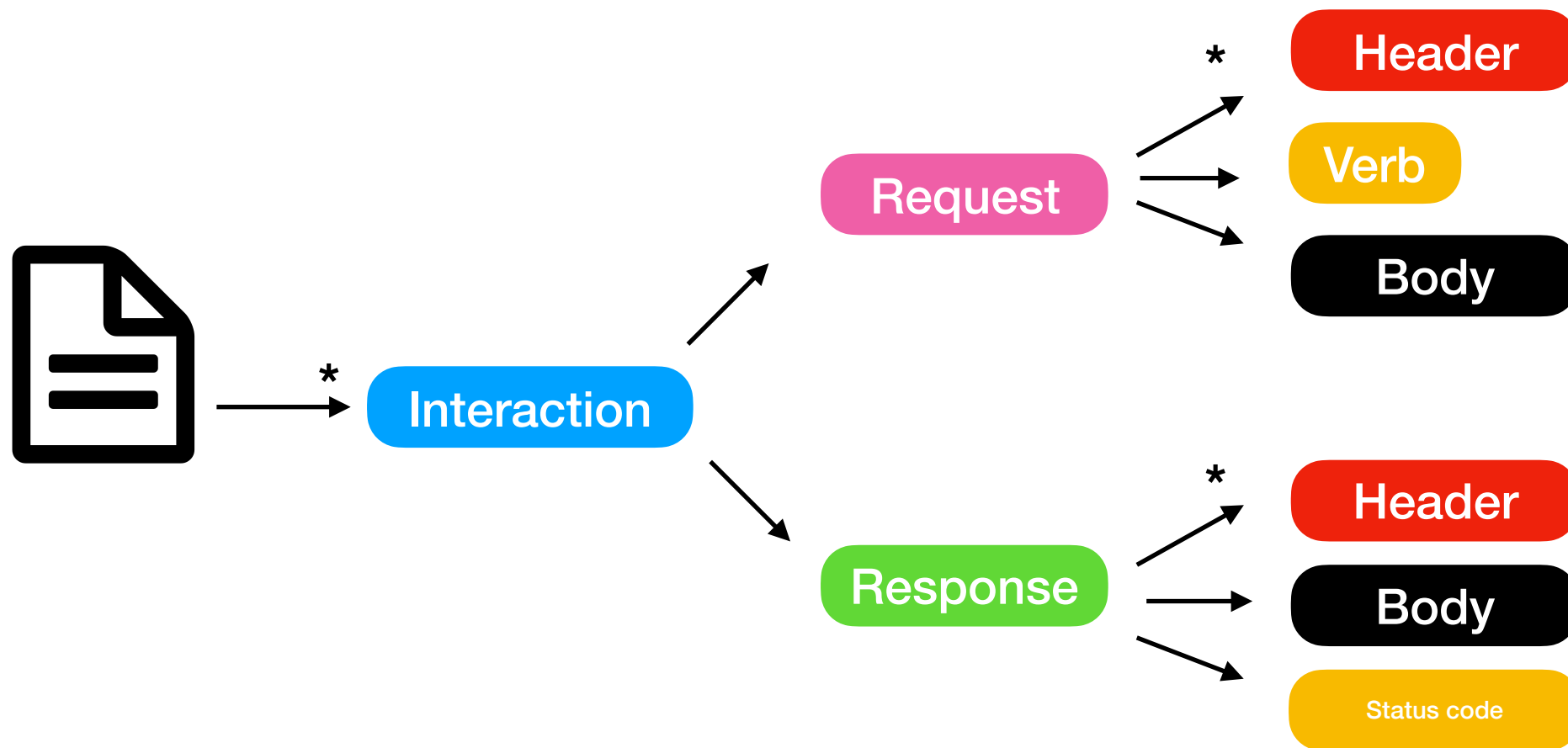
# CI pipeline



# Pact



# Pact



# Integration test

```
var client = new HttpClient { BaseAddress = new Uri(SomeUrl) };

var profileClient = new ProfileClient(client);
var (statusCode, result) =
    await profileClient
        .LookupEmail("torfinn.testeren@gmail.com",
            new AccessToken("accesstoken"));

Assert.NotNull(statusCode);
Assert.Equal(HttpStatusCode.OK, statusCode);
```

# Mock http client

```
var (client, verify) =  
    PactBuilder  
        .UponReceiving("Verify email")  
        .With(new ProviderServiceRequest  
        {  
            Method = HttpVerb.get,  
            Path = "/api/mobile/profile/verifyemail",  
            Query = "email=torfinn.testeren@gmail.com",  
            Headers = new Dictionary<string, string> {{ "Authorization", "Bearer accesstoken" }}  
        })  
        .WillRespondWith(new ProviderServiceResponse  
        {  
            Status = HttpStatusCode.OK,  
            Headers = new Dictionary<string, string> {{ "Content-Type", "application/json" }}  
        }).Client();
```

# Consumer tests

```
[Fact]
public async Task GetUserByEmailTest()
{
    var (client, verify) =
        PactBuilder
            .UponReceiving("Verify email")
            .With(new ProviderServiceRequest
            {
                Method = HttpVerb.get,
                Path = "/api/mobile/profile/verifyemail",
                Query = "email=torfinn.testeren@gmail.com",
                Headers = new Dictionary<string, string> {{ "Authorization", "Bearer accesstoken" }}
            })
            .WillRespondWith(new ProviderServiceResponse
            {
                Status = HttpStatusCode.OK,
                Headers = new Dictionary<string, string> {{ "Content-Type", "application/json" }}
            }).Client();

    var profileClient = new ProfileClient(client);
    var (statusCode, result) =
        await profileClient
            .LookupEmail("torfinn.testeren@gmail.com", new AccessToken("accesstoken"));

    Assert.NotNull(statusCode);
    Assert.Equal(HttpStatusCode.OK, statusCode);

    verify();
}
```



# Provider tests

[Theory]

[MemberData(nameof(InteractionIndexes))]

0 references | Run Test | Debug Test



















```
public async Task EnsureHotelAppPact(int interactionIndex)
{
    using (_host)
    {
        await _host.StartAsync();

        var pactVerifier = new PactVerifier((condition, message) =>
            Assert.True(condition, message), CreatePactFetcher());
        await pactVerifier
            .ProviderState($"provider-states/")
            .ServiceProvider("atlas.profile", ServiceUri)
            .HonoursPactWith("atlas.hotel-app-backend")
            .Verify(interactionIndex);
    }
}
```





# Pacts

Consumer ↓↑		Provider ↓↑	Latest pact published	Webhook status	Last verified
atlas.booking	 	atlas.company	about 9 hours ago	Create	4 months ago
atlas.booking	 	atlas.operaproxy	about 9 hours ago	Create	
atlas.booking	 	atlas.payment	about 9 hours ago	Create	
atlas.booking	 	operaproxy	4 months ago	Create	4 months ago
atlas.hotel.app.backend	 	atlas.booking	about 5 hours ago	Create	
atlas.hotel.app.backend	 	atlas.company	about 5 hours ago	Create	
atlas.hotel.app.backend	 	atlas.profile	about 5 hours ago	Create	
atlas.identity	 	atlas.operaproxy	about 6 hours ago	Create	
atlas.push	 	atlas.profile	6 days ago	Create	

