

# System Integration Testing

---



**Billy Korando**  
SENIOR SOFTWARE  
@BillyKorando

# Agenda

**Define system integration testing**

**Introduction to Test Containers and Docker**

**Overview of Spring Cloud Contract**

# What is System Integration Testing?

---

# What is System Integration Testing?



Verify correctness

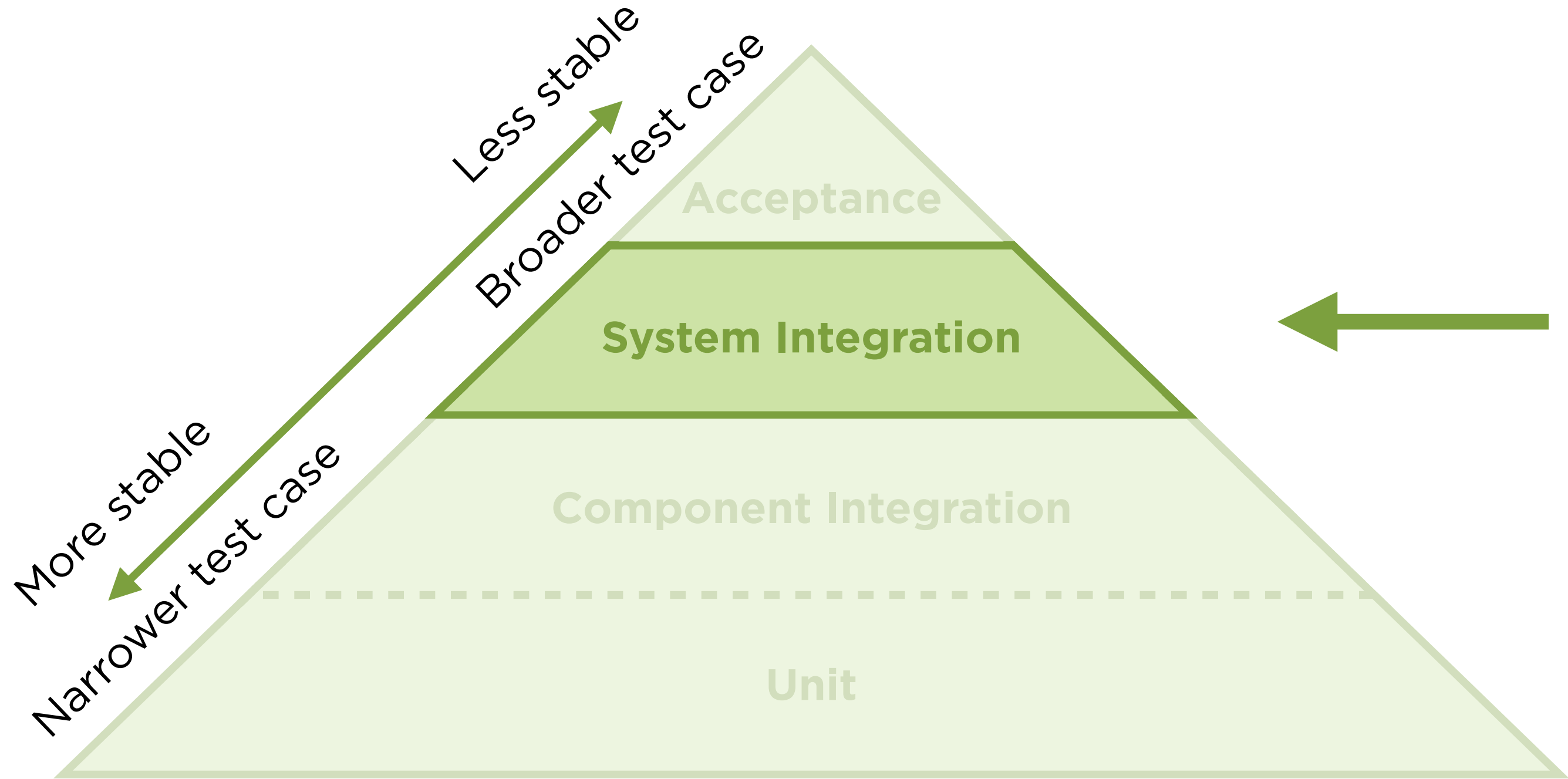


Document behavior

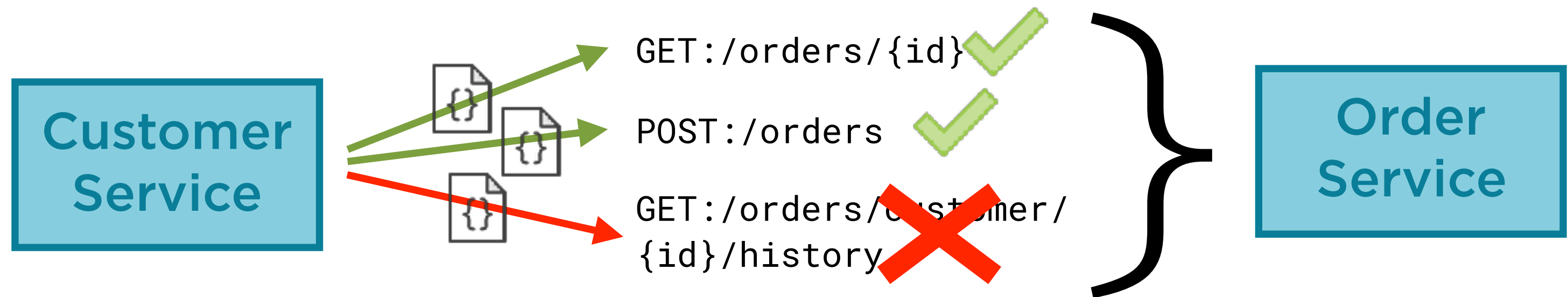


Detect regression

# What is System Integration Testing?



# What is System Integration Testing?

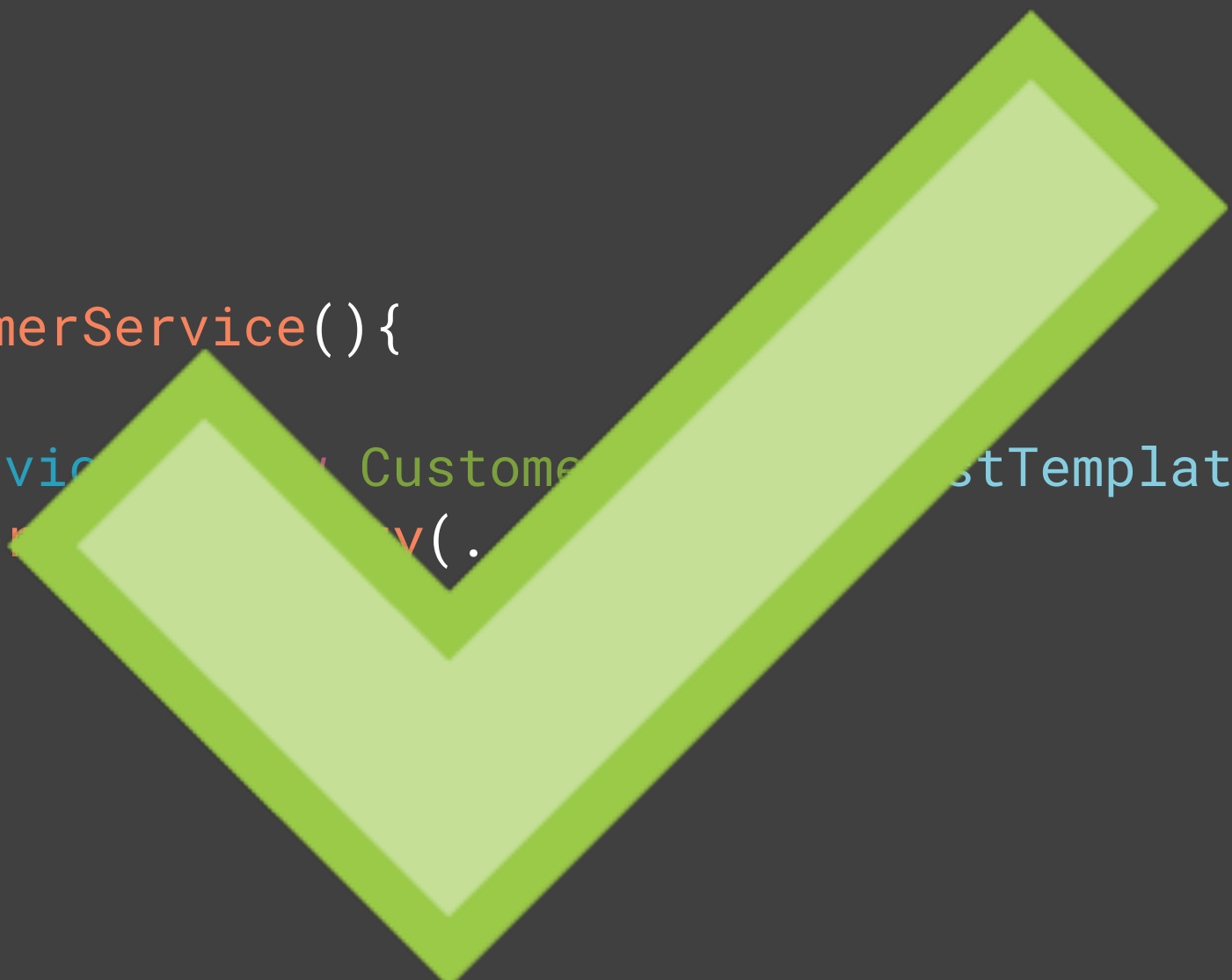


# What is System Integration Testing?

```
@Service
public class CustomerService{
    public List<OrderHistory> findCustomerOrderHistory(...){
        return restTemplate.get("/orders/customer/{id}/history");
    }
}
```

# What is System Integration Testing?

```
@Test
public void testCustomerService(){
    ...
    CustomerService service = new CustomerService(new TestTemplateMock());
    service.findCustomerById(1);
    //some assert
    ...
}
```

A large, stylized green checkmark graphic is positioned in the center of the slide, partially overlapping the code block. It has a thick green outline and a lighter green fill.



# What is System Integration Testing?

```
@RestController("/orders")
public class OrderController{
    ...
    @GetMapping("/order/customer/{id}/order-histories")
    public List<OrderHistory> getCustomerOrderHistory(...){
        ...
        roomService.findOrderHistory(...);
        ...
    }
    ...
}
```

# What is System Integration Testing?

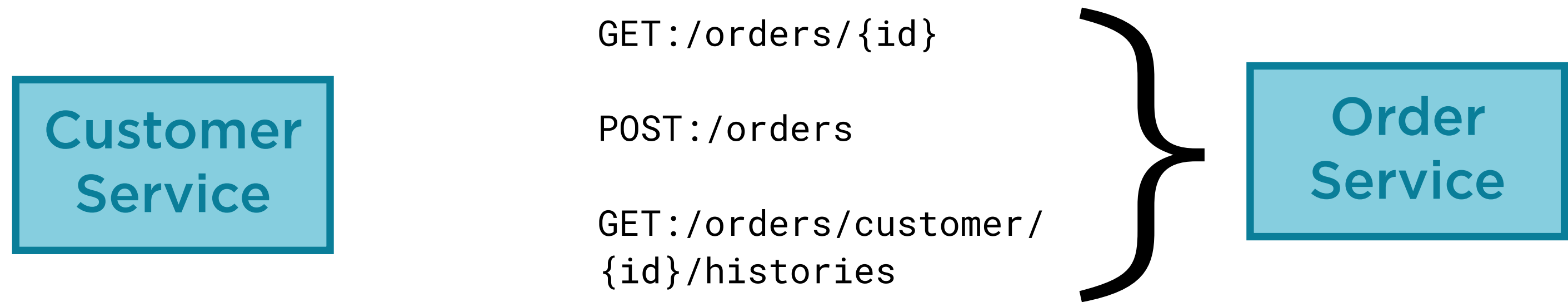
```
@Test
public void testOrderController() {
    ...
    mockMvc.perform(get("/{id}/categories")).andExpect(...);
    ...
}
```

A large, stylized green checkmark graphic is positioned in the center of the slide, partially overlapping the code block. It has a thick green outline and a lighter green fill.



Unit tested  
Not integration tested

# What is System Integration Testing?



Validate contracts

# Validate Contracts

**/api/id**

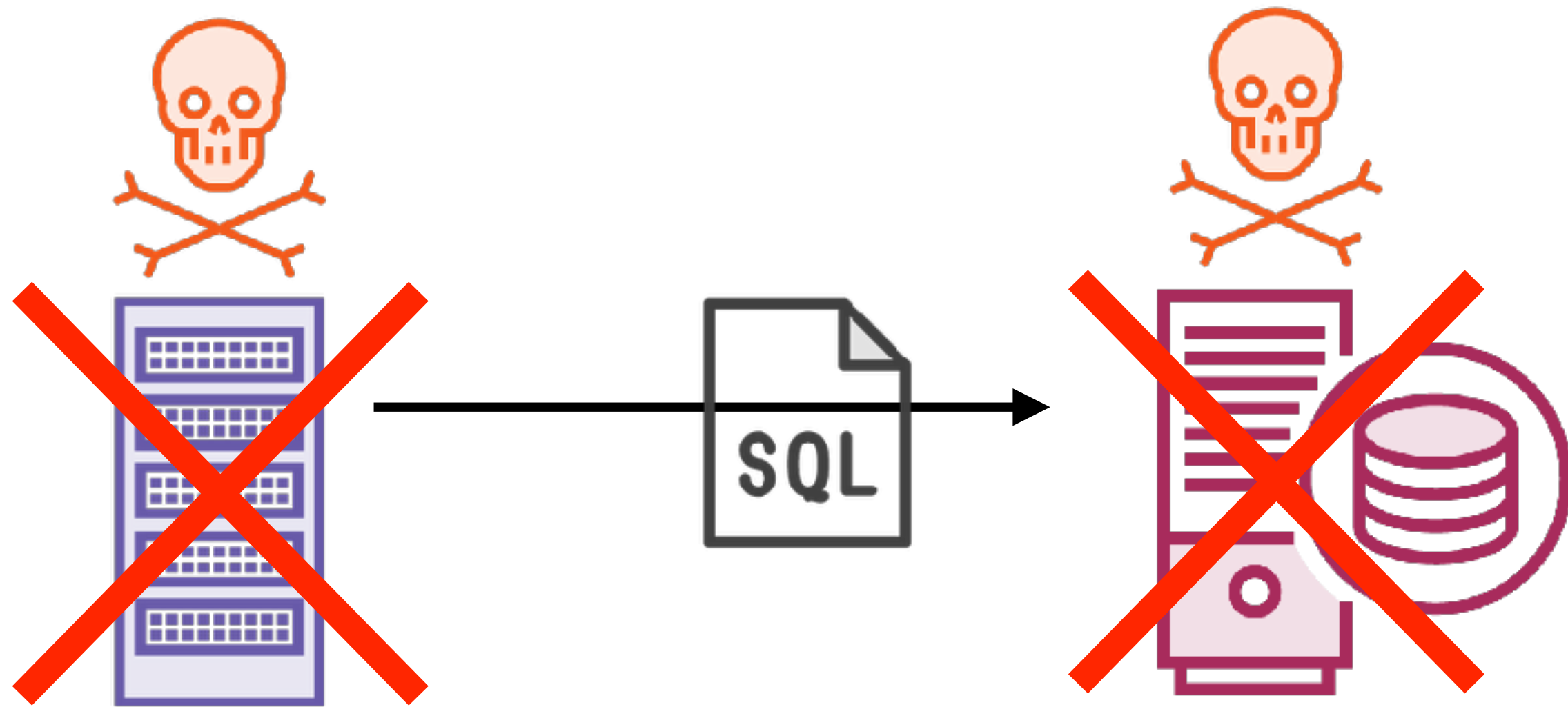
Validate endpoints/addresses



Validate shapes



Validate security requirements



Test failure scenarios



# Goals of System Integration Testing



Validate contracts



Test failure scenarios

# Difficulties with Integration Testing



Time consuming to  
write and run



Unreliable and  
difficult to maintain



# Tools to Help with Integration Testing

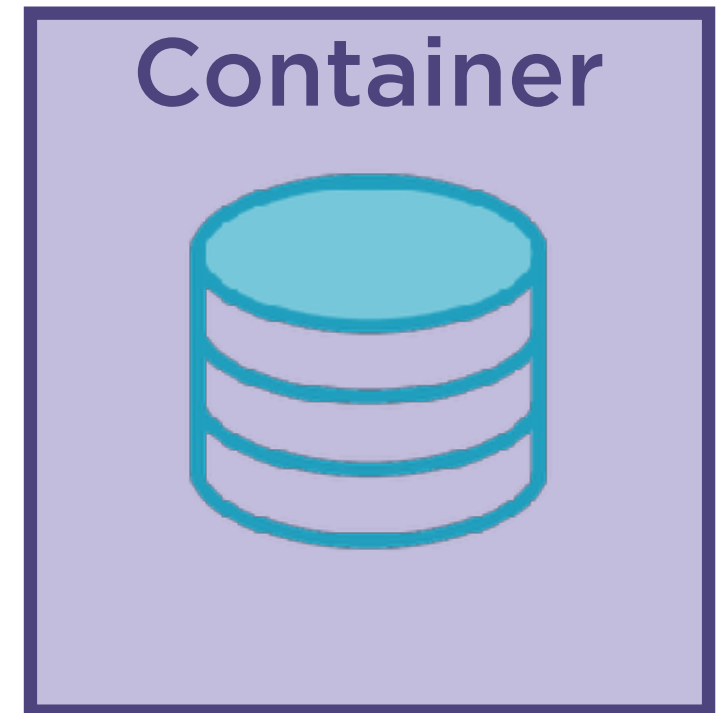
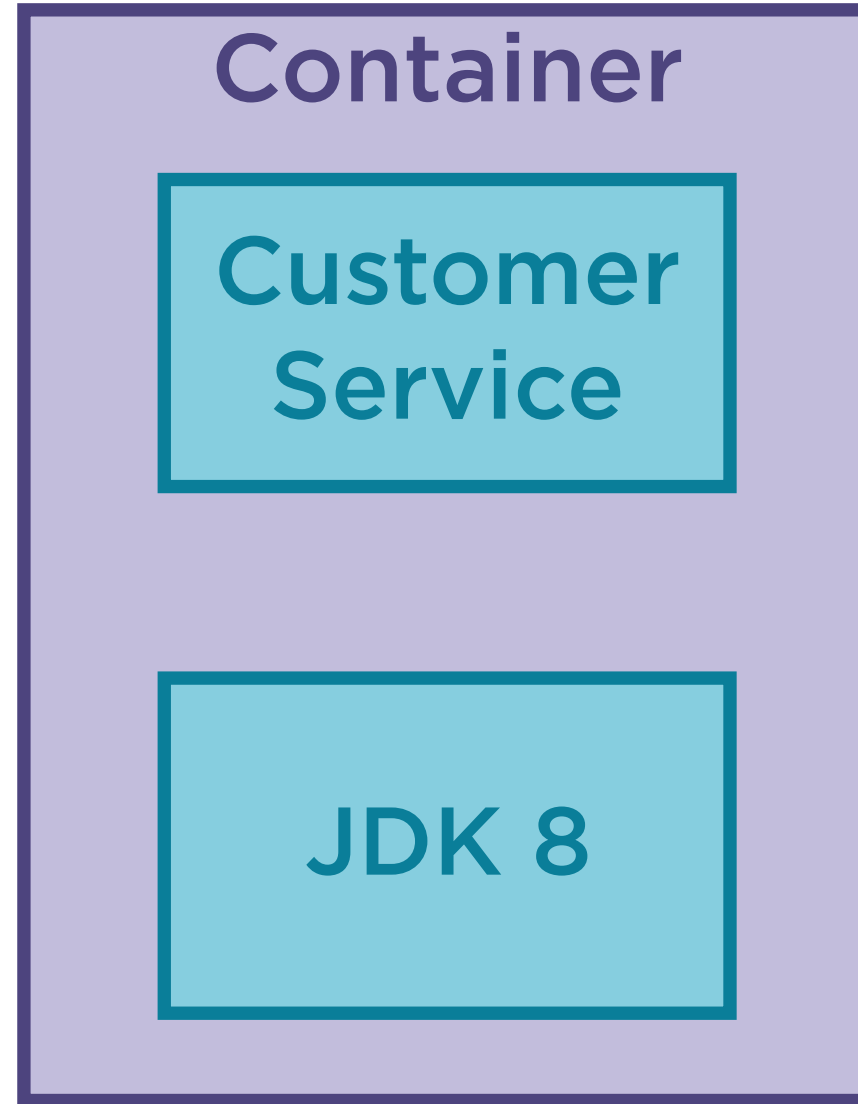
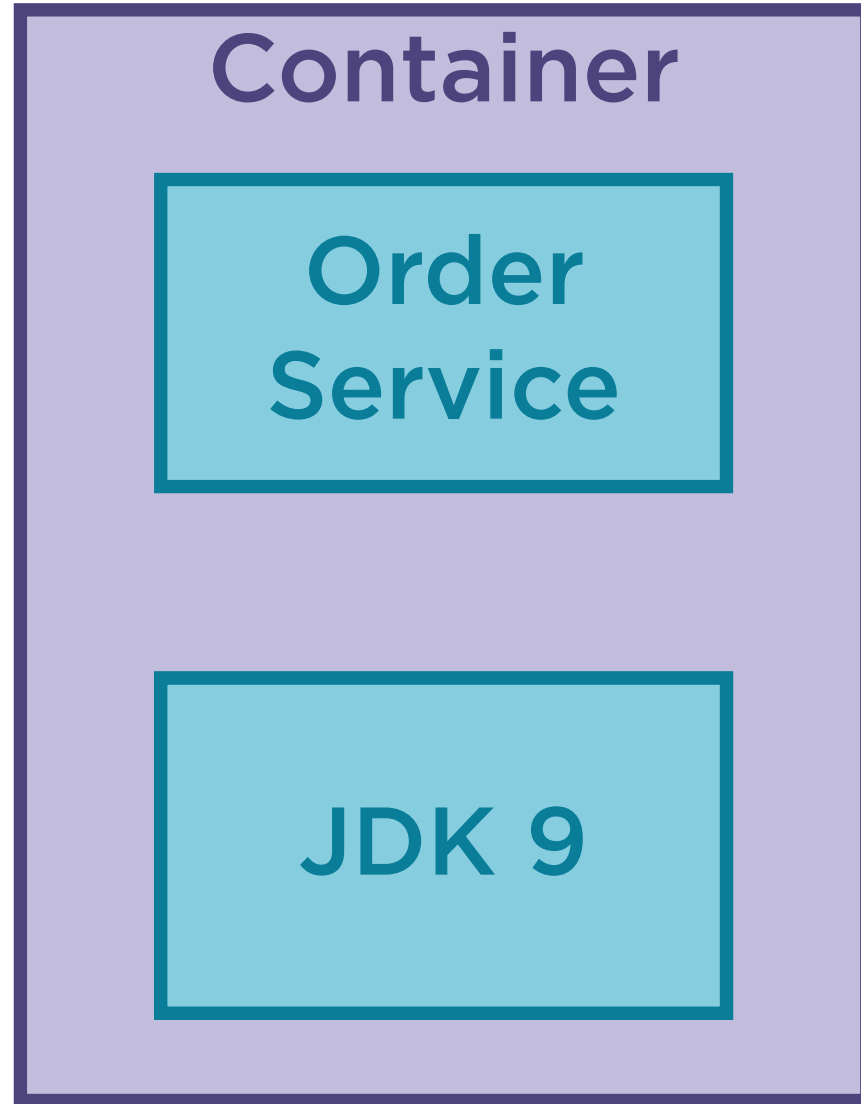


- **Easier to write**
- **More portable**
- **More reliable**
- **Execute faster**
- **More flexible**

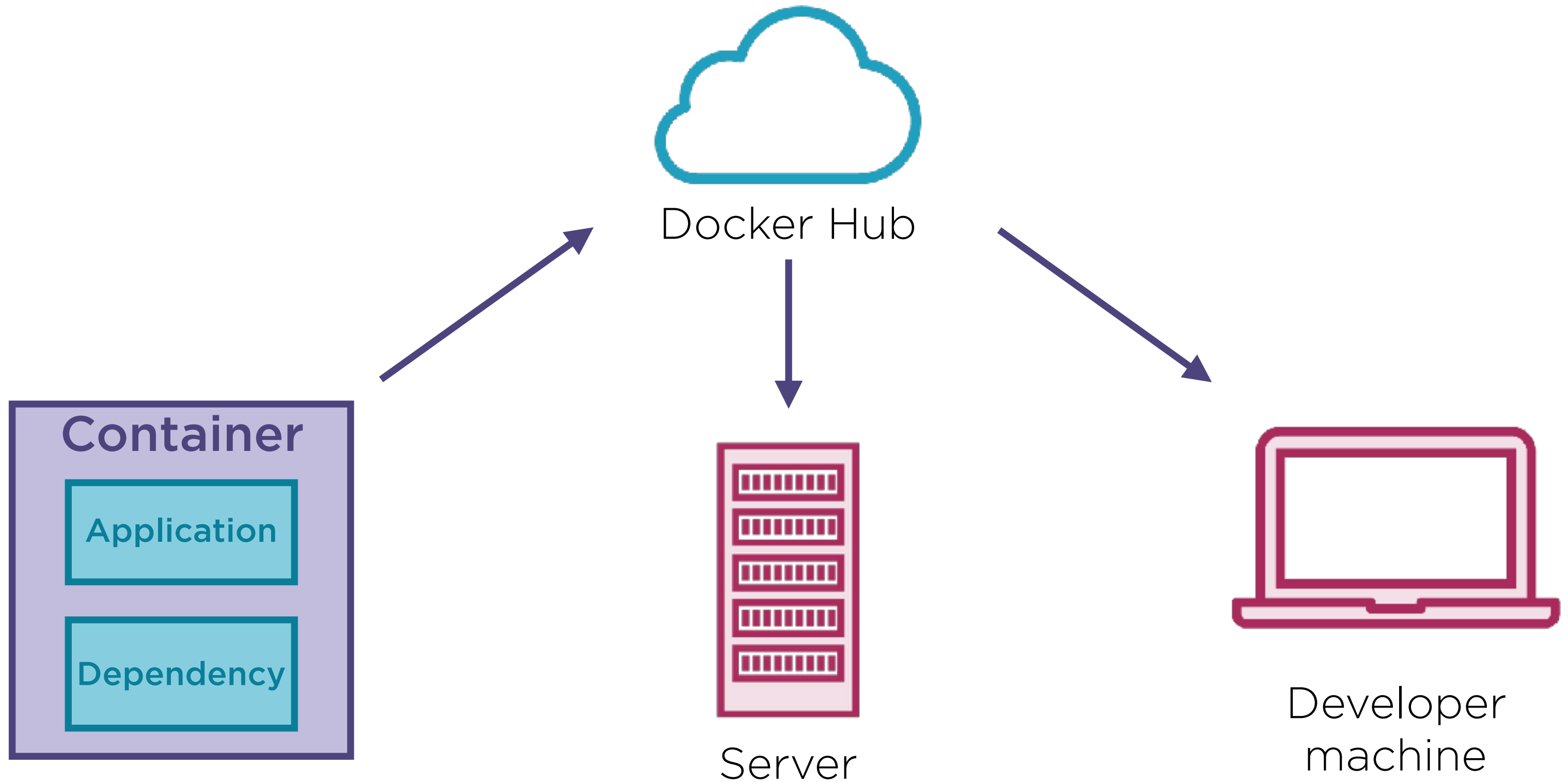
# Introduction to Docker and Test Containers

---

# Introduction to Docker



# Introduction to Docker



```
@RunWith(SpringRunner.class)
@DataJpaTest
public class TestCustomerRepo{

    @Rule
    public static MySQLContainer mySqlDb = new MySQLContainer();
    @Autowired
    private CustomerRepo repo;
    ...
}
```

## Using Test Containers

**Add a `@Rule` or `@ClassRule` and the type of container you want to use**

```
@RunWith(SpringRunner.class)
@DataJpaTest
public class TestCustomerRepo{

    @Rule
    public static GenericContainer someService = new GenericContainer("SomeService");
    @Autowired
    private CustomerRepo repo;
    ...
}
```

## Using Test Containers

**Test Containers also allow for use of arbitrary Docker containers as well.**

# Advantages of Using Spring Cloud Contract



**No test data required**

**No external services required**

**Easier to write tests**

**Tests are more reliable**

# Demo

**Using Test Containers to test a database**

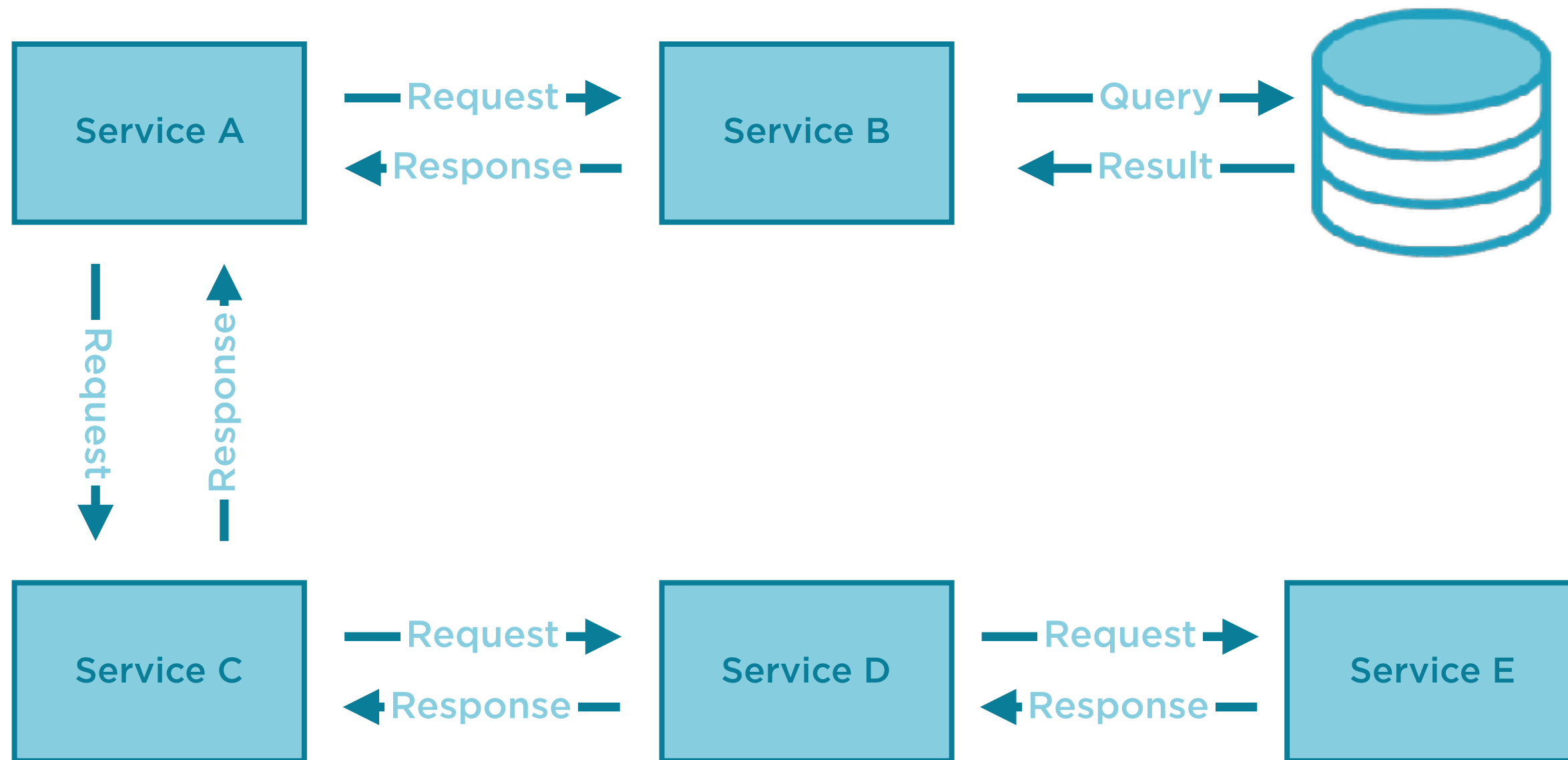
**Reviewing Maven automated testing plugins**



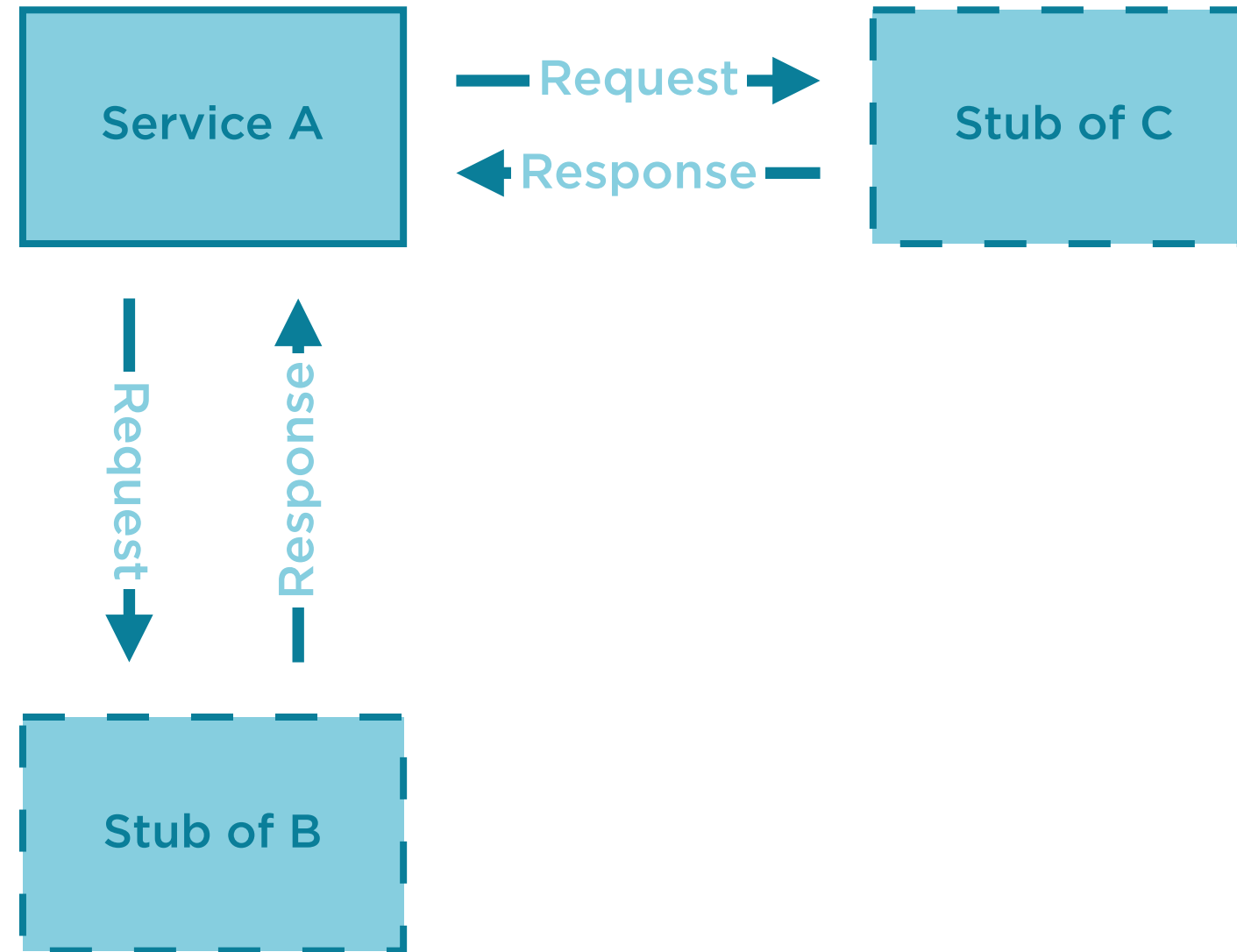
# Spring Cloud Contract

---

# Spring Cloud Contract



# Spring Cloud Contract



```
Contract.make{
  request {
    method 'POST'
    url '/bookings'
    body([
      ...
    ])
    headers {
      contentType('application/json')
    }
  }
  response {
    status 200
    body([
      ...
    ])
    headers {
      contentType('application/json')
    }
  }
}
```

◀ Define a request

◀ Define the response to the request

```

@Test
public void validate_shouldMarkClientAsFraud()
    throws Exception {

    // given:
    MockMvcRequestSpecification request =
        given().header(...).body(...);

    // when:
    ResponseOptions response = given().spec(request)
        .post("/bookings");

    // then:
    assertThat(response.statusCode()).isEqualTo(...);
    assertThat(response.header(...));

}

```

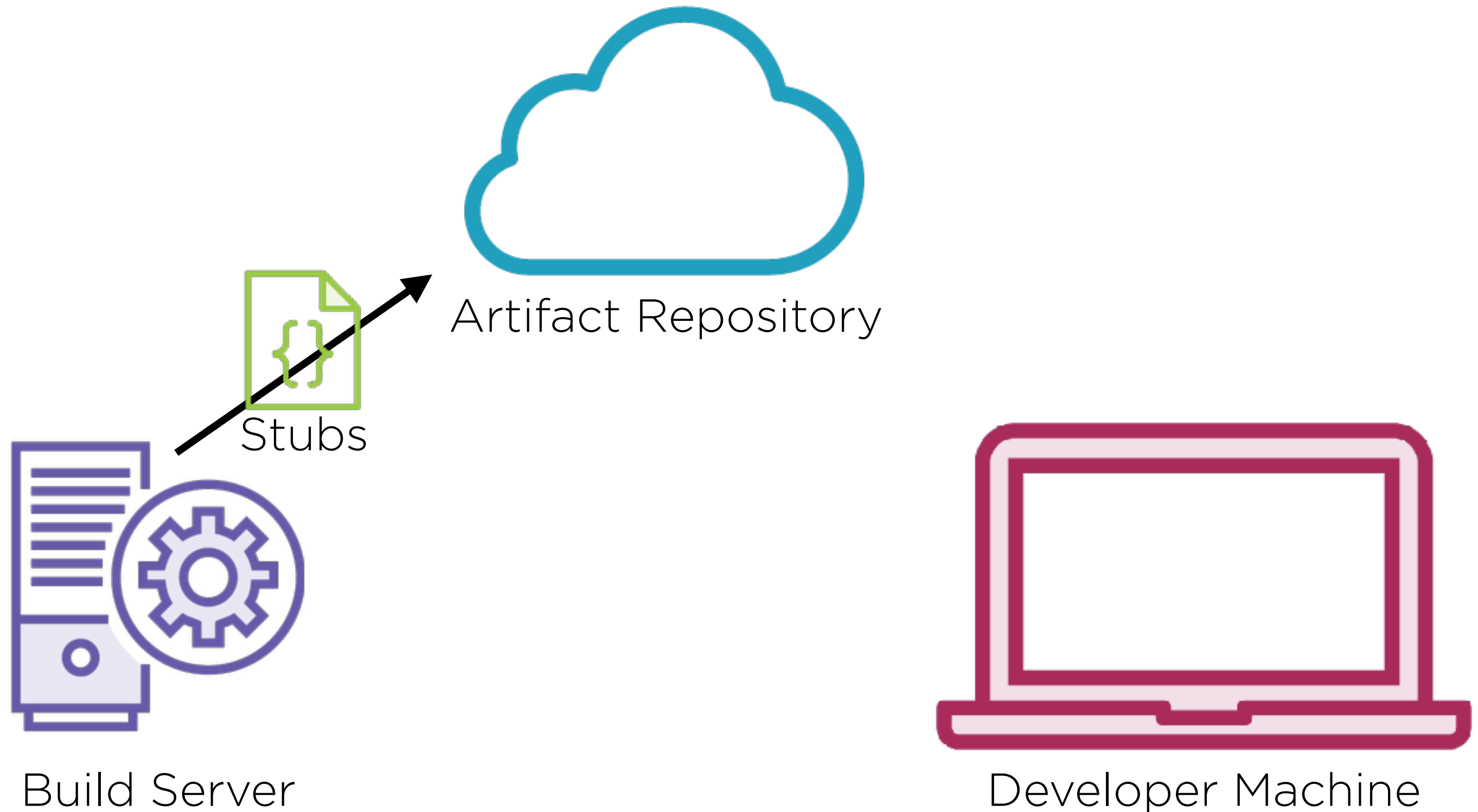
◀ Auto generated stubs to test producer code

◀ Set up expectations

◀ Execute call to endpoint

◀ Validate response

# Spring Cloud Contract



```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment=WebEnvironment.NONE)
@AutoConfigureStubRunner
(ids = {"com.ps.bk.hotel:booking-service:+:8081"})
@DirtiesContext
public class ValidateBookingsContract {
    @Test
    public void testSuccessfulBookingCall(){
        ...
    }
}
```

- ◀ Reference stubs artifact built by Spring Cloud Contract
- ◀ Write a test that calls the stubs

# Integration Testing with Spring Cloud Contract

---



Demo

**Writing a REST call with Spring Cloud  
Contract**

# Summary

**Goals of system integration testing**

**Using tools to make integration testing easier, faster, and more reliable**