MICRO SERVICE WARS

JUnit – Episode 5

TestContainers strikes back

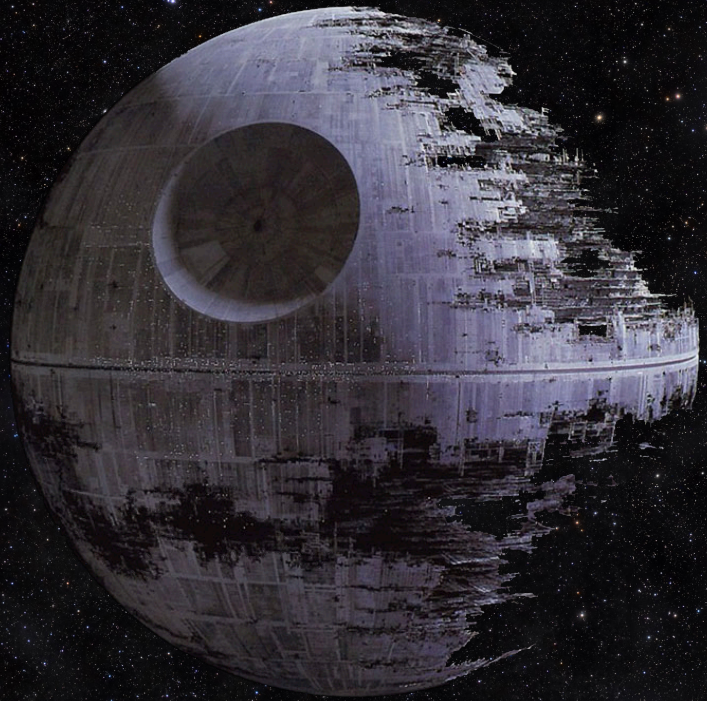
Коровин Анатолий

antkorwin

antkorwin

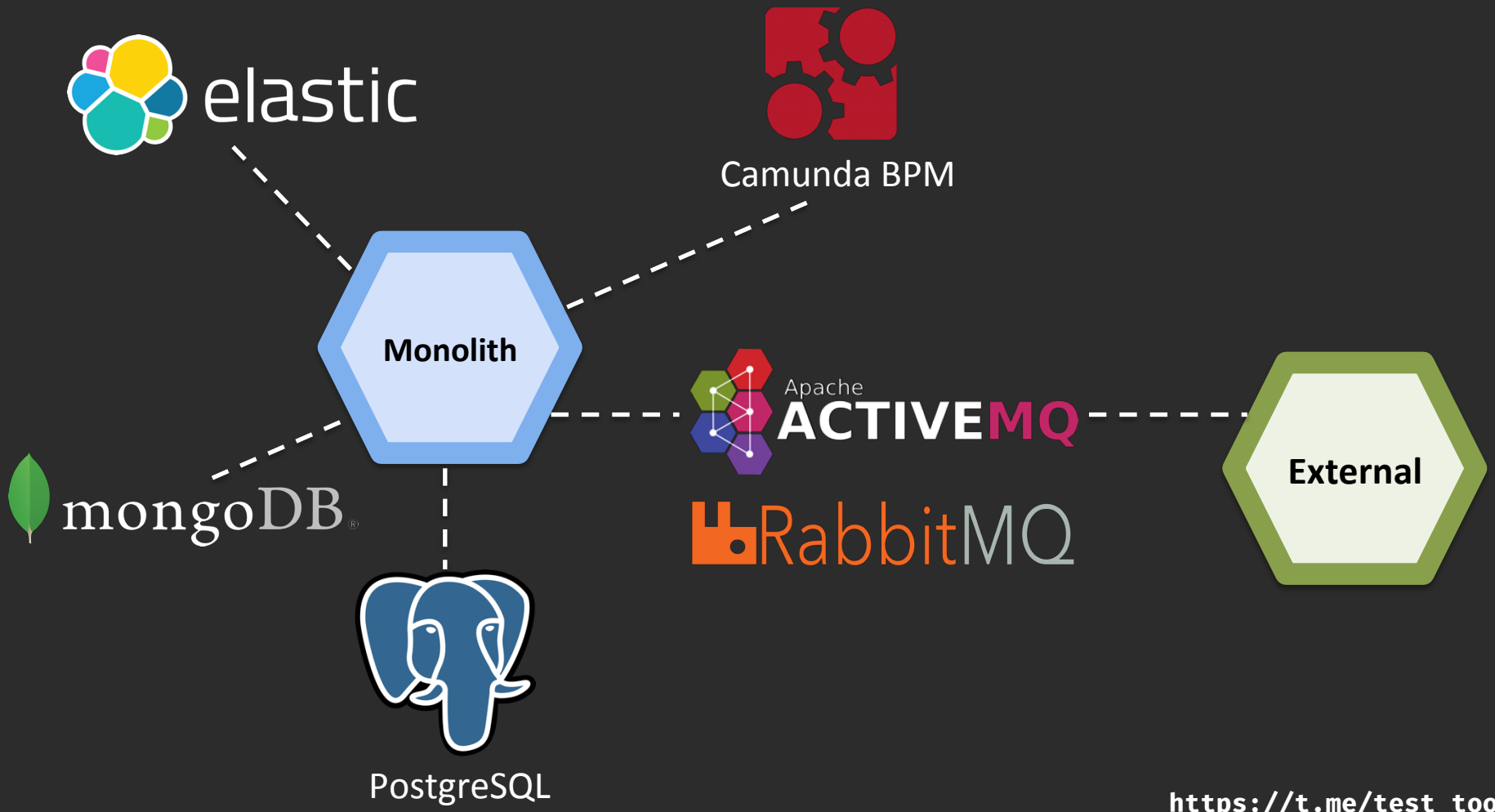
antkorwin.com

t.me/test_tools



Приходишь

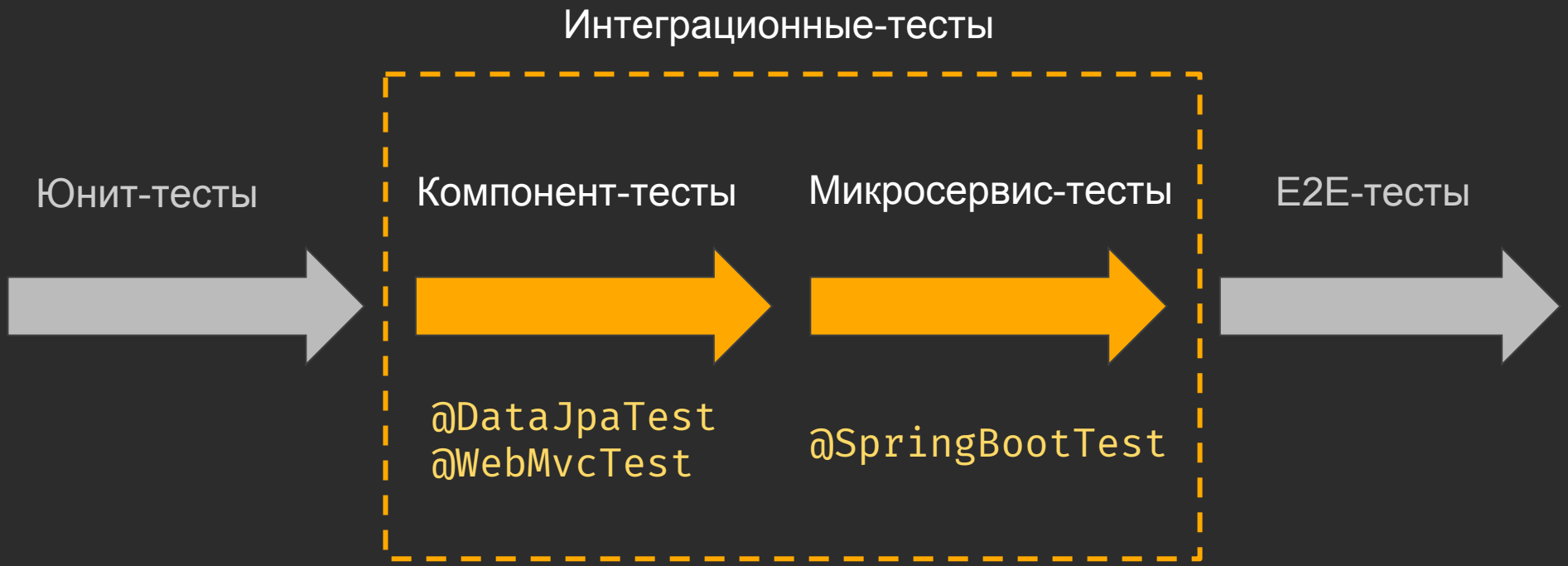
на новую работу




План
- что мы хотим тестировать
- как тестировали монолит
- тесты в микросервисах
- проблемы сопровождения
- как использовать docker
- что может пойти не так


План
- что мы хотим тестировать
- как тестировали монолит
- тесты в микросервисах
- проблемы сопровождения
- как использовать docker
- что может пойти не так

// что мы хотим тестировать



Не видно взгляду,
что ты тестировать
именно хочешь

// что мы хотим тестировать

хранимые процедуры

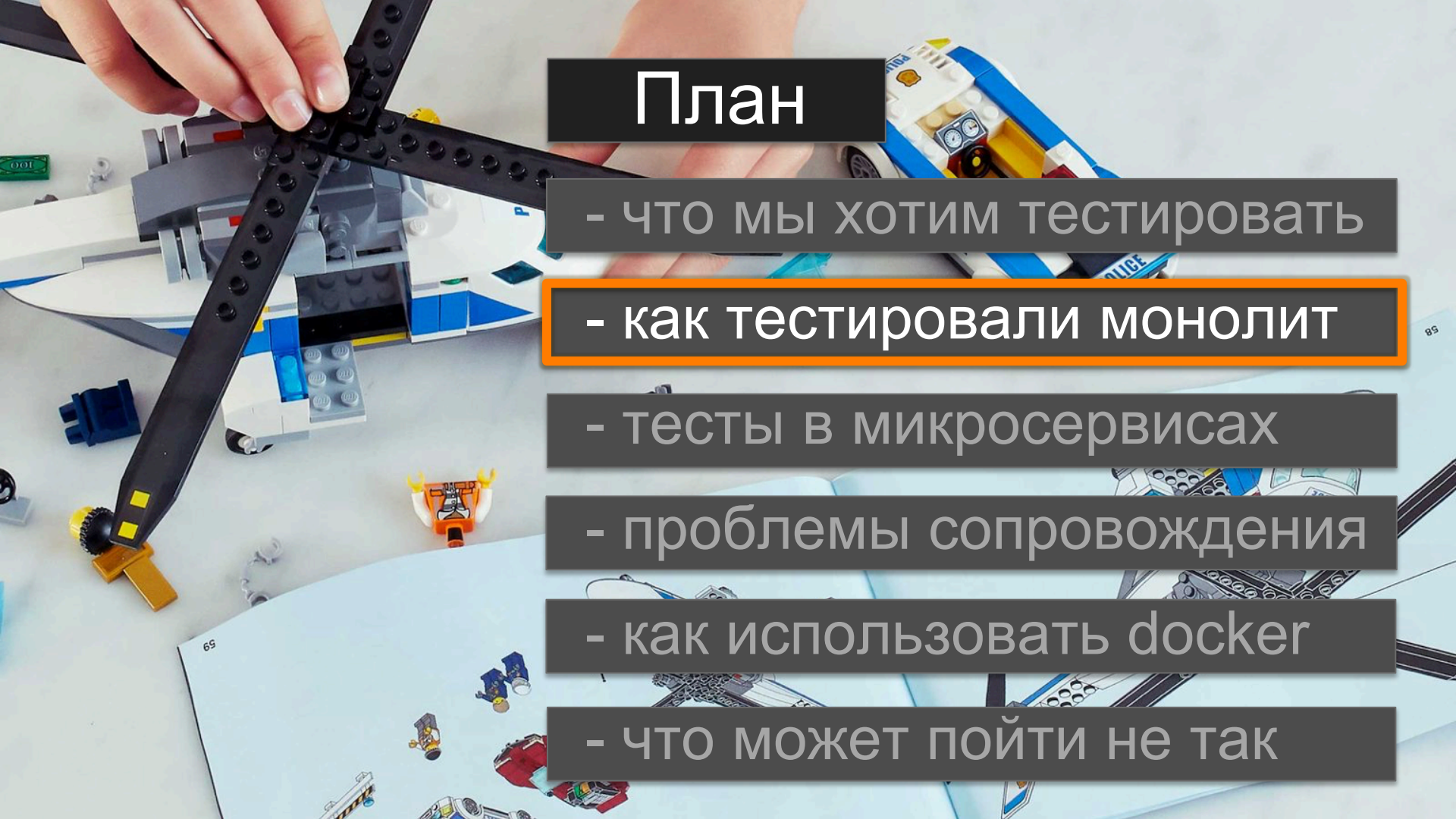
валидация дата-сетов

нативные запросы

асинхронные события

расширения postgres

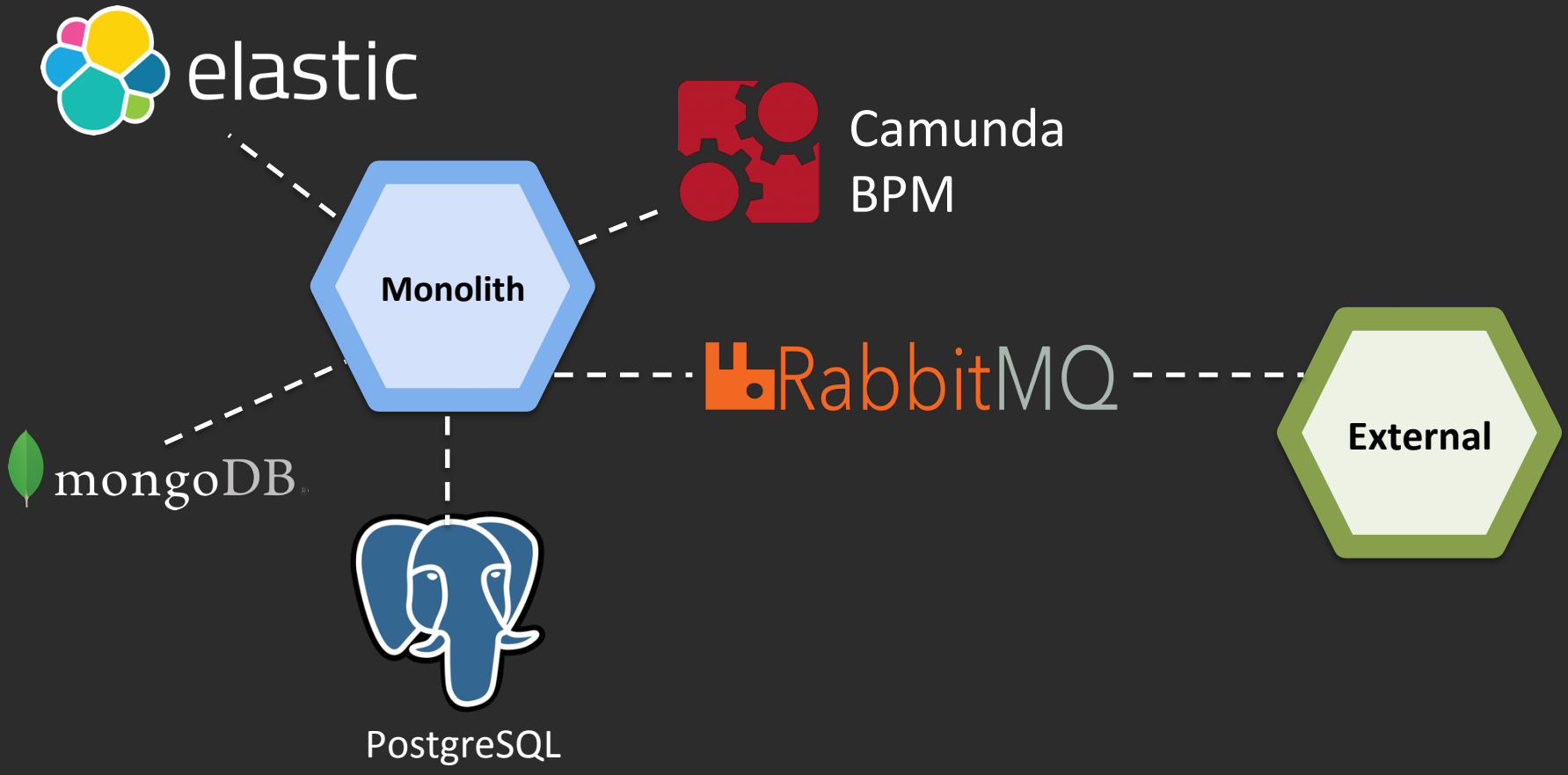
конвертацию данных



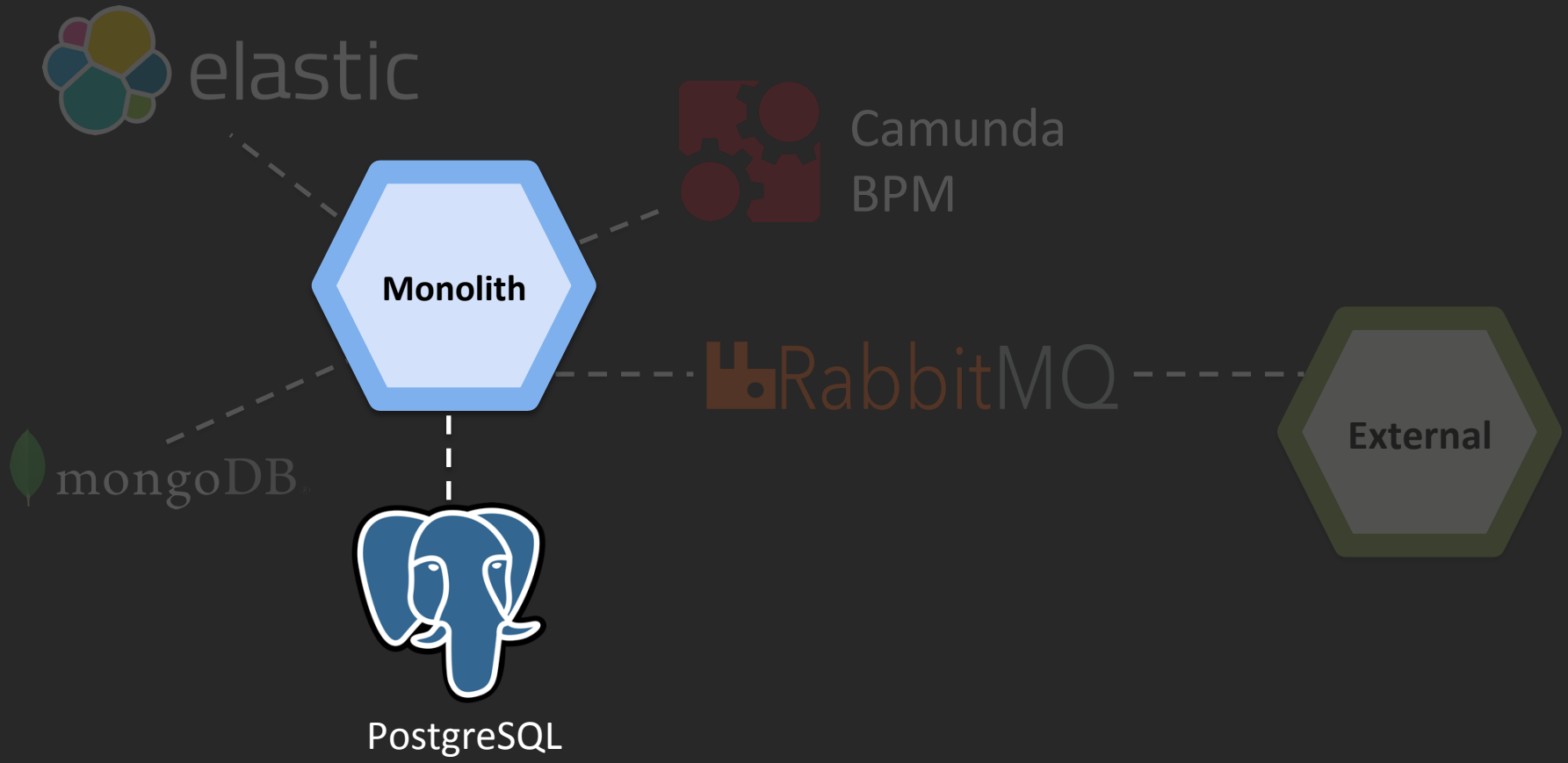
План

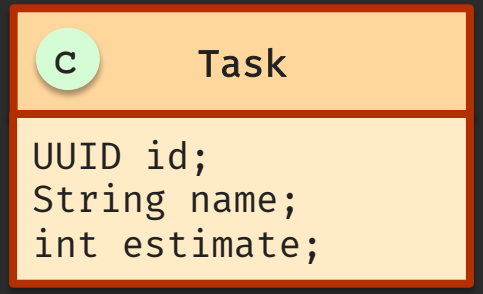
- что мы хотим тестировать
- как тестировали монолит
- тесты в микросервисах
- проблемы сопровождения
- как использовать docker
- что может пойти не так

// как мы тестировали монолит



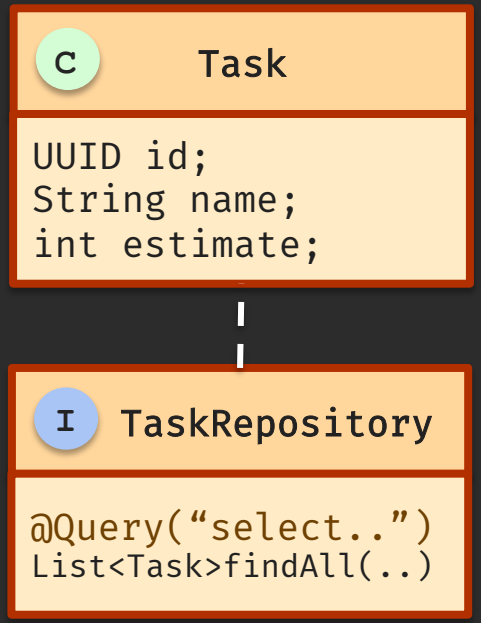
// тестируем работу с БД



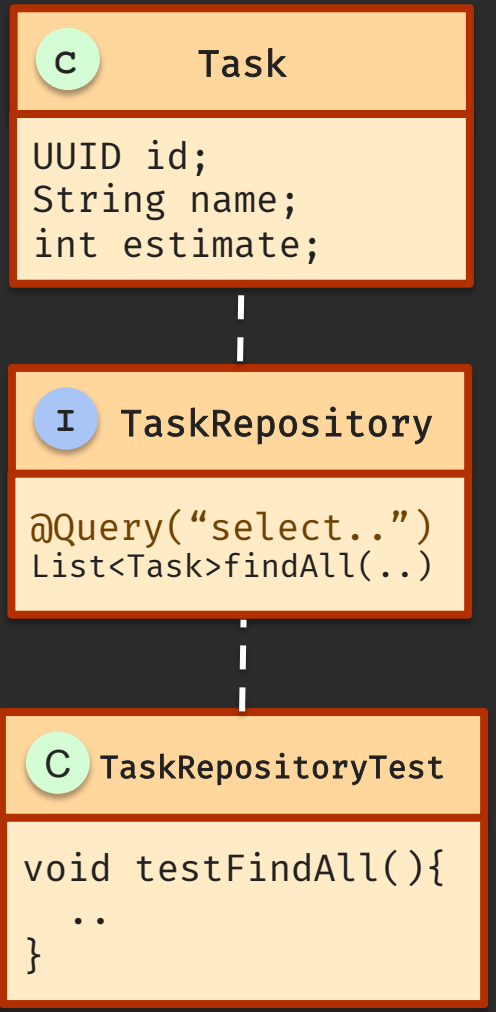


// тестируем работу с БД

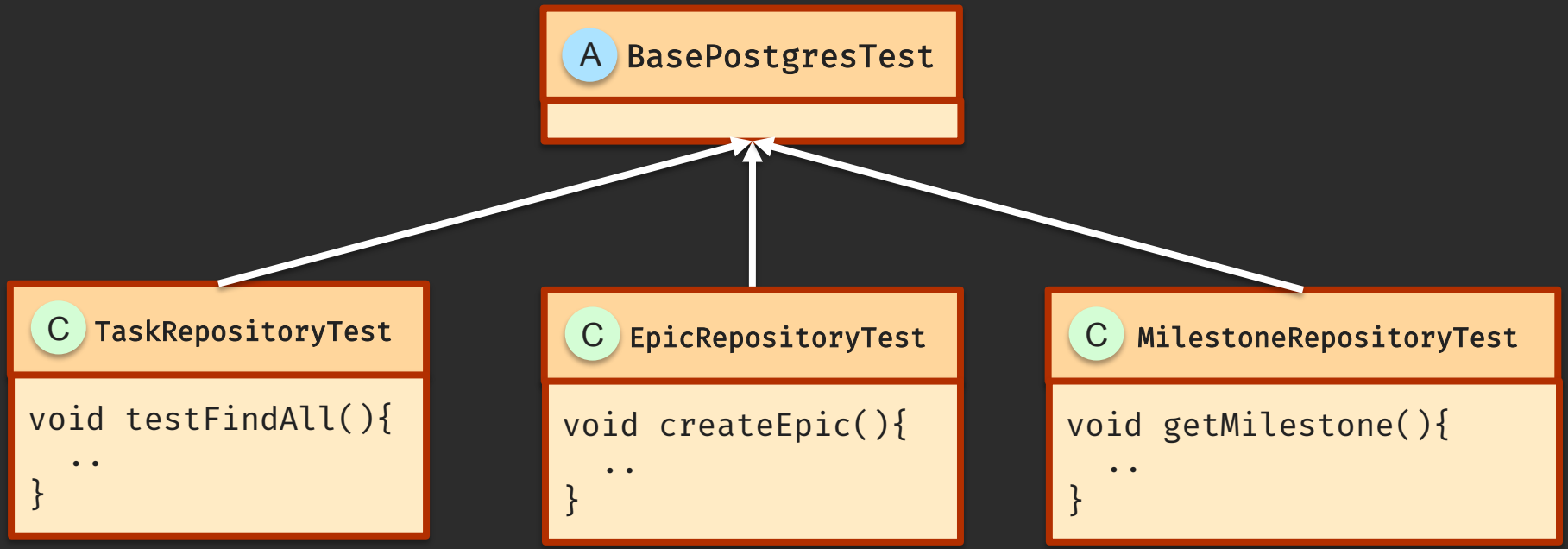
// тестируем работу с БД



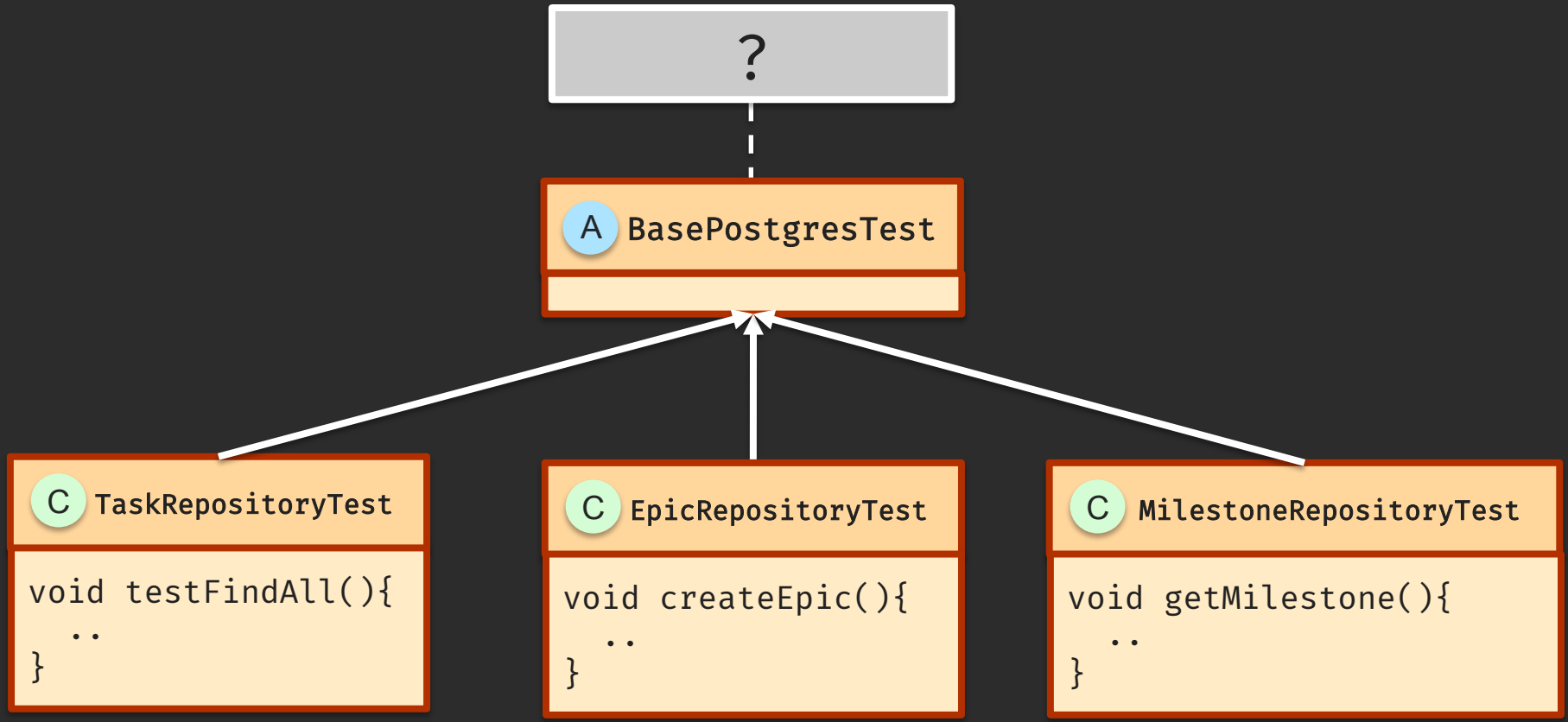
// тестируем работу с БД



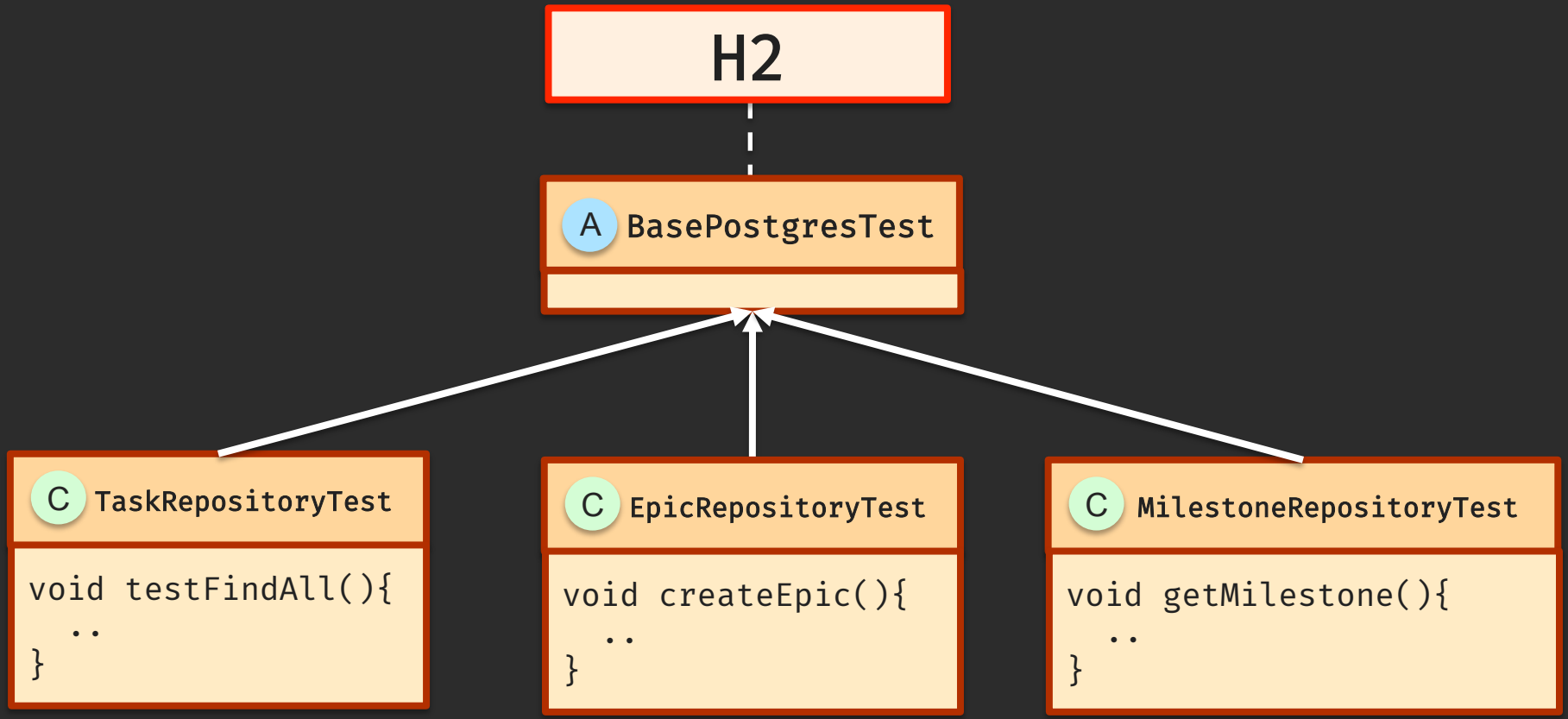
// выносим конфигурацию в базовый класс



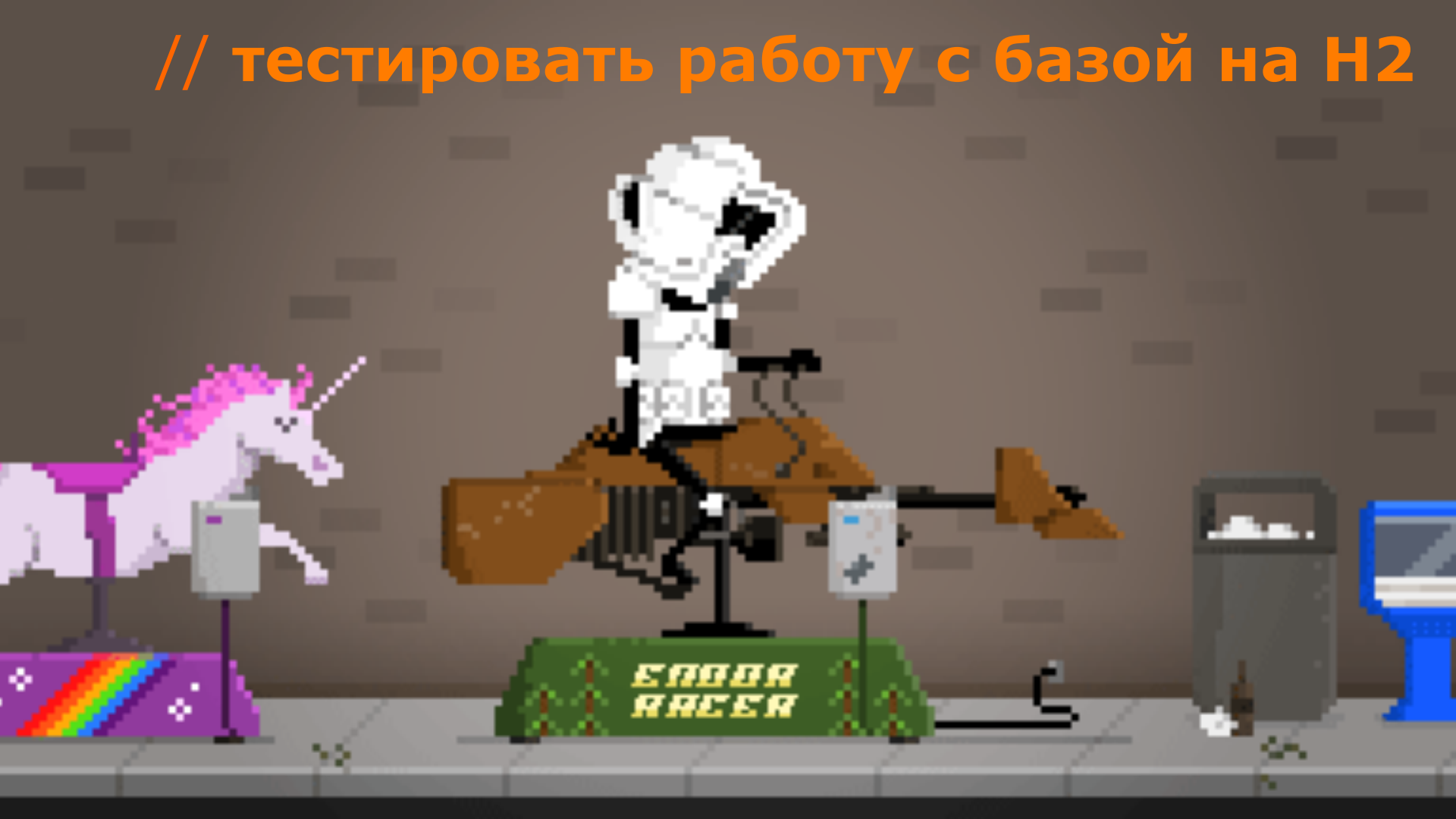
// а на какой базе тестируем?

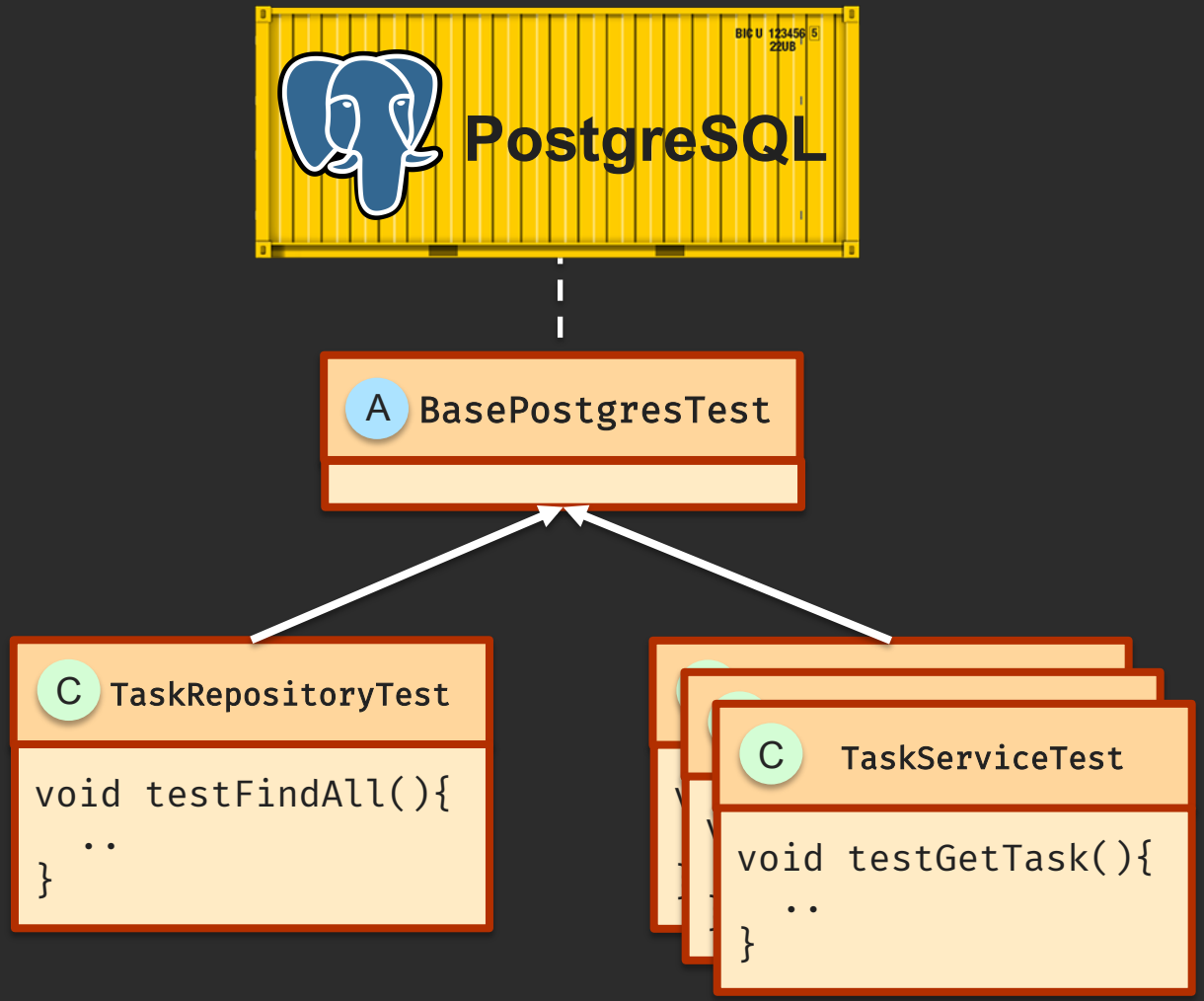


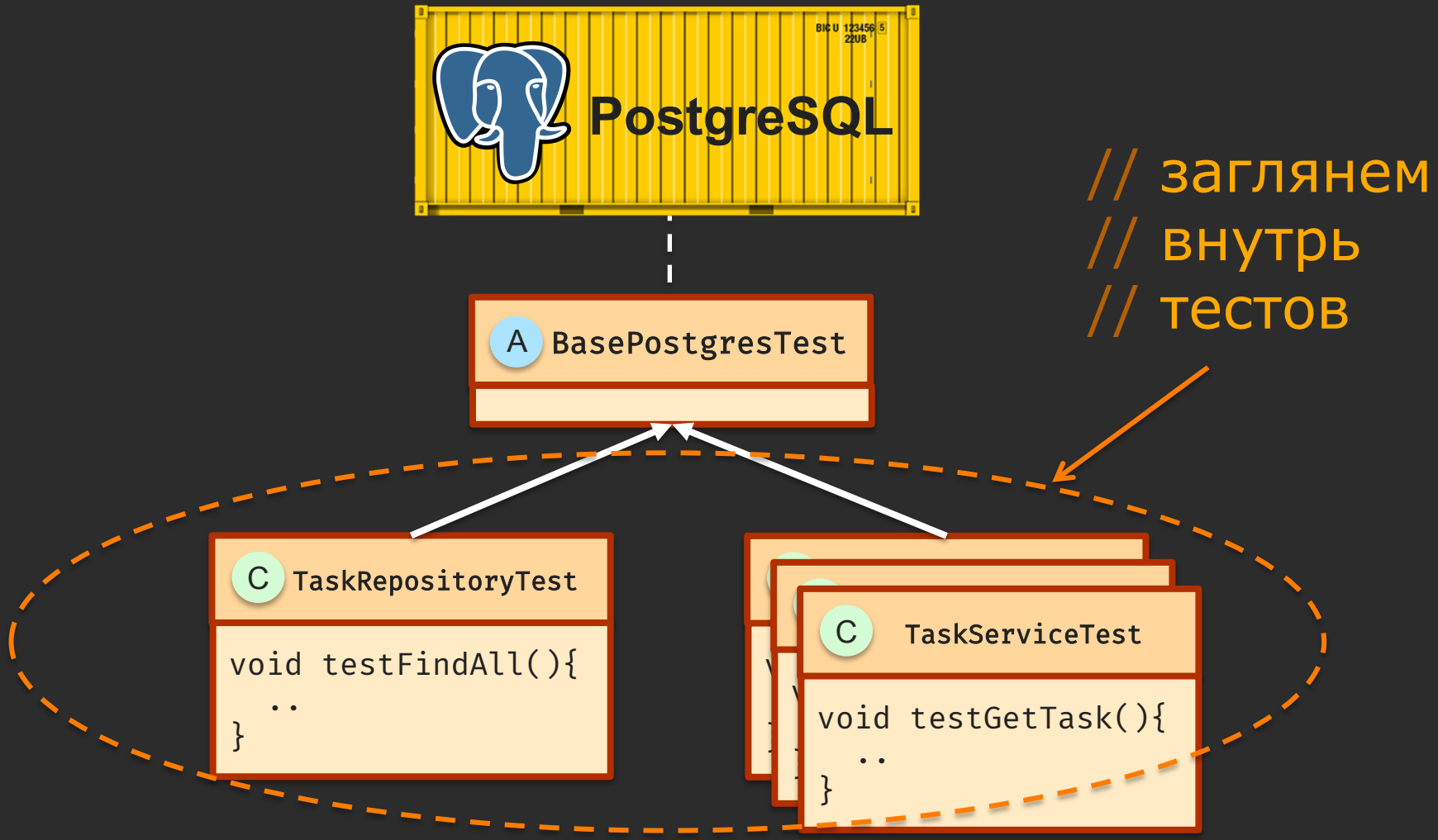
// а на какой базе тестируем?



// тестировать работу с базой на H2




PostgreSQL
A BasePostgresTest
C TaskRepositoryTest
void testFindAll(){
..
}
C TaskServiceTest
void testGetTask(){
..
}


PostgreSQL
// заглянем
// внутрь
// тестов
A BasePostgresTest
C TaskRepositoryTest
void testFindAll(){
..
}
C TaskServiceTest
void testGetTask(){
..
}

```
public class TaskServiceTest extends BasePostgresTest {

      @Test
      @DataSet(value = "datasets/task.json")
      public void getTask() {

            Task task = taskService.get(TASK_ID);
            assertThat(task).isNotNull();
      }

}
```

```
public class TaskServiceTest extends BasePostgresTest {

      @Test
      @DataSet(value = "datasets/task.json”)
      public void getTask() {

            Task task = taskService.get(TASK_ID);
            assertThat(task).isNotNull();
      }

}
```

```
public class TaskServiceTest extends BasePostgresTest {

      @Test
      @DataSet(value = "datasets/task.json")
      public void getTask() {

            Task task = taskService.get(TASK_ID);
            assertThat(task).isNotNull();
      }

}
```

```
public class TaskServiceTest extends BasePostgresTest {

      @Test
      @DataSet(value = "datasets/task.json")
      public void getTask() {

            Task task = taskService.get(TASK_ID);
            assertThat(task).isNotNull();
      }

}
```

// а что внутри BasePostgresTest?

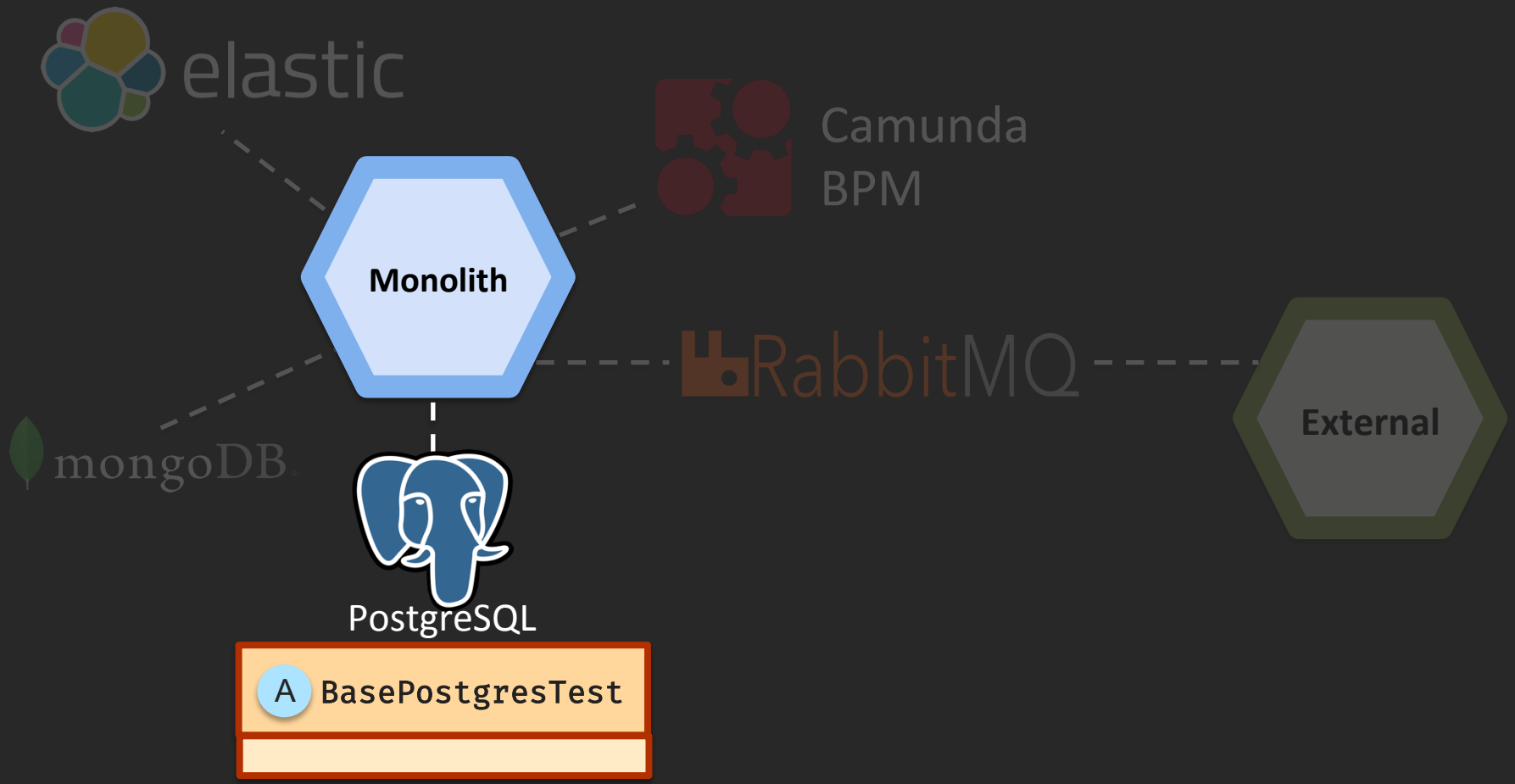
```
public class TaskServiceTest extends BasePostgresTest {

      @Test
      @DataSet(value = "datasets/task.json")
      public void getTask() {

            Task task = taskService.get(TASK_ID);
            assertThat(task).isNotNull();
      }

}
```

```
@RunWith(SpringRunner.class)
@DataJpaTest
public abstract class BasePostgresTest {

  @Rule
  public DBUnitRule dbUnitRule =
                        DBUnitRule.instance(() -> {
        ...
  });

  @Rule
  public PostgreSQLContainer postgres =
                        new PostgreSQLContainer();

}
```

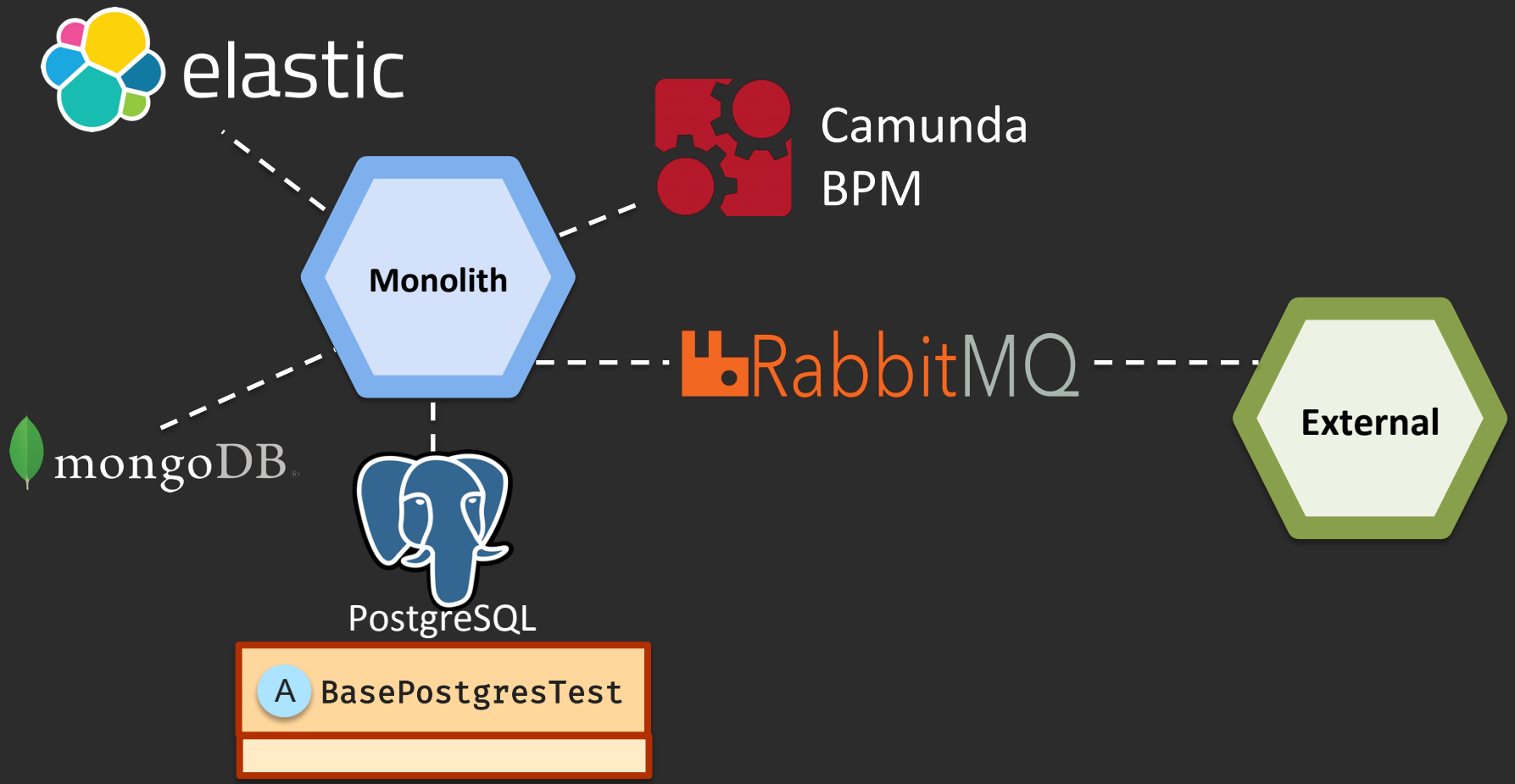
```
@RunWith(SpringRunner.class)
@DataJpaTest
public abstract class BasePostgresTest {

  @Rule
  public DBUnitRule dbUnitRule =
                        DBUnitRule.instance(() -> {
        ...
  });

  @Rule
  public PostgreSQLContainer postgres =
                        new PostgreSQLContainer();

}
```

```
@RunWith(SpringRunner.class)
@DataJpaTest
public abstract class BasePostgresTest {

  @Rule
  public DBUnitRule dbUnitRule =
                        DBUnitRule.instance(() -> {
        ...
  });

  @Rule
  public PostgreSQLContainer postgres =
                        new PostgreSQLContainer();

}
```

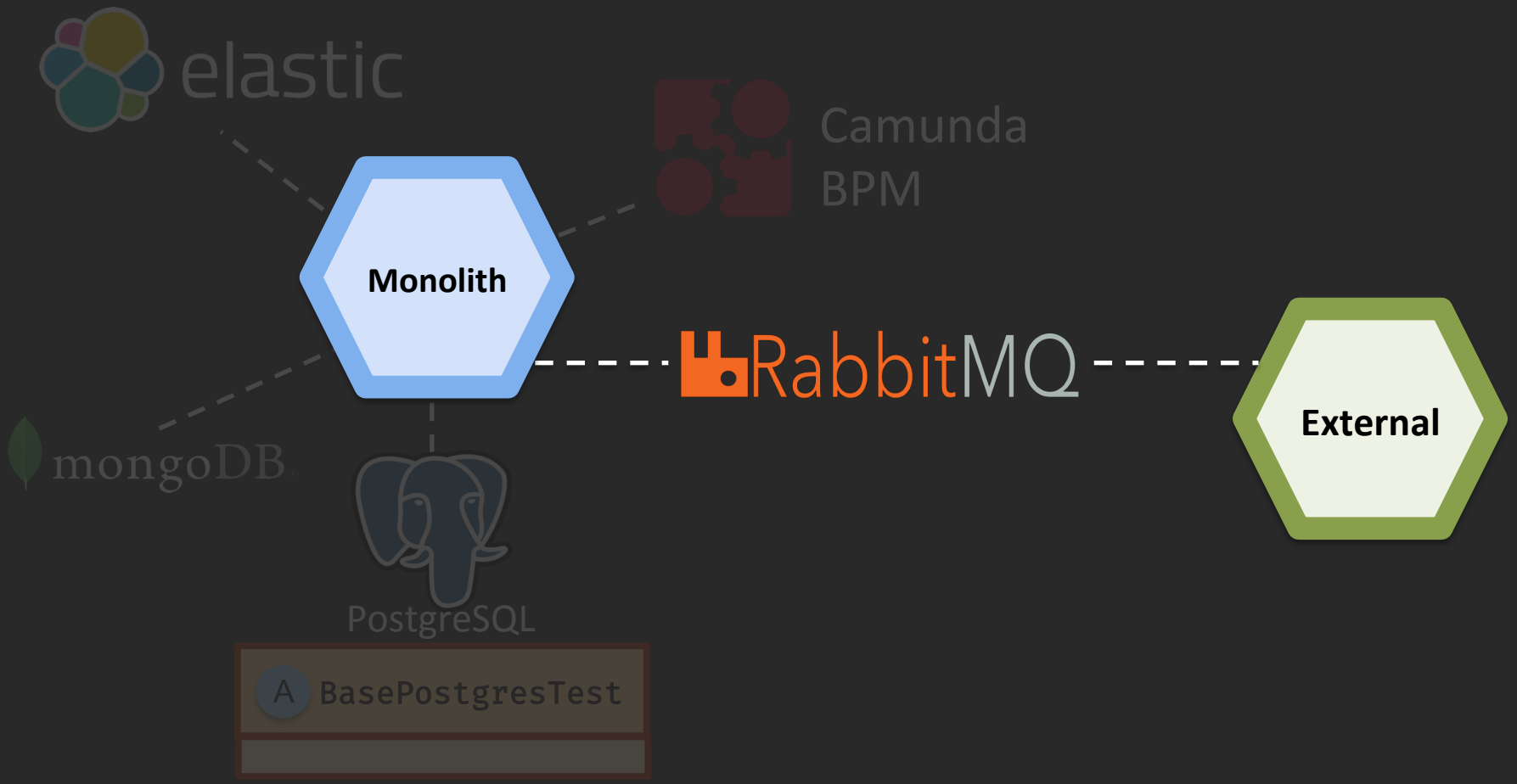
// тестируем работу с БД

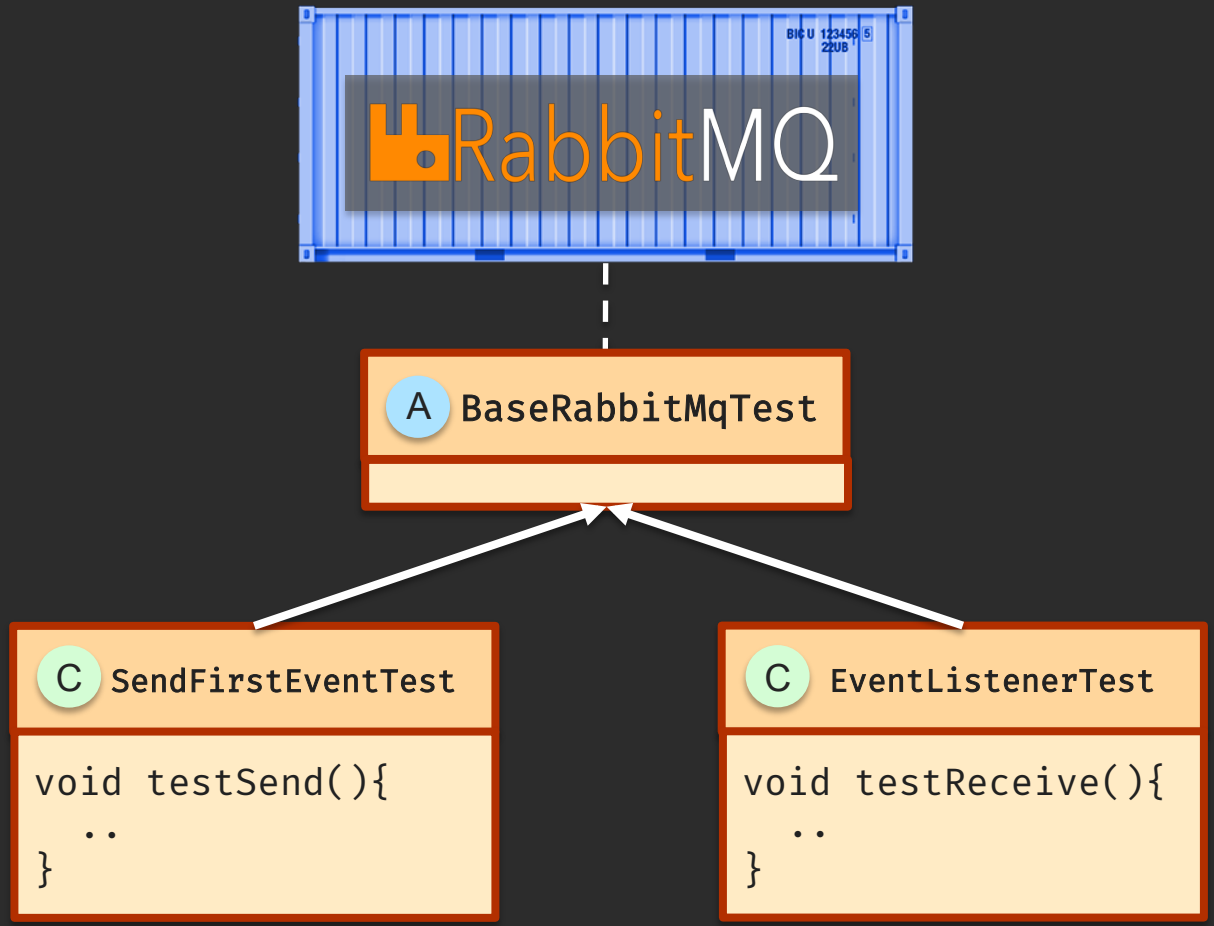


// тестирование монолита

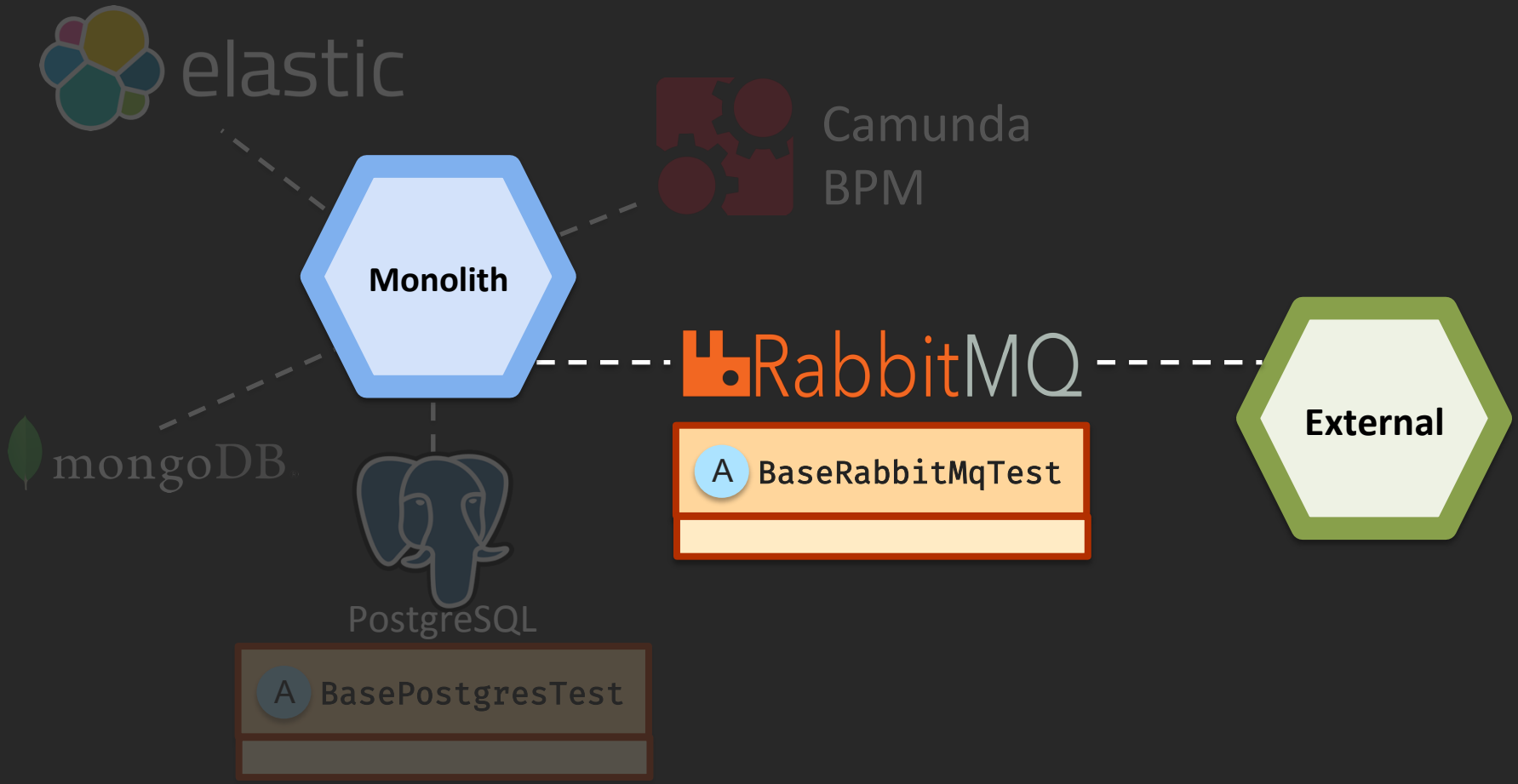


// тестируем отправку сообщений

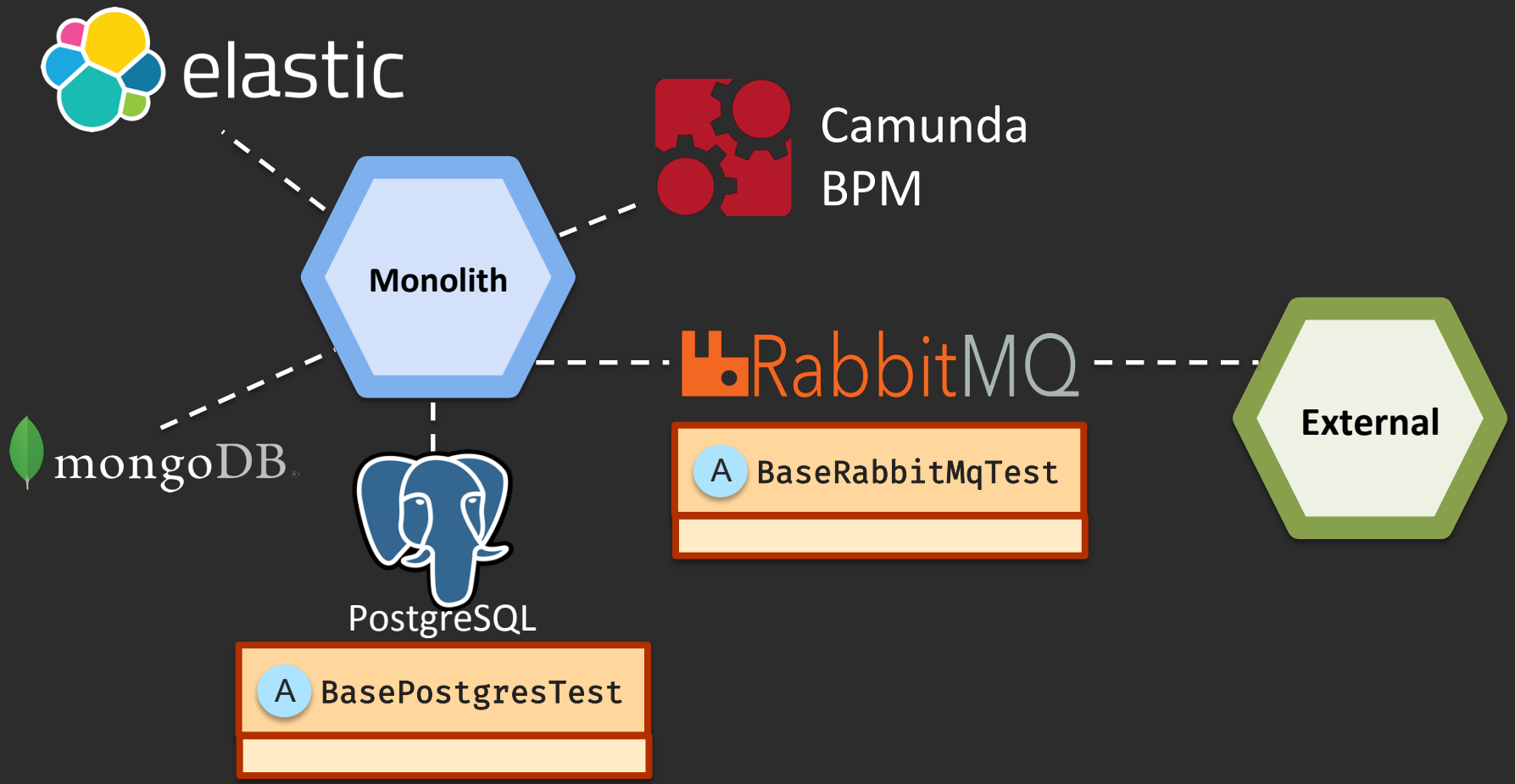



RabbitMQ
A BaseRabbitMqTest
C SendFirstEventTest
void testSend(){
..
}
C EventListenerTest
void testReceive(){
..
}

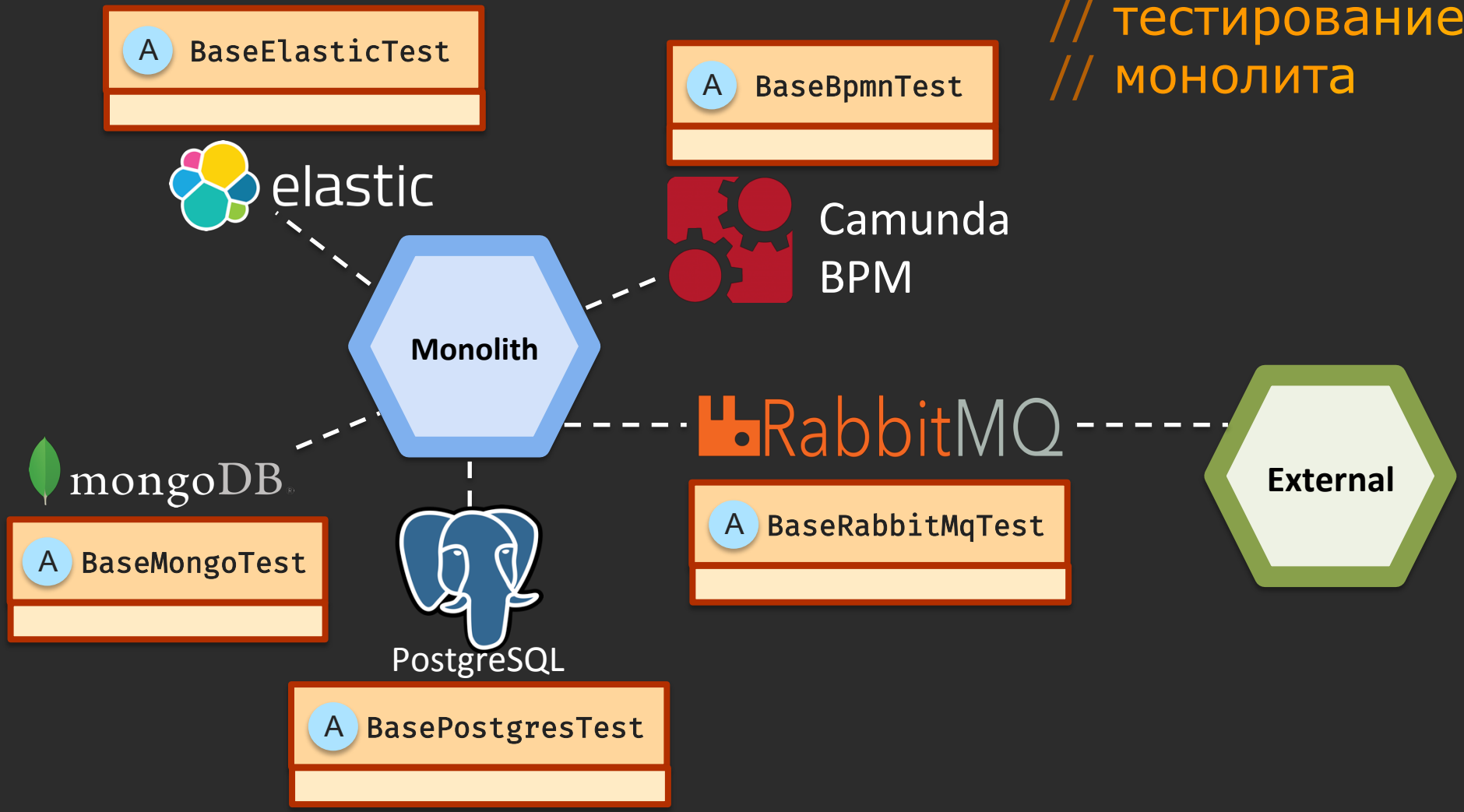
// тестируем отправку сообщений

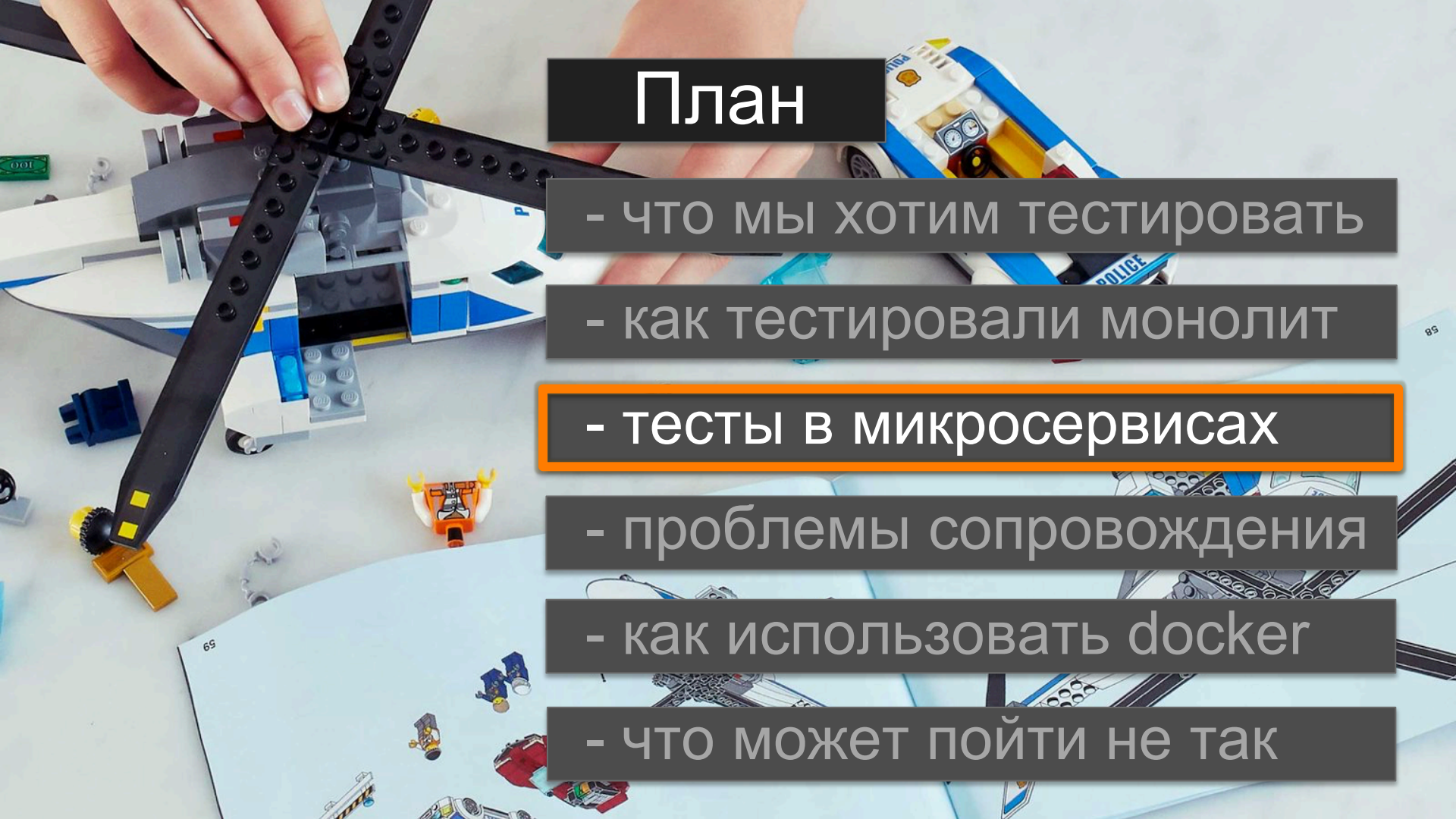


// тестирование монолита



// тестирование
// монолита



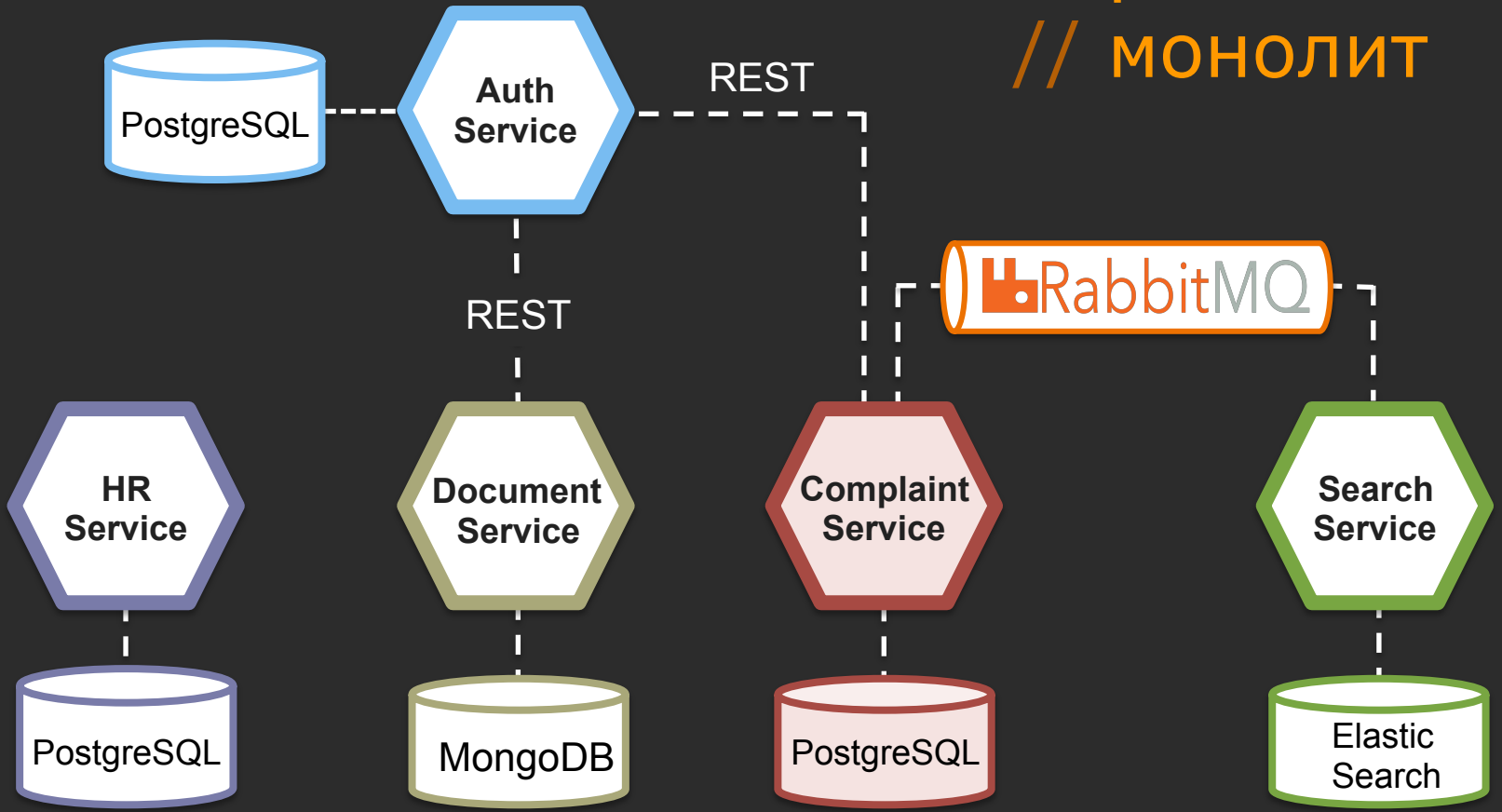


План

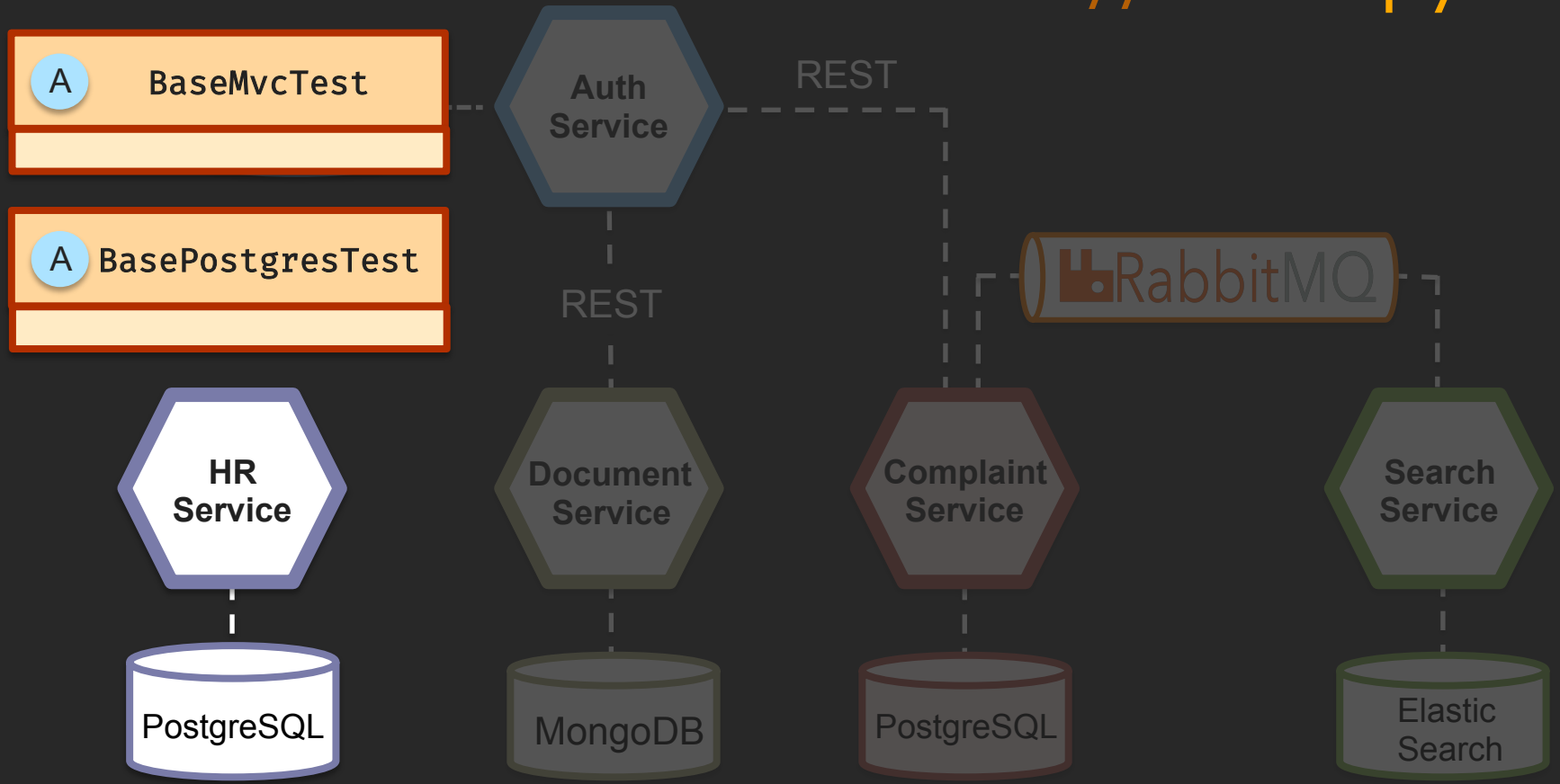
- что мы хотим тестировать
- как тестировали монолит
- тесты в микросервисах
- проблемы сопровождения
- как использовать docker
- что может пойти не так

// распиливаем
// монолит

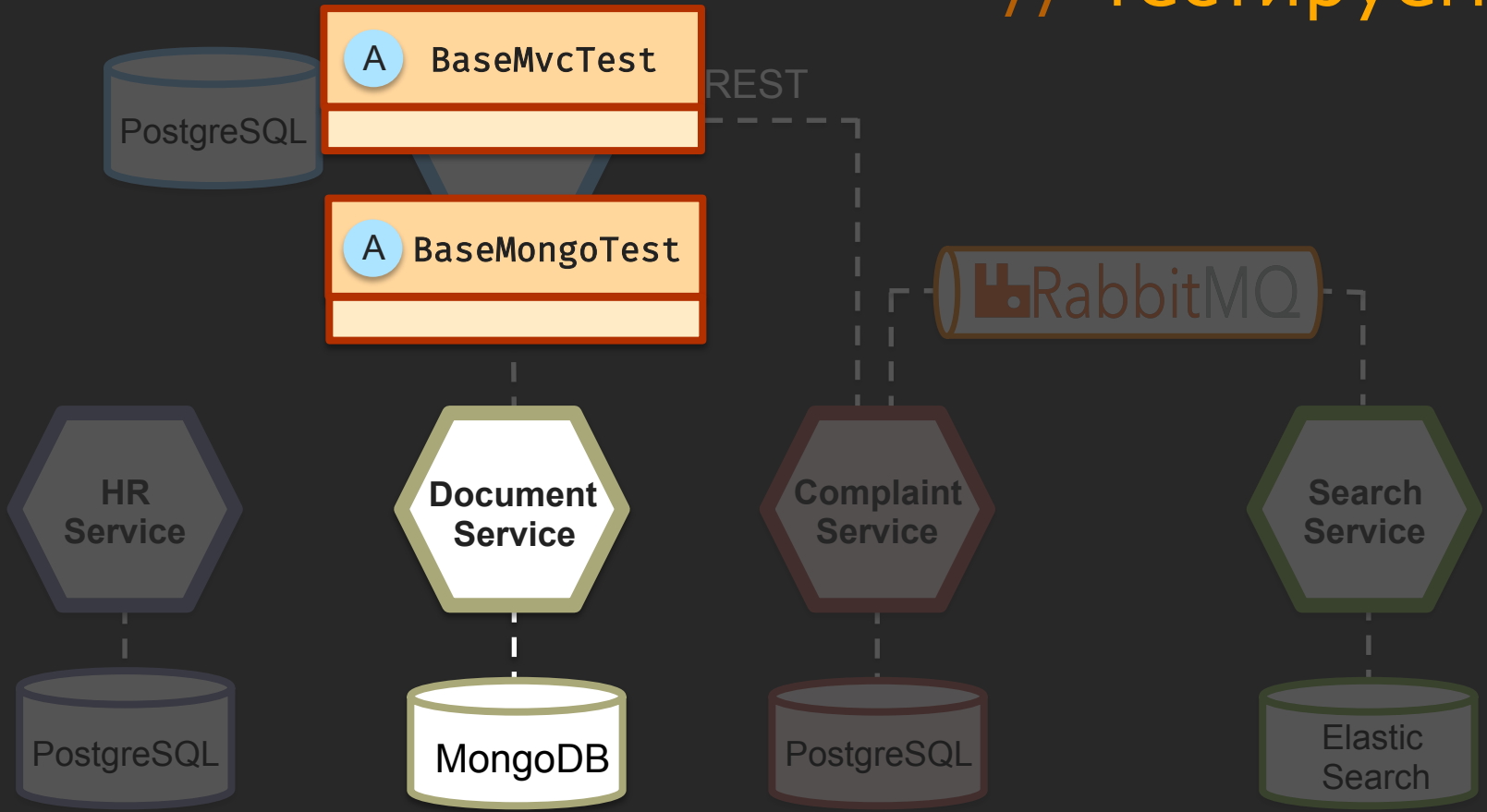
// распиливаем
// монолит



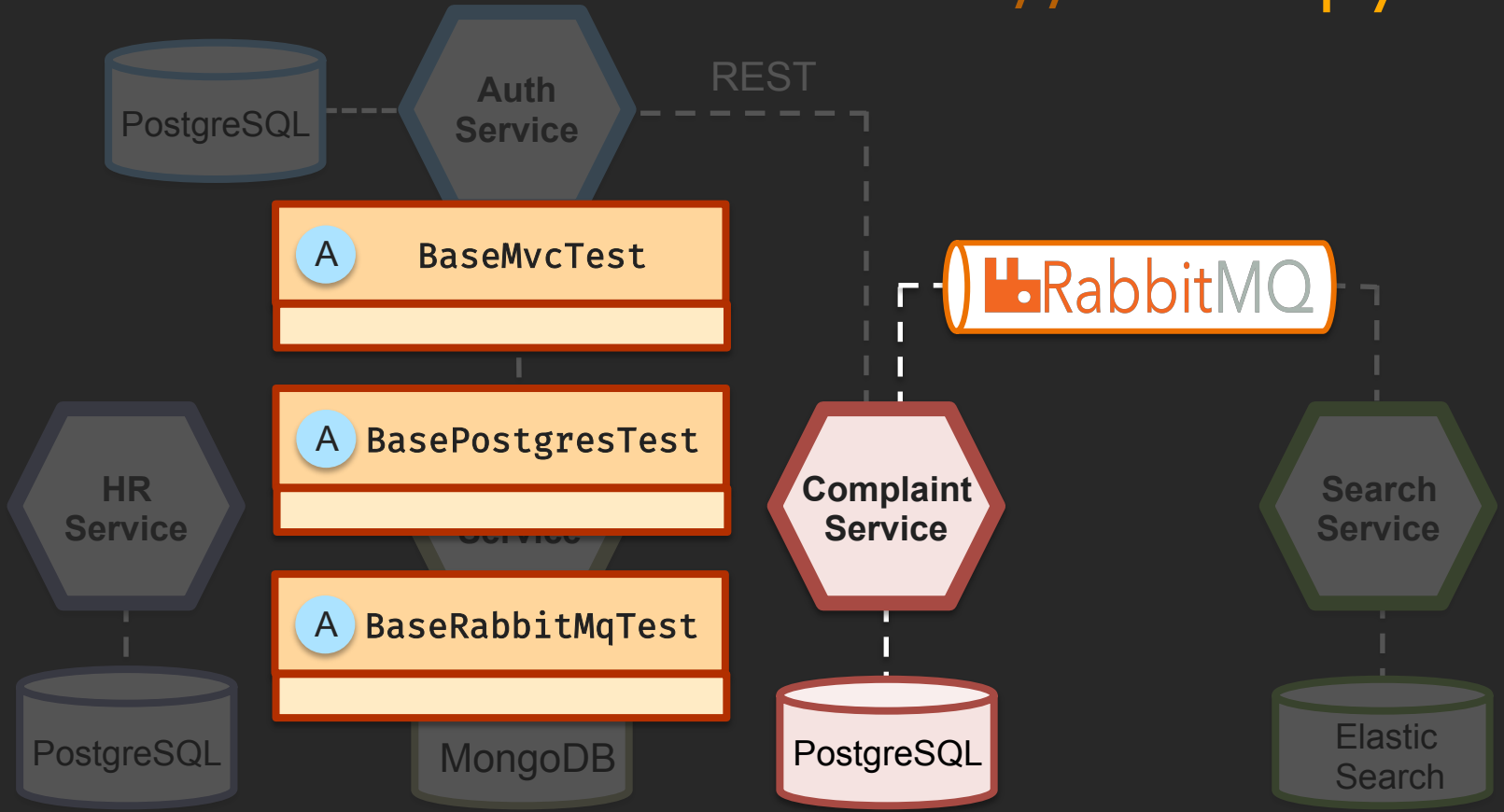
// тестируем


A BaseMvcTest
A BasePostgresTest
Auth Service
REST
REST
RabbitMQ
HR Service
Document Service
Complaint Service
Search Service
PostgreSQL
MongoDB
PostgreSQL
Elastic Search

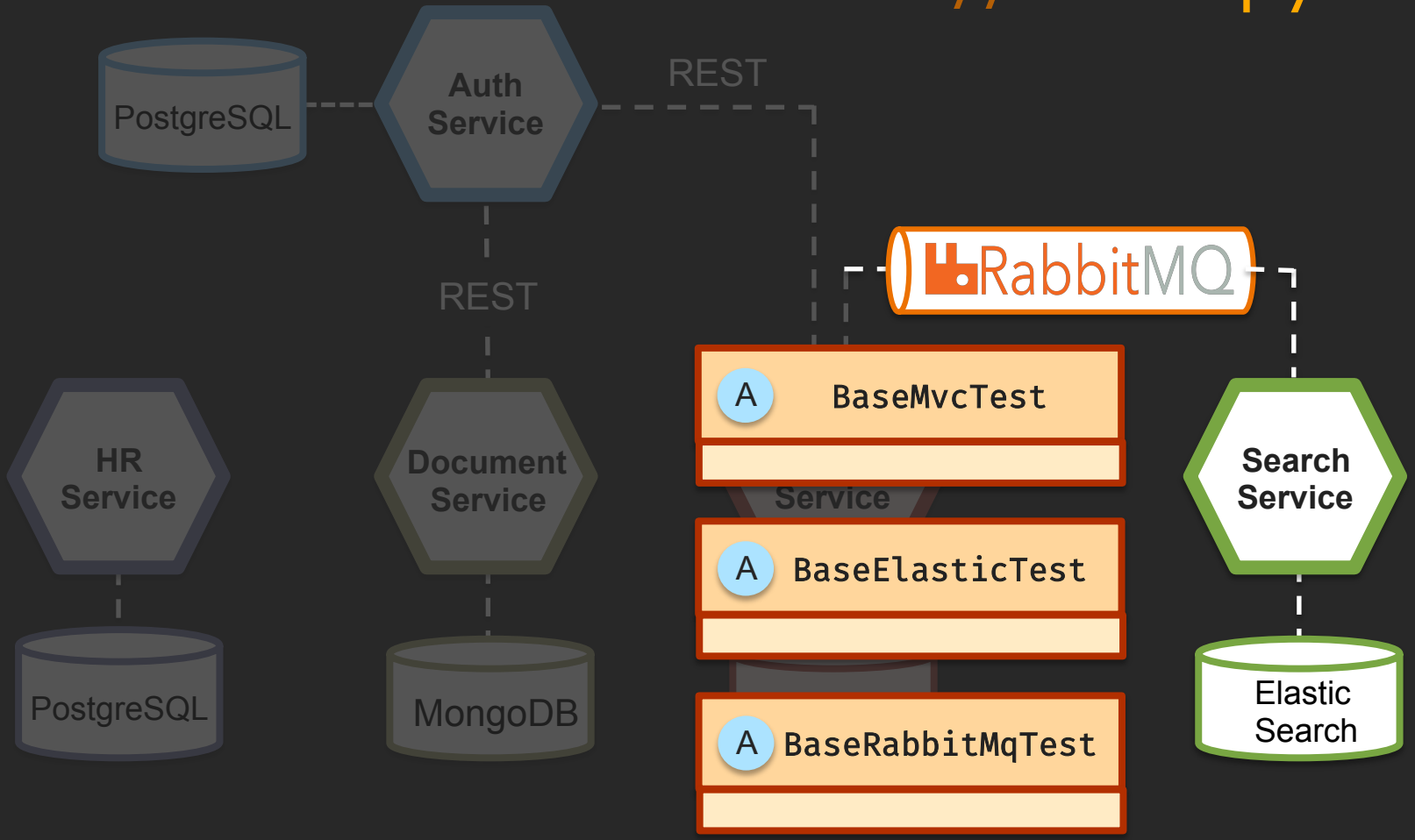
// тестируем

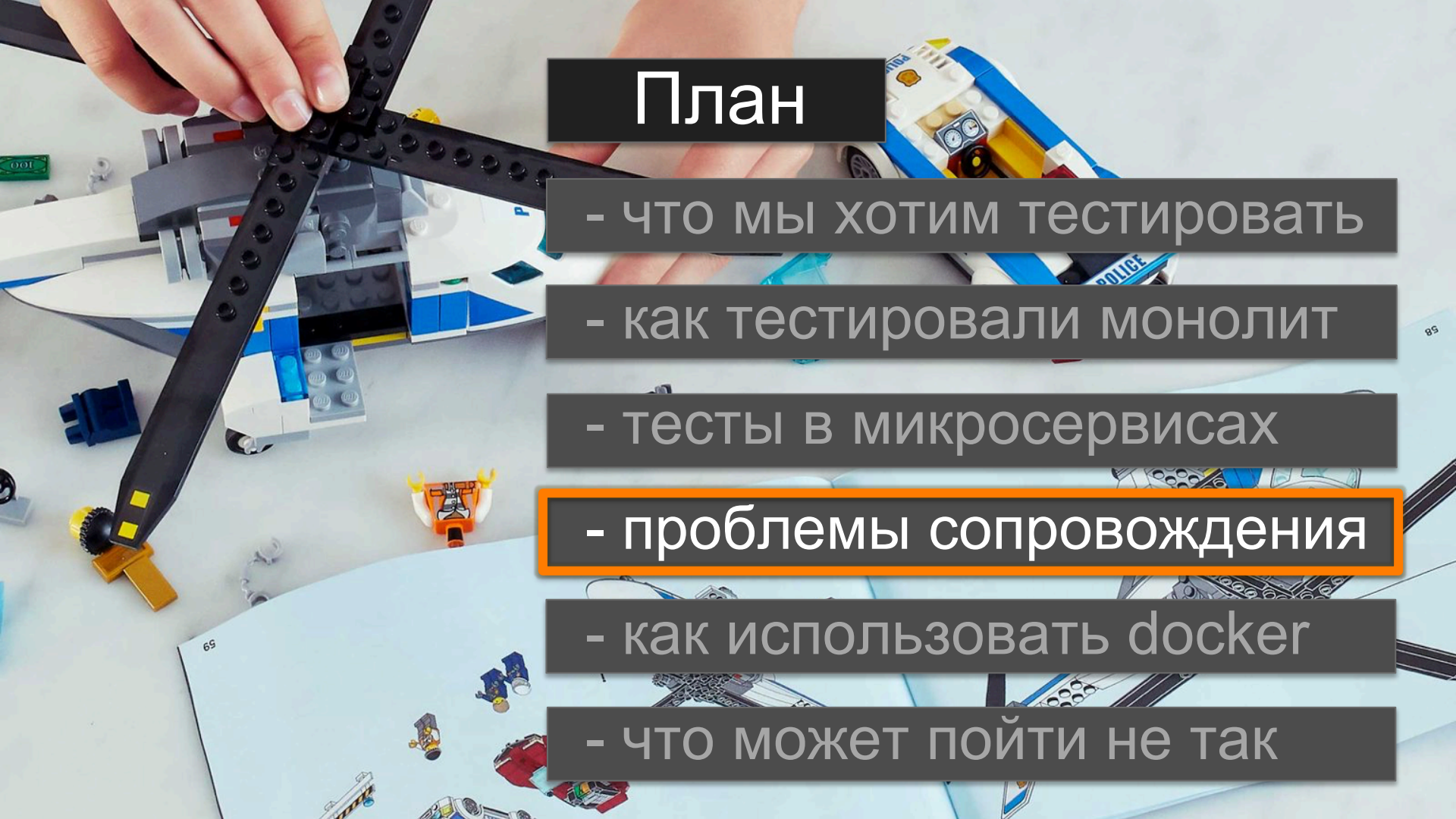

A BaseMvcTest
PostgreSQL
REST
A BaseMongoTest
RabbitMQ
HR Service
Document Service
Complaint Service
Search Service
PostgreSQL
MongoDB
PostgreSQL
Elastic Search

// тестируем


PostgreSQL
Auth Service
REST
A BaseMvcTest
A BasePostgresTest
A BaseRabbitMqTest
RabbitMQ
HR Service
Complaint Service
Search Service
PostgreSQL
MongoDB
PostgreSQL
Elastic Search

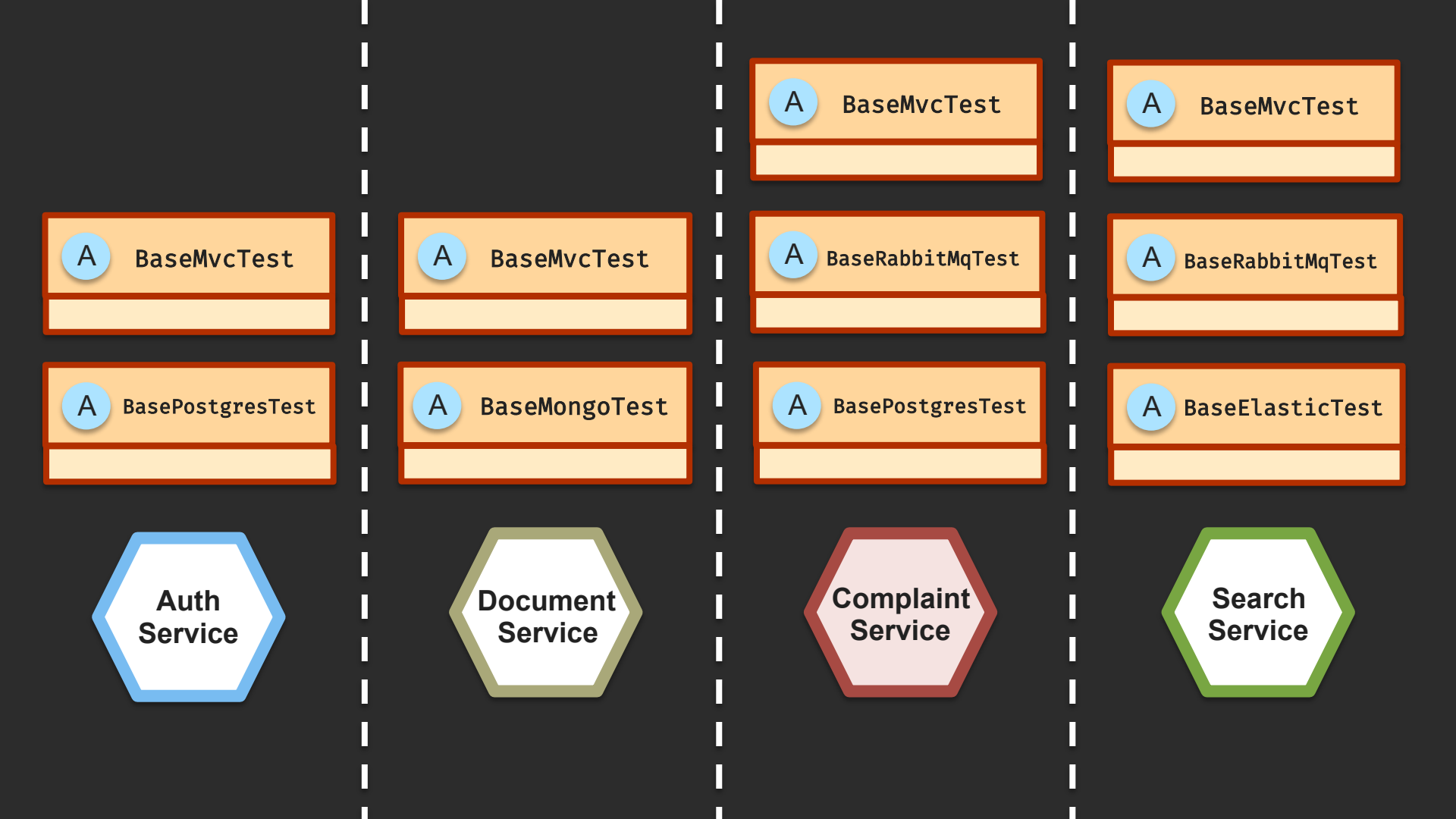
// тестируем

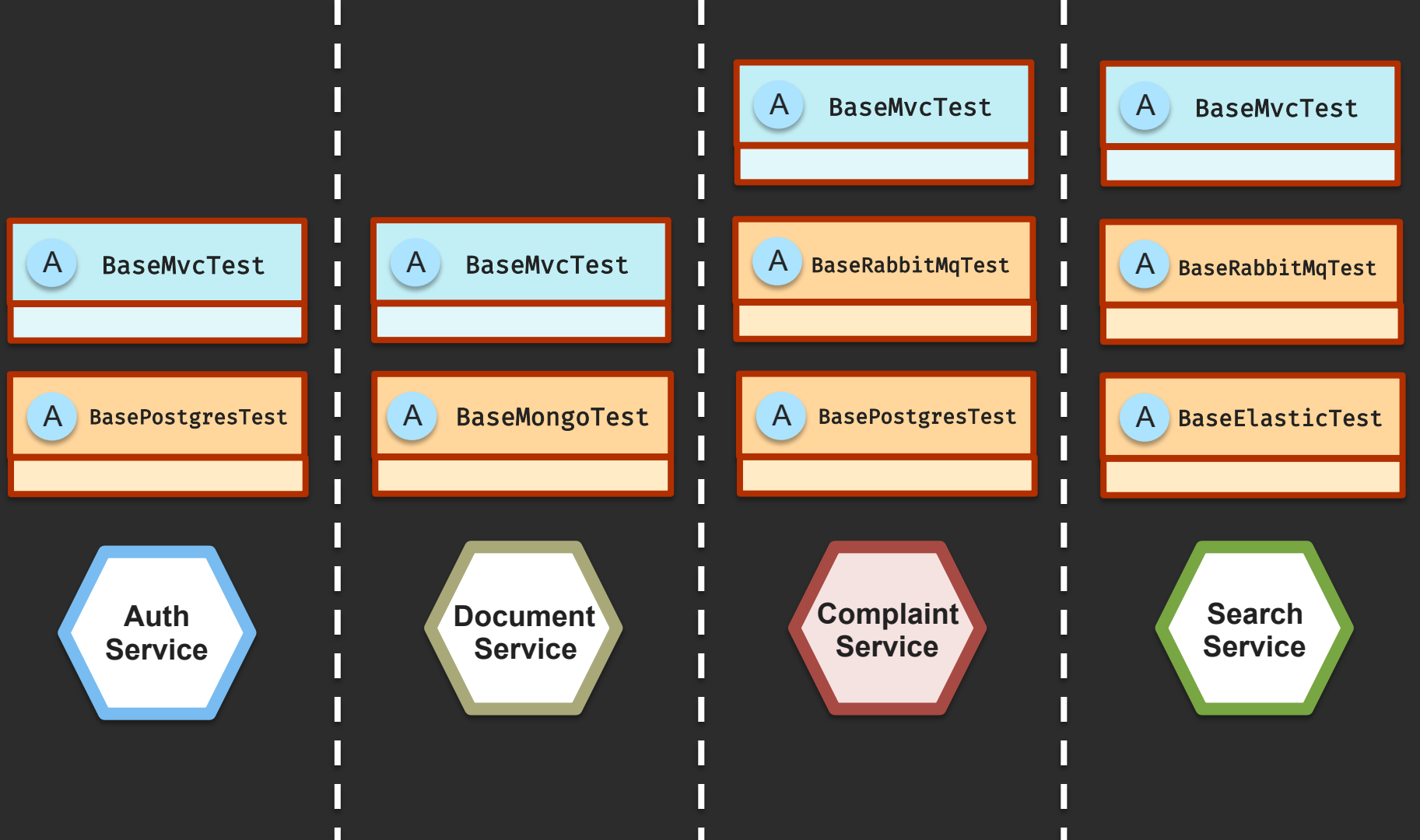

PostgreSQL
Auth Service
REST
REST
RabbitMQ
HR Service
Document Service
Search Service
PostgreSQL
MongoDB
Elastic Search
A BaseMvcTest
Service
A BaseElasticTest
A BaseRabbitMqTest

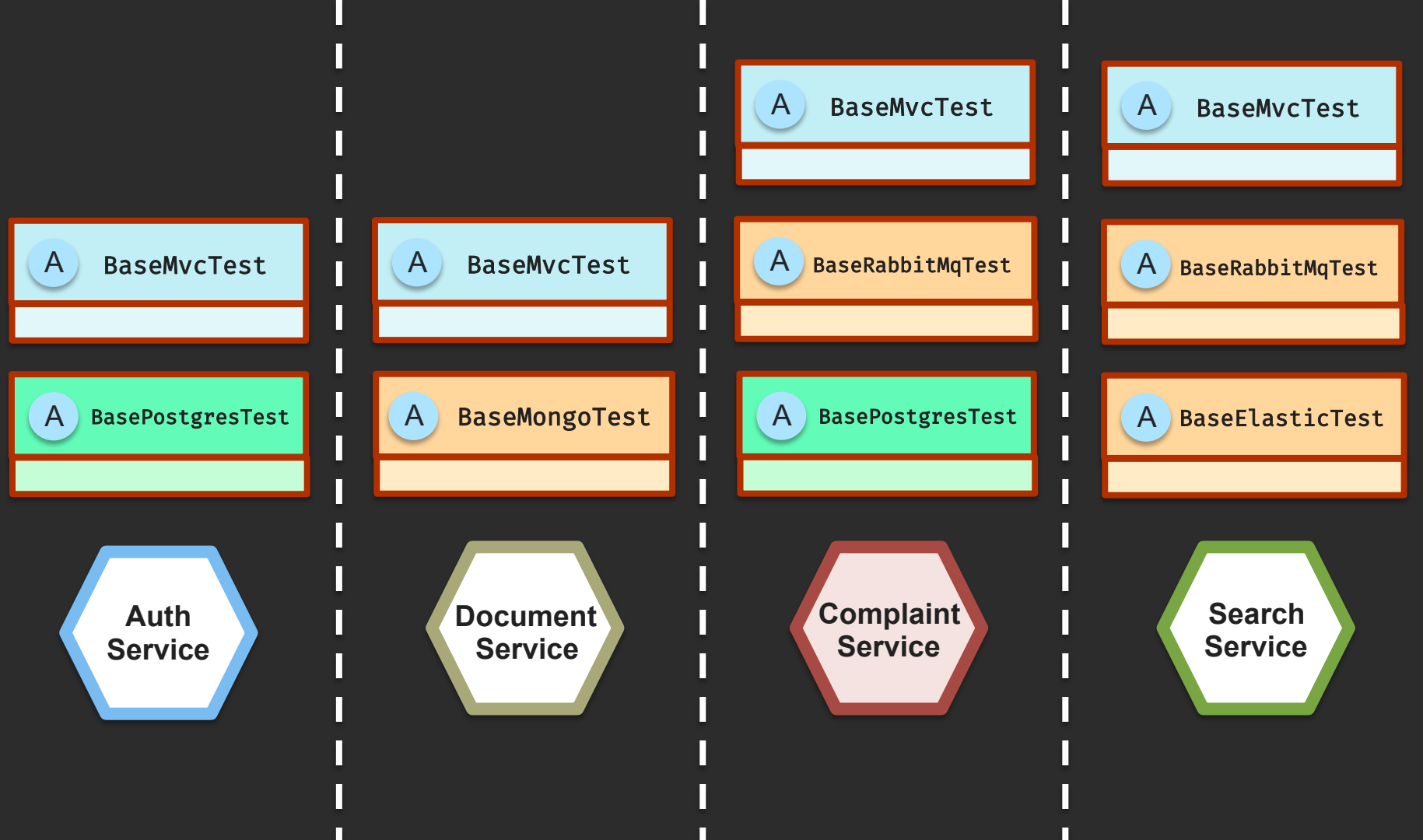


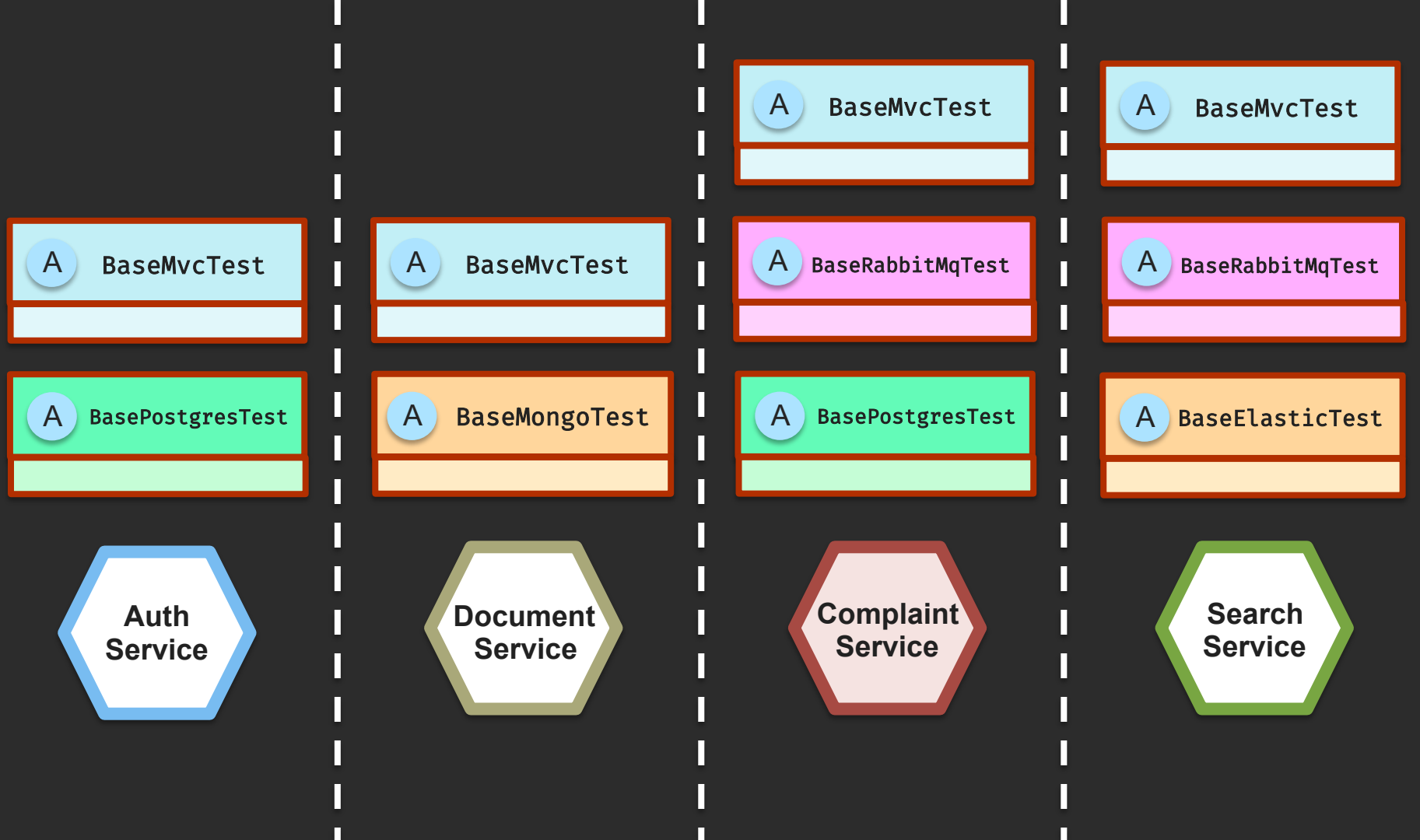
План

- что мы хотим тестировать
- как тестировали монолит
- тесты в микросервисах
- проблемы сопровождения
- как использовать docker
- что может пойти не так


BaseMvcTest
BasePostgresTest
Auth Service
BaseMvcTest
BaseMongoTest
Document Service
BaseMvcTest
BaseRabbitMqTest
BasePostgresTest
Complaint Service
BaseMvcTest
BaseRabbitMqTest
BaseElasticTest
Search Service

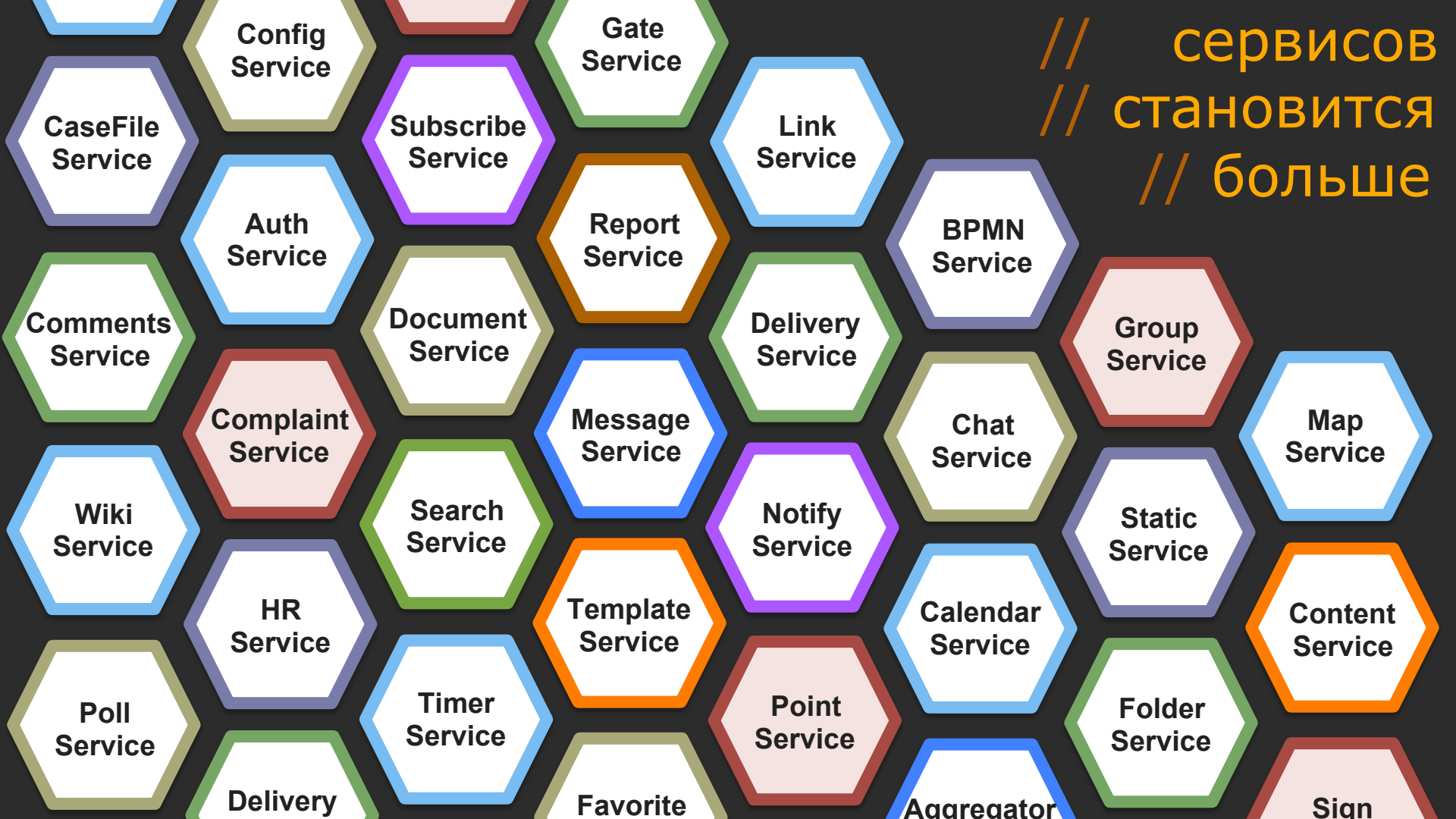

A BaseMvcTest
A BasePostgresTest
Auth Service
A BaseMvcTest
A BaseMongoTest
Document Service
A BaseMvcTest
A BaseRabbitMqTest
A BasePostgresTest
Complaint Service
A BaseMvcTest
A BaseRabbitMqTest
A BaseElasticTest
Search Service


A BaseMvcTest
A BasePostgresTest
Auth Service
A BaseMvcTest
A BaseMongoTest
Document Service
A BaseMvcTest
A BaseRabbitMqTest
A BasePostgresTest
Complaint Service
A BaseMvcTest
A BaseRabbitMqTest
A BaseElasticTest
Search Service


A BaseMvcTest
A BasePostgresTest
Auth Service
A BaseMvcTest
A BaseMongoTest
Document Service
A BaseMvcTest
A BaseRabbitMqTest
A BasePostgresTest
Complaint Service
A BaseMvcTest
A BaseRabbitMqTest
A BaseElasticTest
Search Service

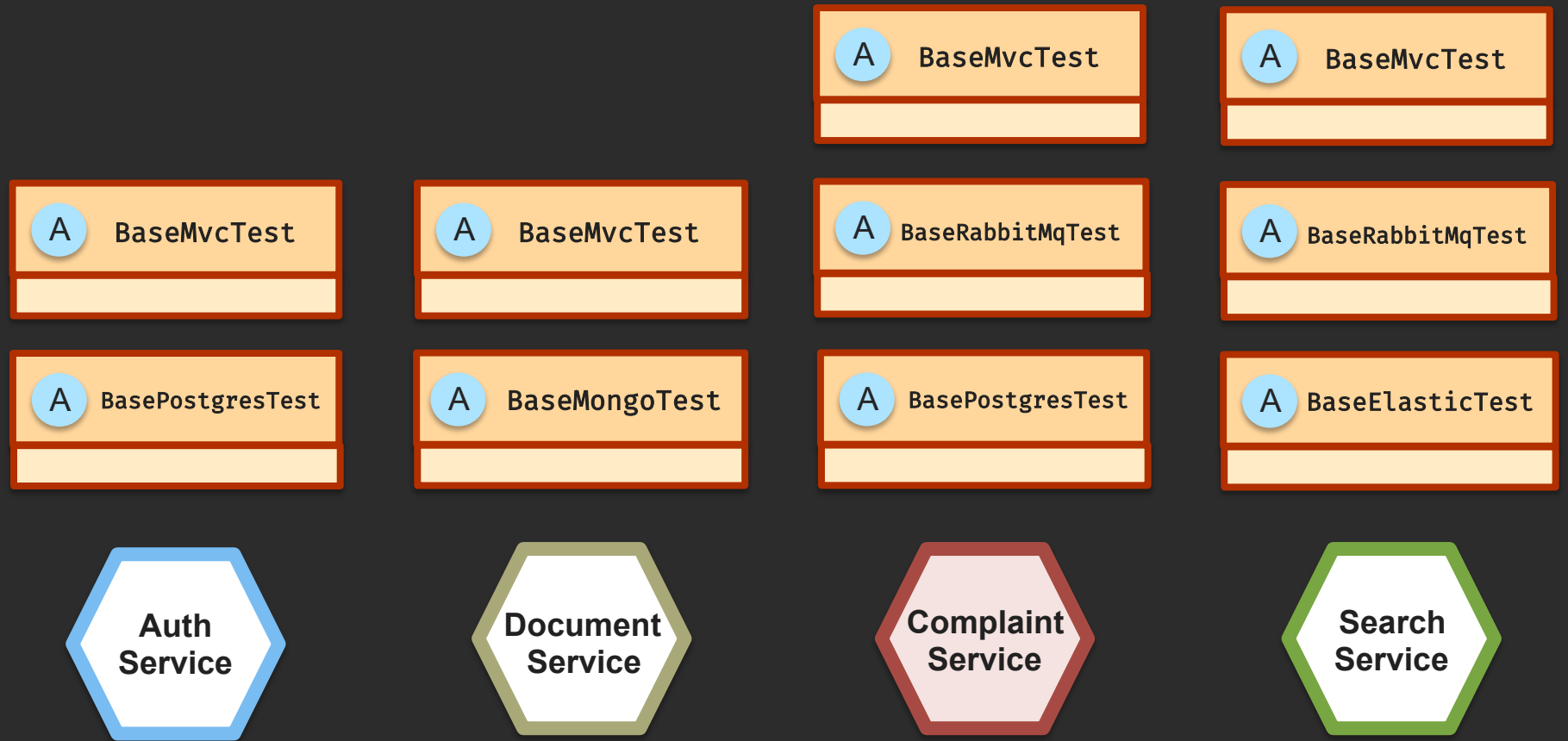
// сервисов
// становится
// больше

// сервисов
// становится
// больше

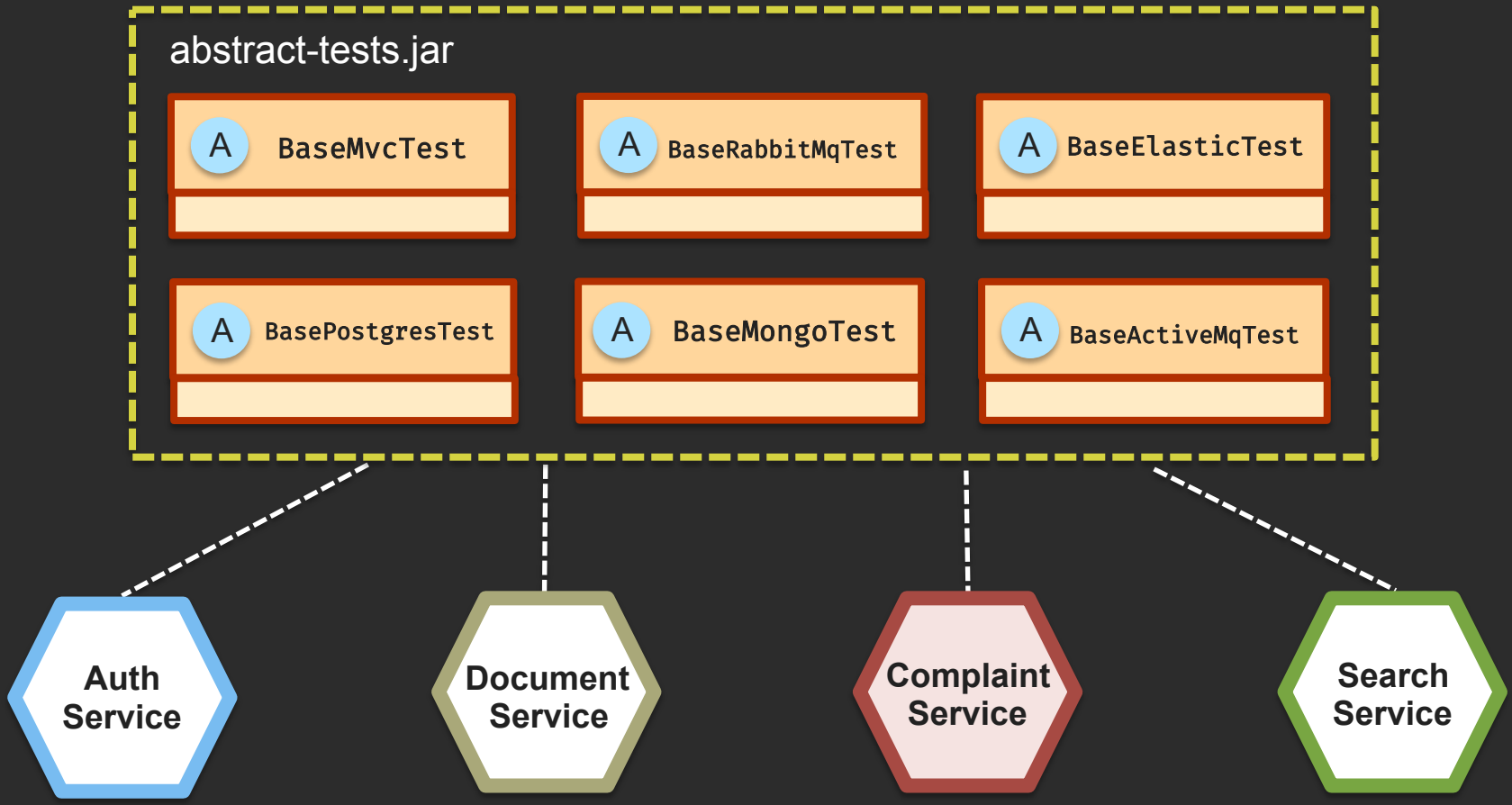

// сервисов
// становится
// больше
Config Service
Gate Service
CaseFile Service
Subscribe Service
Link Service
Auth Service
Report Service
BPMN Service
Comments Service
Document Service
Delivery Service
Group Service
Complaint Service
Message Service
Chat Service
Map Service
Wiki Service
Search Service
Notify Service
Static Service
HR Service
Template Service
Calendar Service
Content Service
Poll Service
Timer Service
Point Service
Folder Service
Delivery
Favorite
Aggregator
Sign

// Продолжать копировать базовые классы?

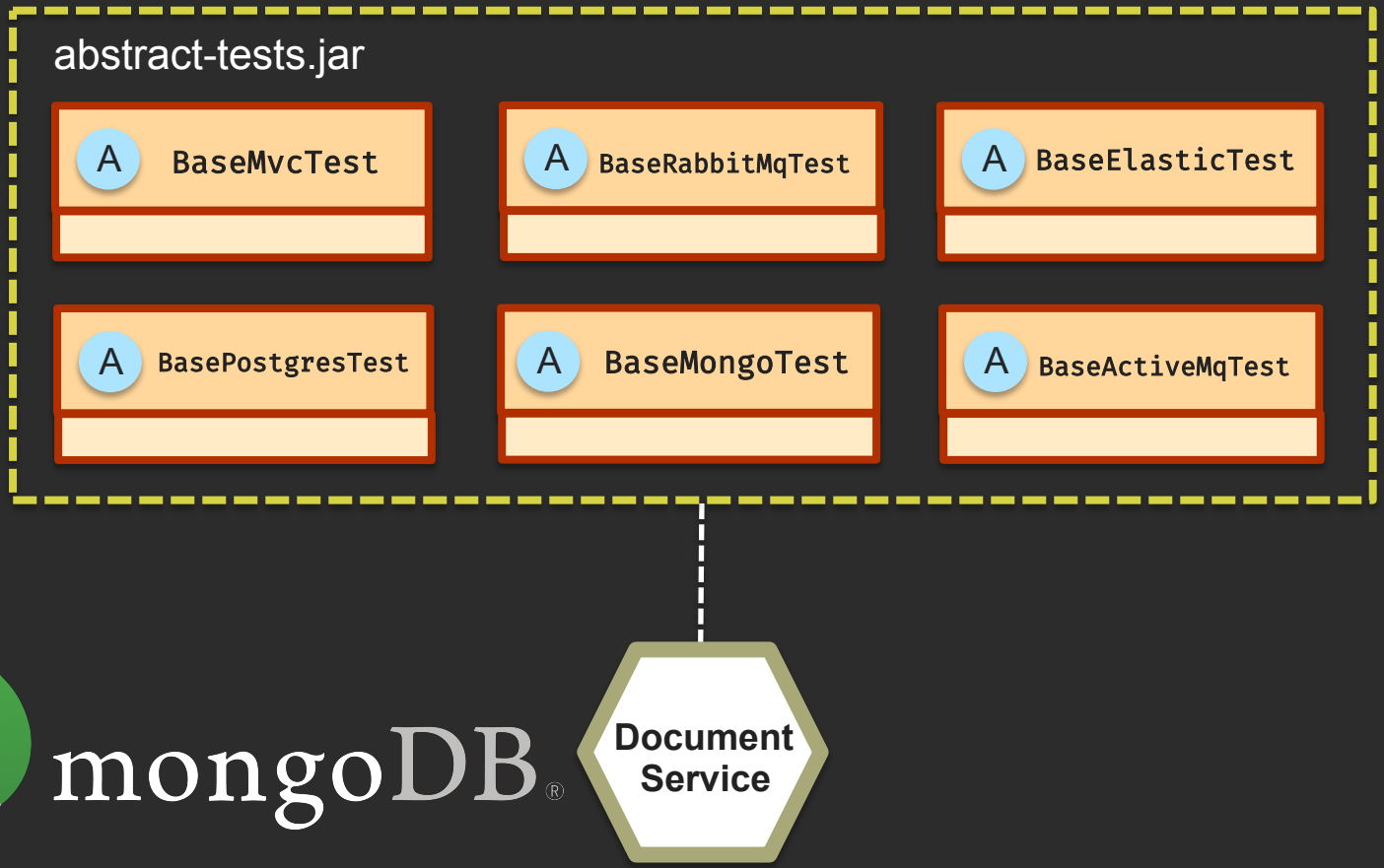
// как можно решить эту проблему?



// вынесем в отдельную зависимость



// всегда ли нужны все конфигурации



// много лишних зависимостей

spring-boot-starter-activemq

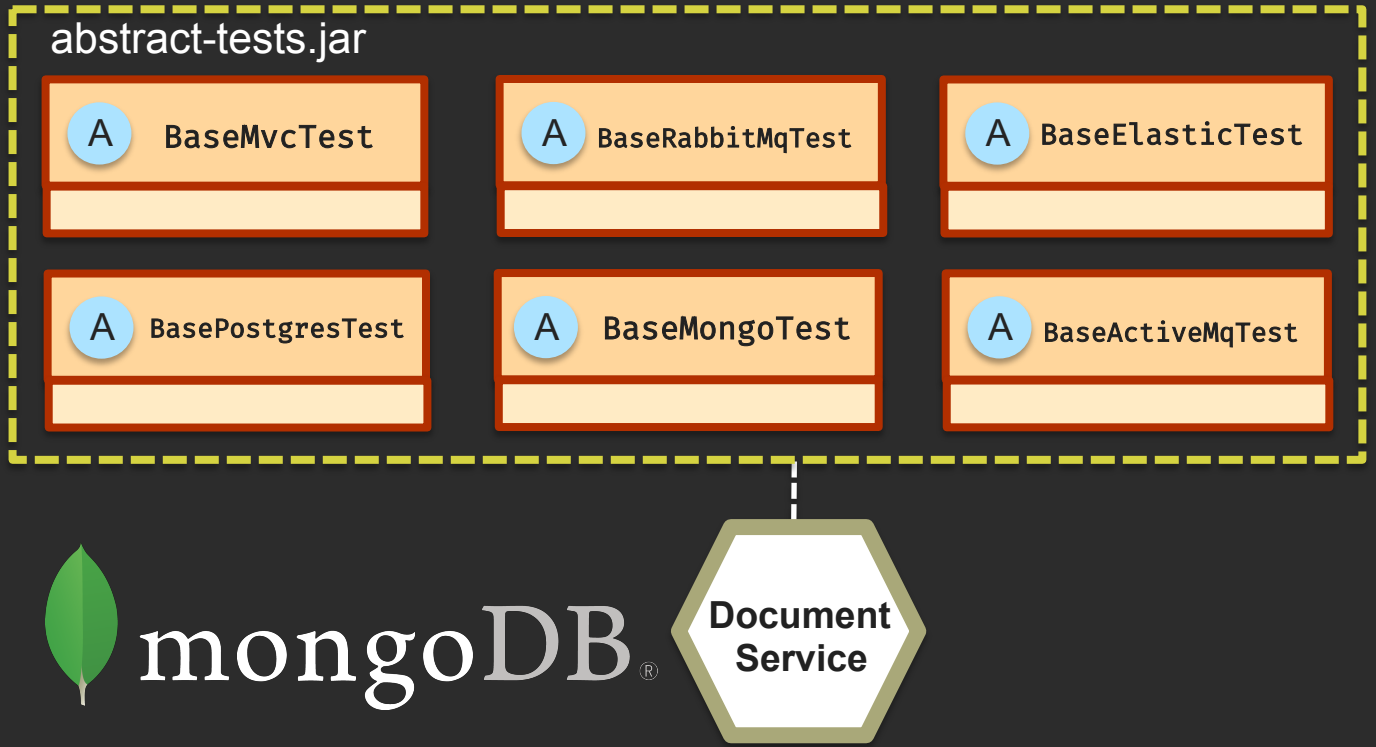
spring-boot-starter-data-jpa

spring-boot-starter-web

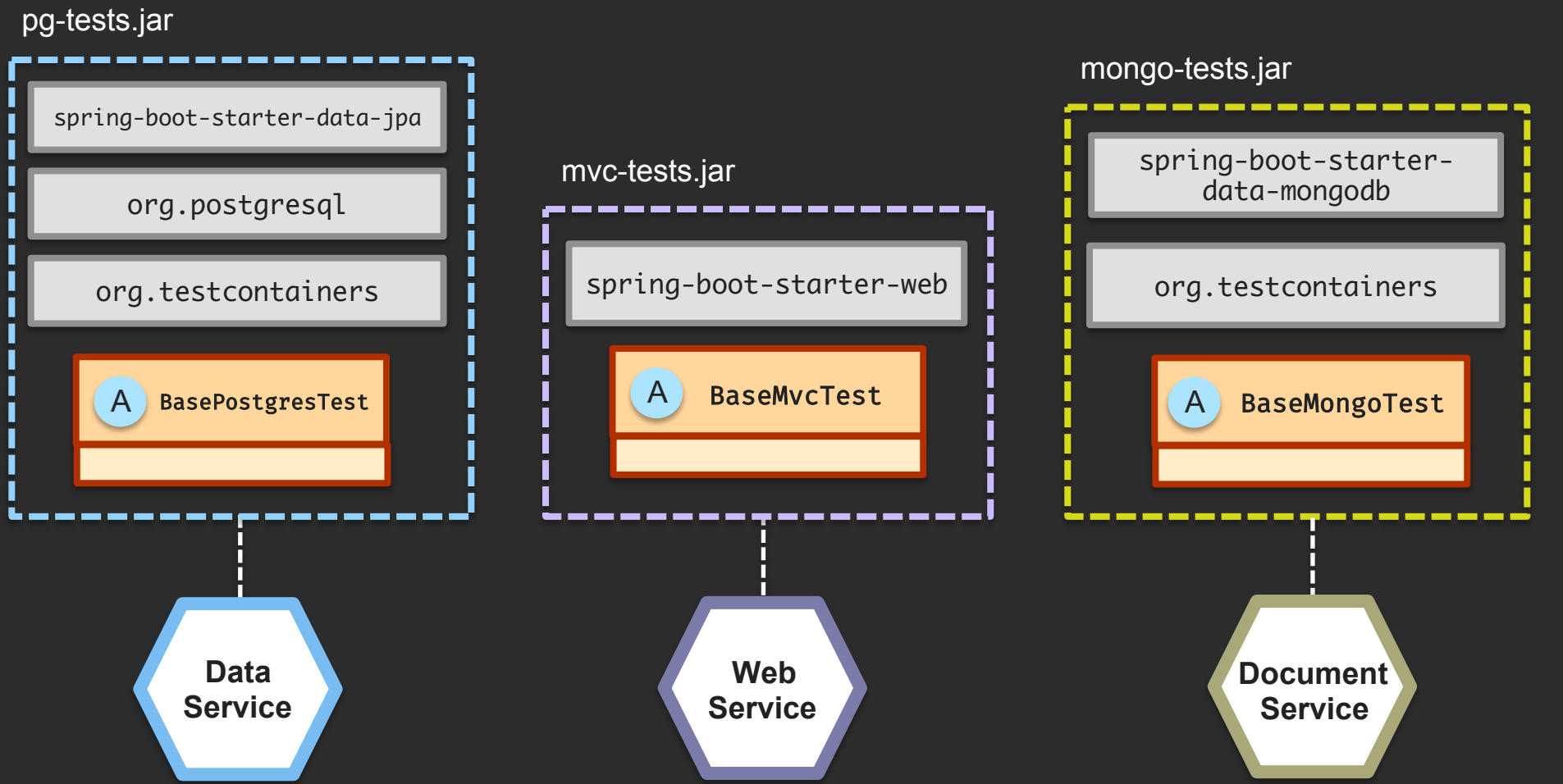
spring-boot-starter-data-mongodb

org.testcontainers

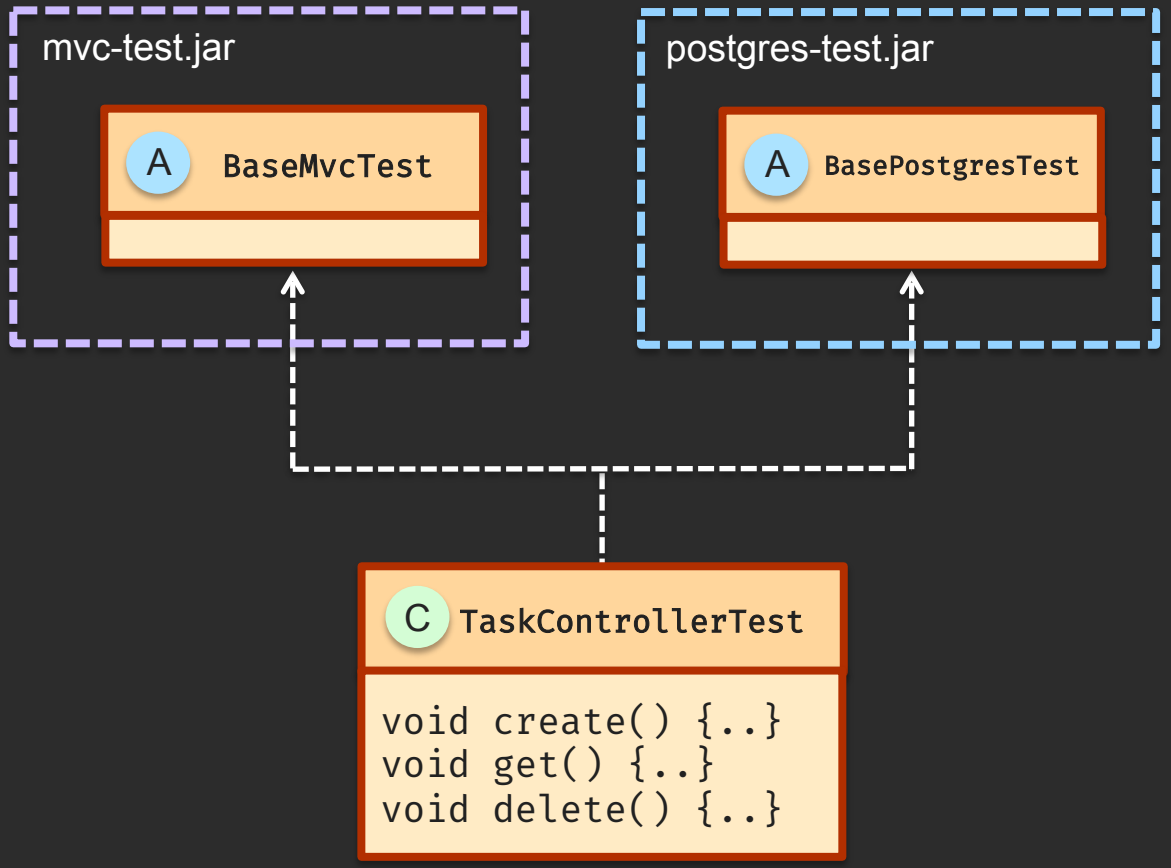
org.postgresql



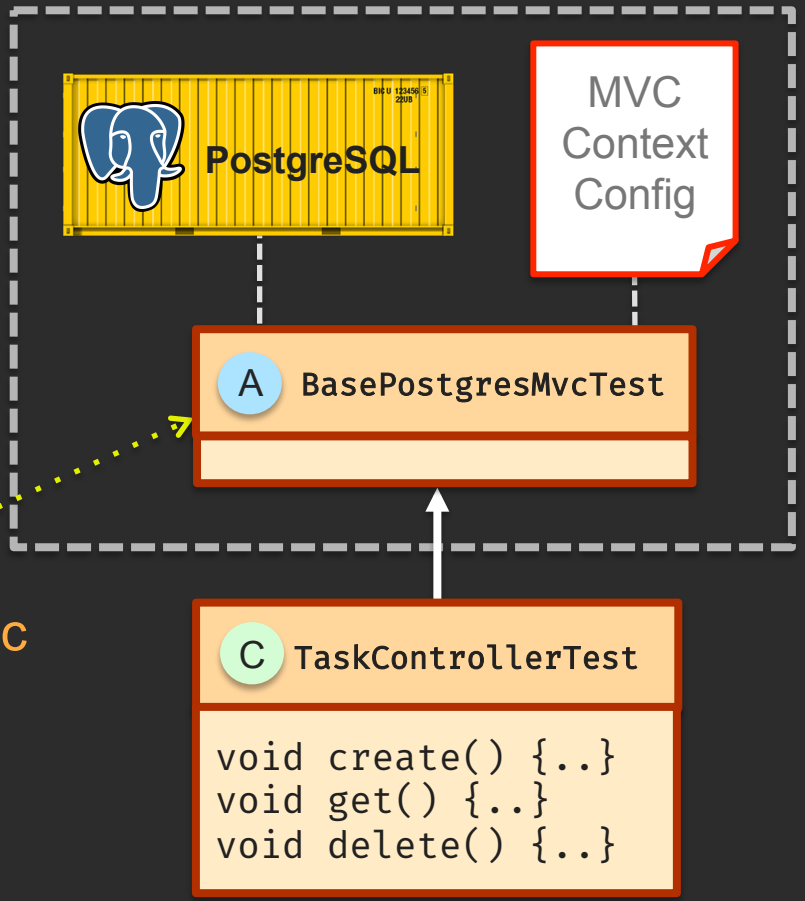
// разделим по логике



// А что, если нужно и то и другое?

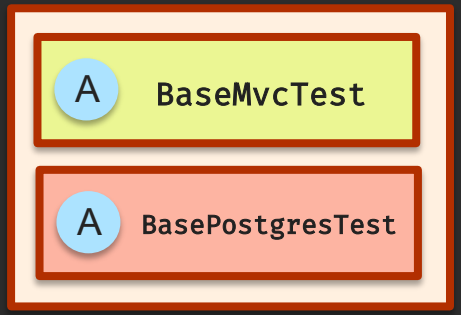


// как бы мы сделали это в монолите



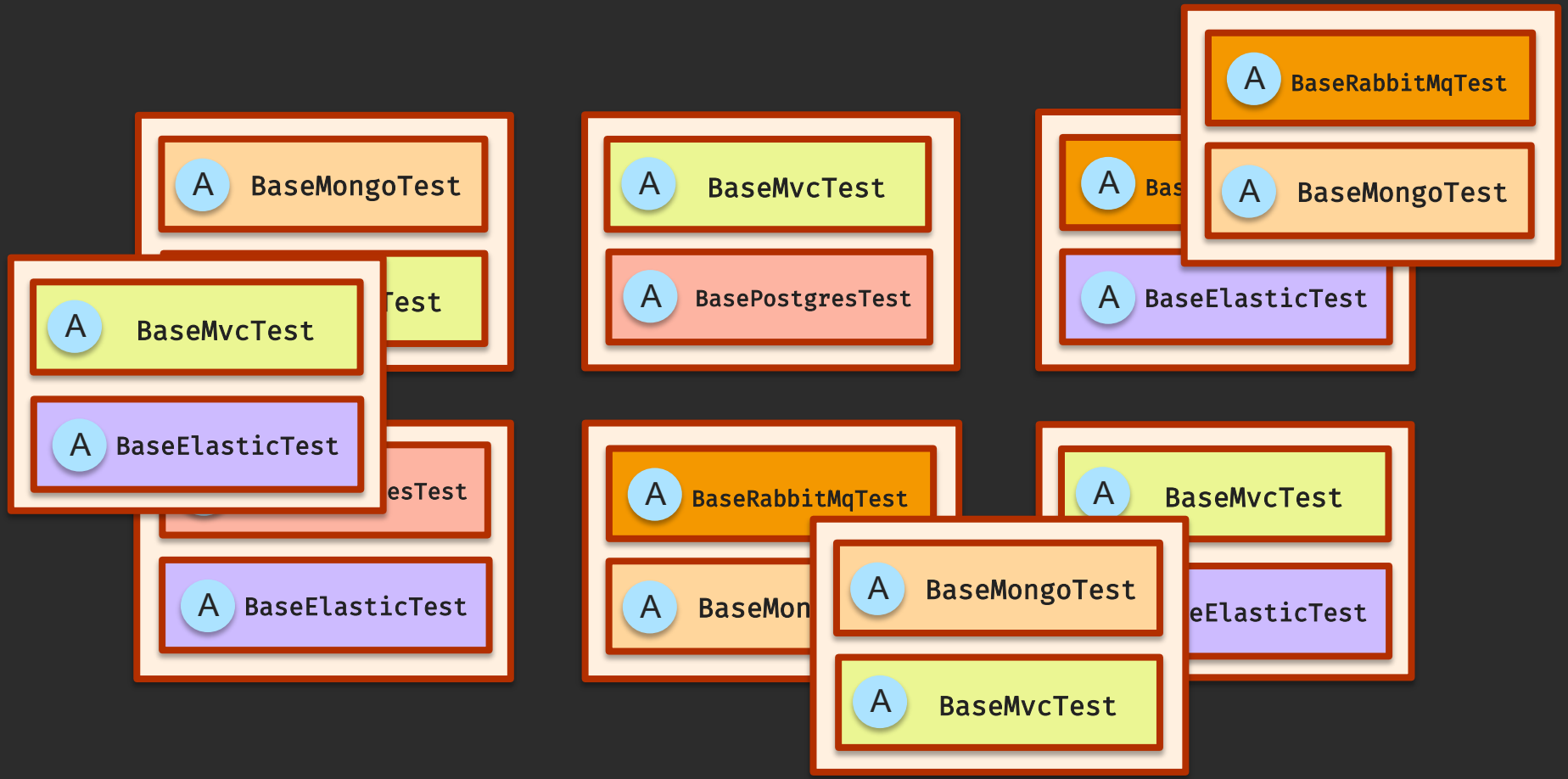
Еще один
абстрактный класс

// и так придется делать для каждой комбинации

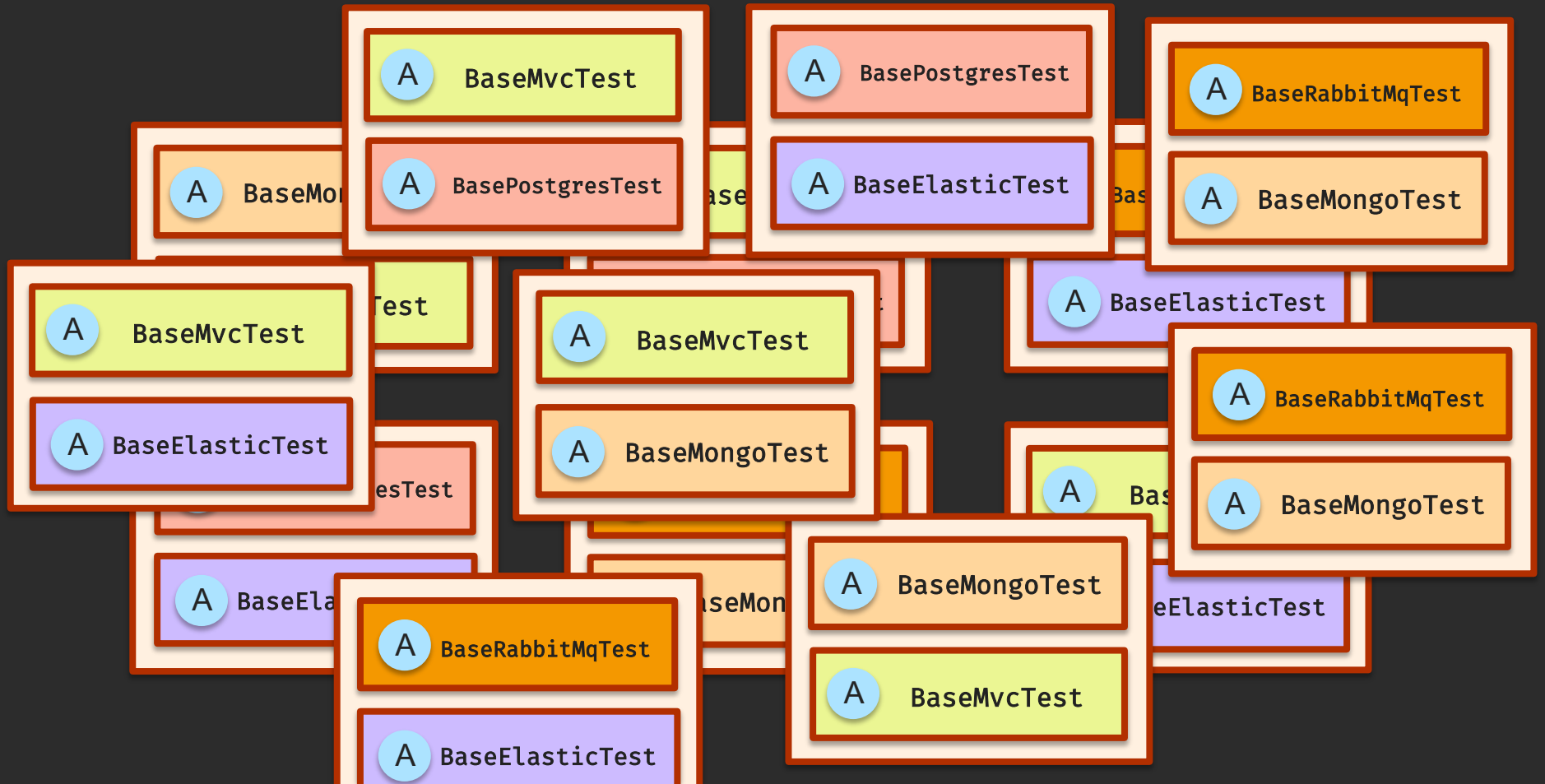


// и так придется делать для каждой комбинации

// и так придется делать для каждой комбинации



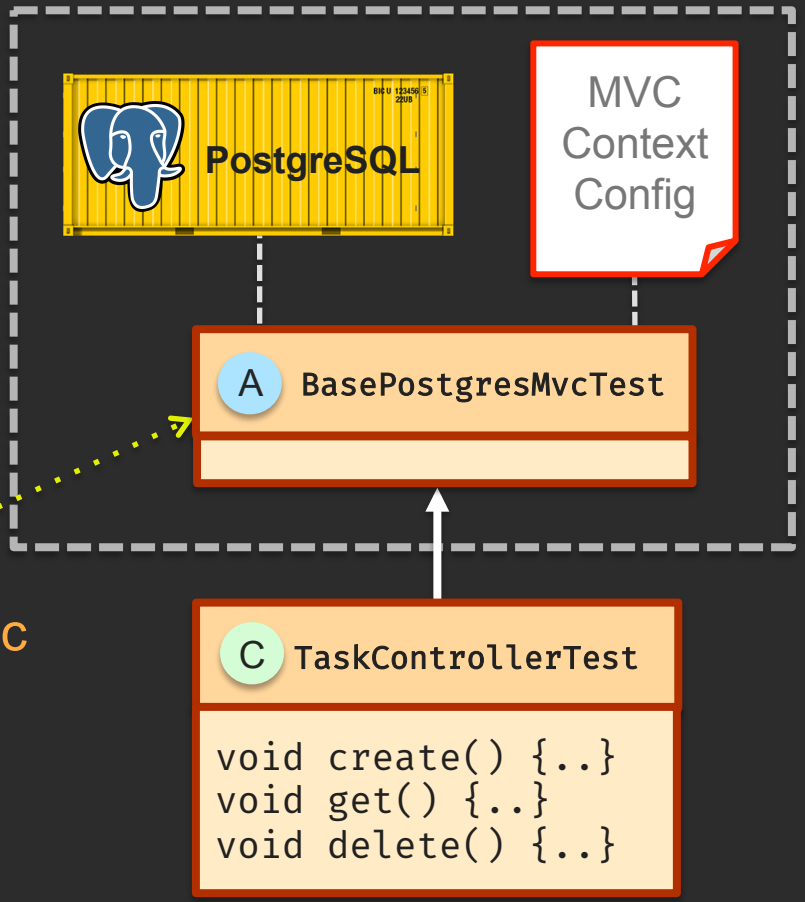
// и так придется делать для каждой комбинации



Не - не - не,
моя умея только копироватя,

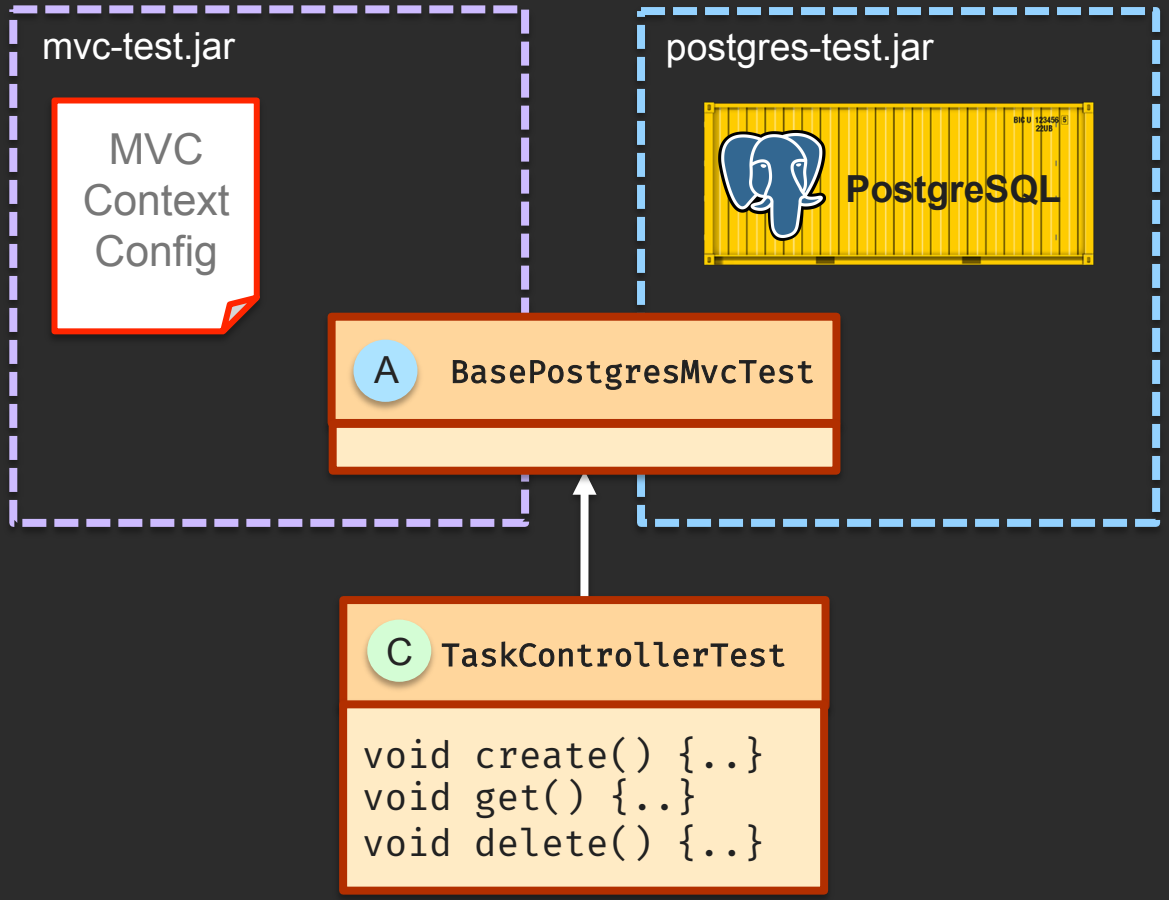
Сопровождать
моя не училса

// как бы мы сделали это в монолите

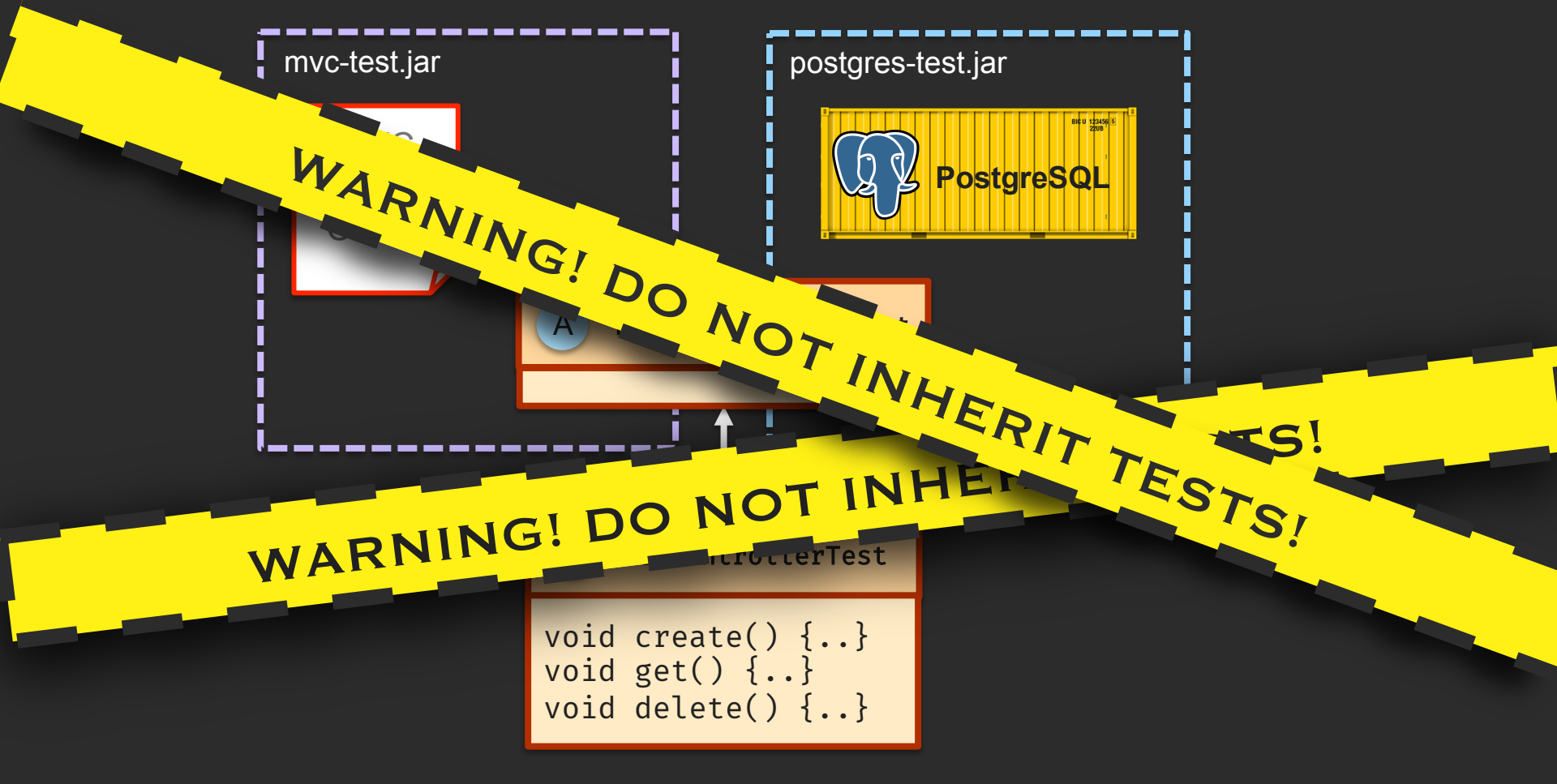


Еще один
абстрактный класс

// А что в микросервисах?



// А что в микросервисах?



```
public class TaskServiceTest extends BasePostgresMvcTest {

      @Test
      @DataSet(value = "datasets/task.json")
      public void getTask() {

            Task task = taskService.get(TASK_ID);
            assertThat(task).isNotNull();
      }

}
```

// заменим наследование мета-аннотациями

```
@EnableMvcTest
@EnablePostgresTest
public class TaskServiceTest {

    @Test
    public void getTask() {
        ...
    }
}
```

// что внутри @EnableMvcTests?

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@RunWith(SpringRunner.class)
@AutoConfigureMockMvc
public @interface EnableMvcTests {

}
```

// что внутри @EnablePostgresTests?

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@RunWith(SpringRunner.class)
@DataJpaTest
@AutoConfigureTestDatabase(replace = NONE)
public @interface EnablePostgresTests {

}
```

// Не все так просто

```
1: @Retention(RetentionPolicy.RUNTIME)
2: @Target(ElementType.TYPE)
3: @RunWith(SpringRunner.class)
4: @DataJpaTest
5: @AutoConfigureTestDatabase(replace = NONE)
6: public @interface EnablePostgresTests {
7:
8: }
```

// Не все так просто

```
1: @Retention(RetentionPolicy.RUNTIME)
2: @Target(ElementType.TYPE)
3: @RunWith(SpringRunner.class)
4: @DataJpaTest
5: @AutoConfigureTestDatabase(replace = NONE)
6: public @interface EnablePostgresTests {
7:
8: }
```


JUnit4 @RunWith
• class level
• non repeatable

// Решение

// Мета-аннотации и ExtendWith

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@ExtendWith(SpringExtension.class)
@DataJpaTest
@AutoConfigureTestDatabase(replace = NONE)
public @interface EnableDataTests {

}
```

Repeatable & Inherited

// Компонуем, как хотим

```
@EnablePostgresTest
class TaskServiceTest {

      @Test
      void getTask() {
            ...
      }

}
```

// Компонуем, как хотим

```
@EnableMvcTest
@EnablePostgresTest
class TaskServiceTest {

      @Test
      void getTask() {
            ...
      }

}
```

// Компонуем, как хотим

```
@EnableMvcTest
@EnablePostgresTest
@EnableRabbitMqTest
class TaskServiceTest {

    @Test
    void getTask() {
        ...
    }

}
```

// Компонуем, как хотим

```
@EnableMongoDbTest
class DocumentServiceTest {

      @Test
      void get() {
            ...
      }
}
```

// Компонуем, как хотим

```
@EnableMvcTest
@EnableMongoDbTest
class DocumentServiceTest {

      @Test
      void get() {
            ...
      }
}
```

// Компонуем, как хотим

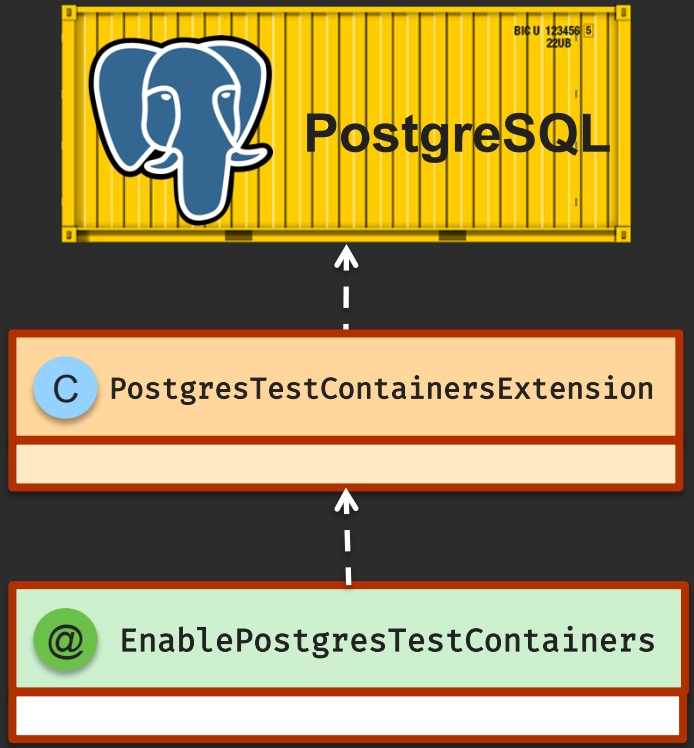
```
@EnableMvcTest
@EnableMongoDbTest
@EnableActiveMqTest
class DocumentServiceTest {

    @Test
    void get() {
        ...
    }
}
```

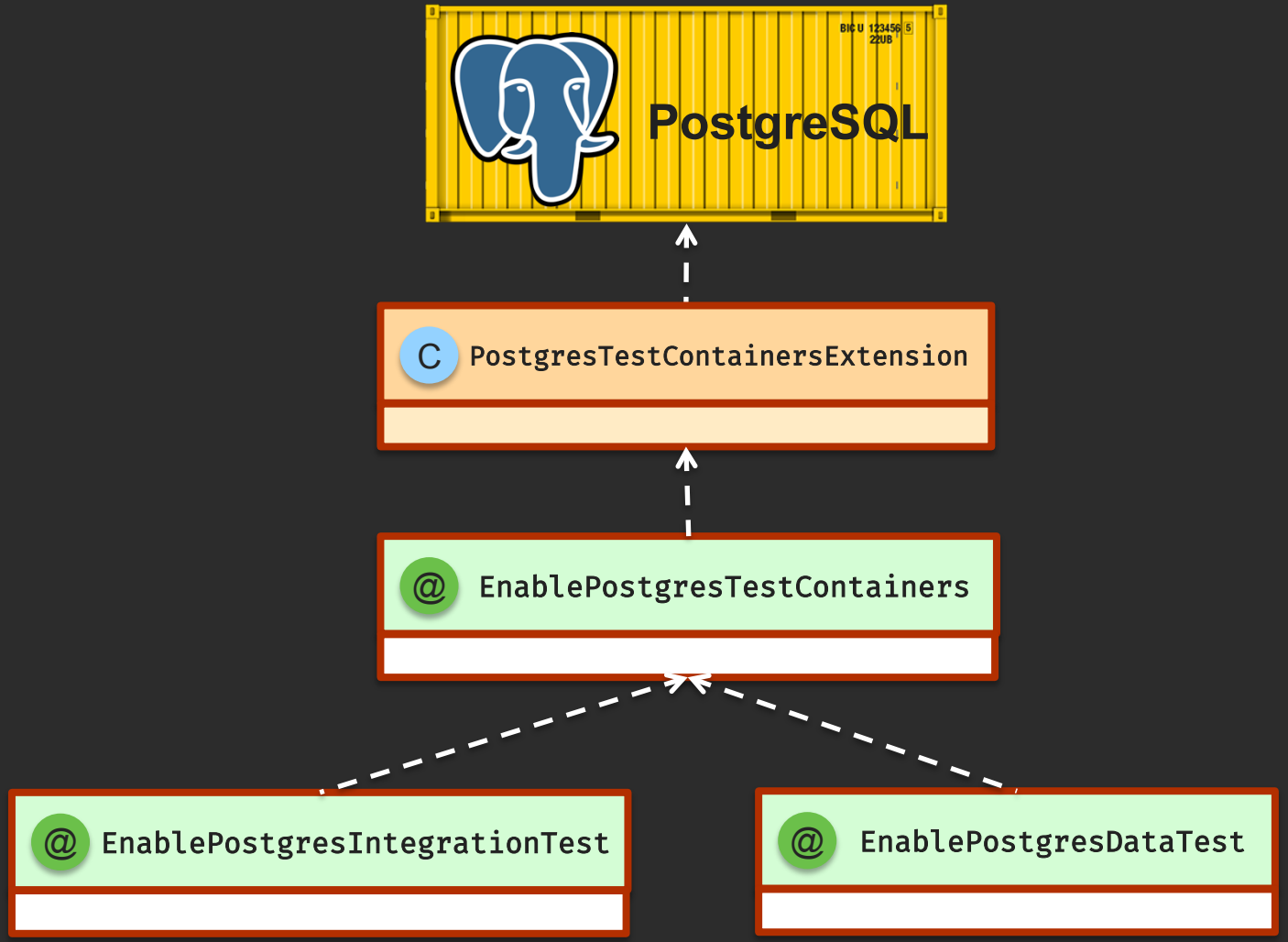
// тестируем

хранимые процедуры

нативные запросы

расширения postgres


PostgreSQL
PostgresTestContainersExtension
EnablePostgresTestContainers


PostgreSQL
C
PostgresTestContainersExtension
@
EnablePostgresTestContainers
@
EnablePostgresIntegrationTest
@
EnablePostgresDataTest

```
@EnablePostgresDataTest
class NativeFunctionTest {

    @PersistenceContext private EntityManager entityManager;

    @Test
    @Sql("/stored_functions/magic_func.sql")
    void testStoredFunc() {
        StoredProcedureQuery query =
          entityManager.createStoredProcedureQuery("magic");

        query.execute();
        List resultList = query.getResultList();

        int result = (int) resultList.get(0);
        Assertions.assertThat(result).isEqualTo(1234);
    }
}
```

```
@EnablePostgresDataTest
class NativeFunctionTest {

    @PersistenceContext private EntityManager entityManager;

    @Test
    @Sql("/stored_functions/magic_func.sql")
    void testStoredFunc() {
        StoredProcedureQuery query =
          entityManager.createStoredProcedureQuery("magic");

        query.execute();
        List resultList = query.getResultList();

        int result = (int) resultList.get(0);
        Assertions.assertThat(result).isEqualTo(1234);
    }
}
```

```
@EnablePostgresDataTest
class NativeFunctionTest {

    @PersistenceContext private EntityManager entityManager;

    @Test
    @Sql("/stored_functions/magic_func.sql")
    void testStoredFunc() {
        StoredProcedureQuery query =
          entityManager.createStoredProcedureQuery("magic");

        query.execute();
        List resultList = query.getResultList();

        int result = (int) resultList.get(0);
        Assertions.assertThat(result).isEqualTo(1234);
    }
}
```

```
DROP FUNCTION IF EXISTS magic(regCardId character varying);

CREATE FUNCTION magic(regCardId character varying)
RETURNS refcursor AS
'
  DECLARE
    ref refcursor;
  BEGIN
    OPEN ref FOR
 WITH temp_assignments AS
 ( SELECT assignment.id, assignment.executor_id
      FROM execution execution
      LEFT JOIN assignment assignment ON assignment.parent_execution_id = execution.id
      WHERE execution.regcard_id = regCardId
      AND execution.parent_execution_id IS NULL ),

temp_executors AS
( SELECT executor_employee.name as name -- главный исполнитель
  FROM temp_assignments assignment
  INNER JOIN employee executor_employee ON executor_employee.id = assignment.executor_id

  UNION ALL

  SELECT executor_employee.name as name -- остальные исполнители
  FROM temp_assignments assignment
  LEFT JOIN assignment_executors executors ON executors.assignment_id = assignment.id
  INNER JOIN employee executor_employee ON executor_employee.id = executors.employee_id )

SELECT DISTINCT name FROM temp_executors;
    RETURN ref;
  END;'
LANGUAGE plpgsql;
```

```
@EnablePostgresDataTest
class NativeFunctionTest {

    @PersistenceContext private EntityManager entityManager;

    @Test
    @Sql("/stored_functions/magic_func.sql")
    void testStoredFunc() {
        StoredProcedureQuery query =
          entityManager.createStoredProcedureQuery("magic");

        query.execute();
        List resultList = query.getResultList();

        int result = (int) resultList.get(0);
        Assertions.assertThat(result).isEqualTo(1234);
    }
}
```

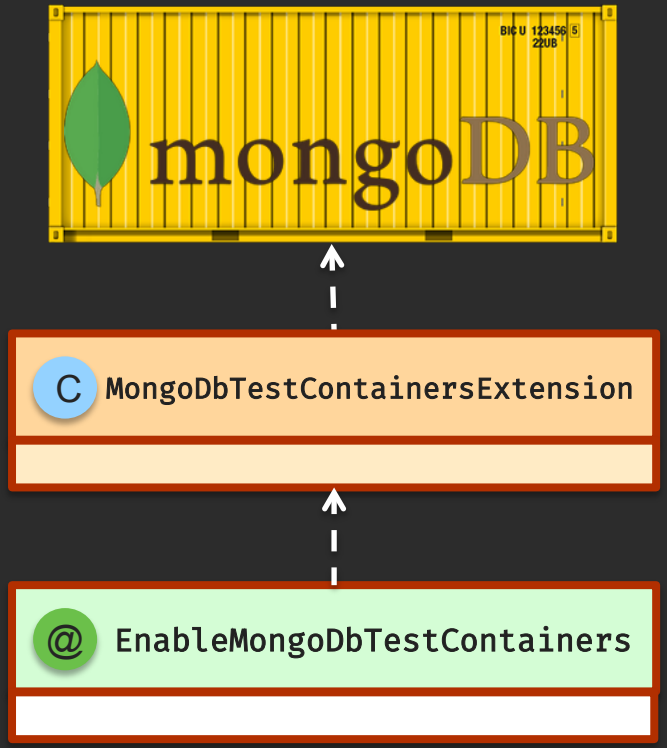
```
@EnablePostgresDataTest
class NativeFunctionTest {

    @PersistenceContext private EntityManager entityManager;

    @Test
    @Sql("/stored_functions/magic_func.sql")
    void testStoredFunc() {
        StoredProcedureQuery query =
          entityManager.createStoredProcedureQuery("magic");

        query.execute();
        List resultList = query.getResultList();

        int result = (int) resultList.get(0);
        Assertions.assertThat(result).isEqualTo(1234);
    }
}
```

// тестируем

валидация дата-сетов

агрегации в mongodb

// Тоже самое для MongoDB



// Тоже самое для MongoDB

C MongoDbTestContainersExtension

@ EnableMongoDbTestContainers

C MongoDbExtension

@ ExpectedMongoDataSet

@ ExpectedMongoDataSet

```
@MongoDbIntegrationTest
class MongoTest {

    @Autowired
    private MongoTemplate mongoTemplate;

    @Test
    @MongoDataSet(value = "/dataset/bar_dataset.json")
    void testFindById() {

        Bar bar = mongoTemplate.findById("55ffed", Bar.class);

        Assertions.assertThat(bar)
                  .isNotNull()
                  .extracting(Bar::getData)
                  .containsOnly(“A New Hope");
    }
}
```

```
@MongoDbIntegrationTest
class MongoTest {

    @Autowired
    private MongoTemplate mongoTemplate;

    @Test
    @MongoDataSet(value = "/dataset/bar_dataset.json")
    void testFindById() {

        Bar bar = mongoTemplate.findById("55ffed", Bar.class);

        Assertions.assertThat(bar)
                  .isNotNull()
                  .extracting(Bar::getData)
                  .containsOnly(“A New Hope");
    }
}
```

```
@MongoDbIntegrationTest
class MongoTest {

    @Autowired
    private MongoTemplate mongoTemplate;

    @Test
    @MongoDataSet(value = "/dataset/bar_dataset.json")
    void testFindById() {

        Bar bar = mongoTemplate.findById("55ffed", Bar.class);

        Assertions.assertThat(bar)
                  .isNotNull()
                  .extracting(Bar::getData)
                  .containsOnly(“A New Hope");
    }
}
```

```
{
  "com.antkorwin.springtestmongo.Bar": [
    {
      "id” : "18300a",
      "data” : "Wild Coyote"
    },
    {
      "id” : "55ffed",
      "data” : "A New Hope"
    }
  ]
}
```

```
@MongoDbIntegrationTest
class MongoTest {

    @Autowired
    private MongoTemplate mongoTemplate;

    @Test
    @MongoDataSet(value = "/dataset/bar_dataset.json")
    void testFindById() {

        Bar bar = mongoTemplate.findById("55ffed", Bar.class);

        Assertions.assertThat(bar)
                  .isNotNull()
                  .extracting(Bar::getData)
                  .containsOnly(“A New Hope");
    }
}
```

```
@MongoDbIntegrationTest
class MongoTest {

    @Autowired
    private MongoTemplate mongoTemplate;

    @Test
    @MongoDataSet(value = "/dataset/bar_dataset.json")
    void testFindById() {

        Bar bar = mongoTemplate.findById("55ffed", Bar.class);

        Assertions.assertThat(bar)
                  .isNotNull()
                  .extracting(Bar::getData)
                  .containsOnly(“A New Hope");
    }
}
```

```
@Test
@ExpectedMongoDataSet("dataset/expected_tasks.json")
void dateTimeNow() {

    Task firstTask = Task.builder()
                    .title("Turn to the dark side of the Force")
                    .dueDate(new Date())
                    .build();

    Task secondTask = Task.builder()
                    .title("Defeat the resistance forces")
                    .dueDate(tomorrow())
                    .build();

    mongoTemplate.save(firstTask);
    mongoTemplate.save(secondTask);
}
```

```
@Test
@ExpectedMongoDataSet("dataset/expected_tasks.json")
void dateTimeNow() {

    Task firstTask = Task.builder()
                    .title("Turn to the dark side of the Force")
                    .dueDate(new Date())
                    .build();

    Task secondTask = Task.builder()
                    .title("Defeat the resistance forces")
                    .dueDate(tomorrow())
                    .build();

    mongoTemplate.save(firstTask);
    mongoTemplate.save(secondTask);
}
```

```
@Test
@ExpectedMongoDataSet("dataset/expected_tasks.json")
void dateTimeNow() {

    Task firstTask = Task.builder()
                    .title("Turn to the dark side of the Force")
                    .dueDate(new Date())
                    .build();

    Task secondTask = Task.builder()
                    .title("Defeat the resistance forces")
                    .dueDate(tomorrow())
                    .build();

    mongoTemplate.save(firstTask);
    mongoTemplate.save(secondTask);
}
```

```
@Test
@ExpectedMongoDataSet("dataset/expected_tasks.json")
void dateTimeNow() {

    Task firstTask = Task.builder()
                    .title("Turn to the dark side of the Force")
                    .dueDate(new Date())
                    .build();

    Task secondTask = Task.builder()
                    .title("Defeat the resistance forces")
                    .dueDate(tomorrow())
                    .build();

    mongoTemplate.save(firstTask);
    mongoTemplate.save(secondTask);
}
```

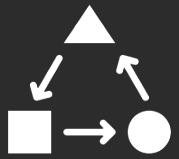
```
@Test
@ExpectedMongoDataSet("dataset/expected_tasks.json")
void dateTimeNow() {

    Task firstTask = Task.builder()
                    .title("Turn to the dark side of the Force")
                    .dueDate(new Date())
                    .build();

    Task secondTask = Task.builder()
                    .title("Defeat the resistance forces")
                    .dueDate(tomorrow())
                    .build();

    mongoTemplate.save(firstTask);
    mongoTemplate.save(secondTask);
}
```

```
{
  "com.antkorwin.starwars.mongotest.Task": [
    {
      "dueDate” : "date-match:[NOW]",
      "title" : "Turn to the dark side of the Force"
    },
    {
      ”dueDate” : "date-match:[NOW]+1(DAYS)",
      "title” : "Defeat the resistance forces"
    }
  ]
}
```

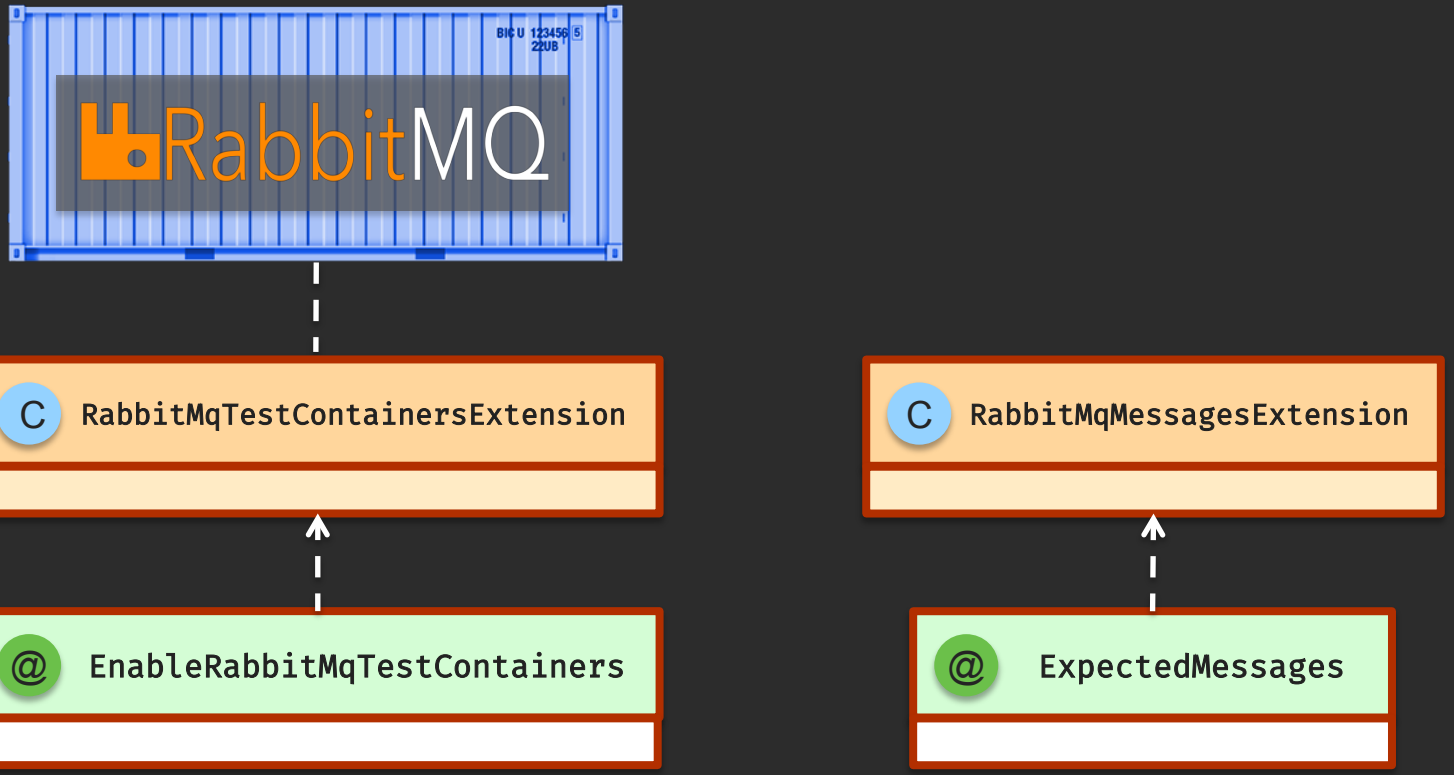
```
{
  "com.antkorwin.starwars.mongotest.Task": [
    {
      "dueDate” : "date-match:[NOW]",
      "title" : "Turn to the dark side of the Force"
    },
    {
      ”dueDate” : "date-match:[NOW]+1(DAYS)",
      "title” : "Defeat the resistance forces"
    }
  ]
}
```

// тестируем RabbitMQ

валидация дата-сетов



конвертацию данных


RabbitMQ
C
RabbitMqTestContainersExtension
@
EnableRabbitMqTestContainers
C
RabbitMqMessagesExtension
@
ExpectedMessages

```
@EnableRabbitMqTest
class StarWarsAsyncTest{

  @Autowired
  private AmqpTemplate amqpTemplate;

  @Test
  @ExpectedMessages(queue = ”build-queue",
                    messagesFile = “expected_messages.json")
  void testSendListOfMessages() {

    amqpTemplate.convertAndSend("build-queue",
                                new Dreadnought(20000));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Alpha", 3));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Bravo", 4));
}
```

```
@EnableRabbitMqTest
class StarWarsAsyncTest{

  @Autowired
  private AmqpTemplate amqpTemplate;

  @Test
  @ExpectedMessages(queue = ”build-queue",
                    messagesFile = “expected_messages.json")
  void testSendListOfMessages() {

    amqpTemplate.convertAndSend("build-queue",
                                new Dreadnought(20000));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Alpha", 3));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Bravo", 4));
}
```

```
@EnableRabbitMqTest
class StarWarsAsyncTest{

  @Autowired
  private AmqpTemplate amqpTemplate;

  @Test
  @ExpectedMessages(queue = ”build-queue",
                    messagesFile = “expected_messages.json")
  void testSendListOfMessages() {

    amqpTemplate.convertAndSend("build-queue",
                                new Dreadnought(20000));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Alpha", 3));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Bravo", 4));
}
```

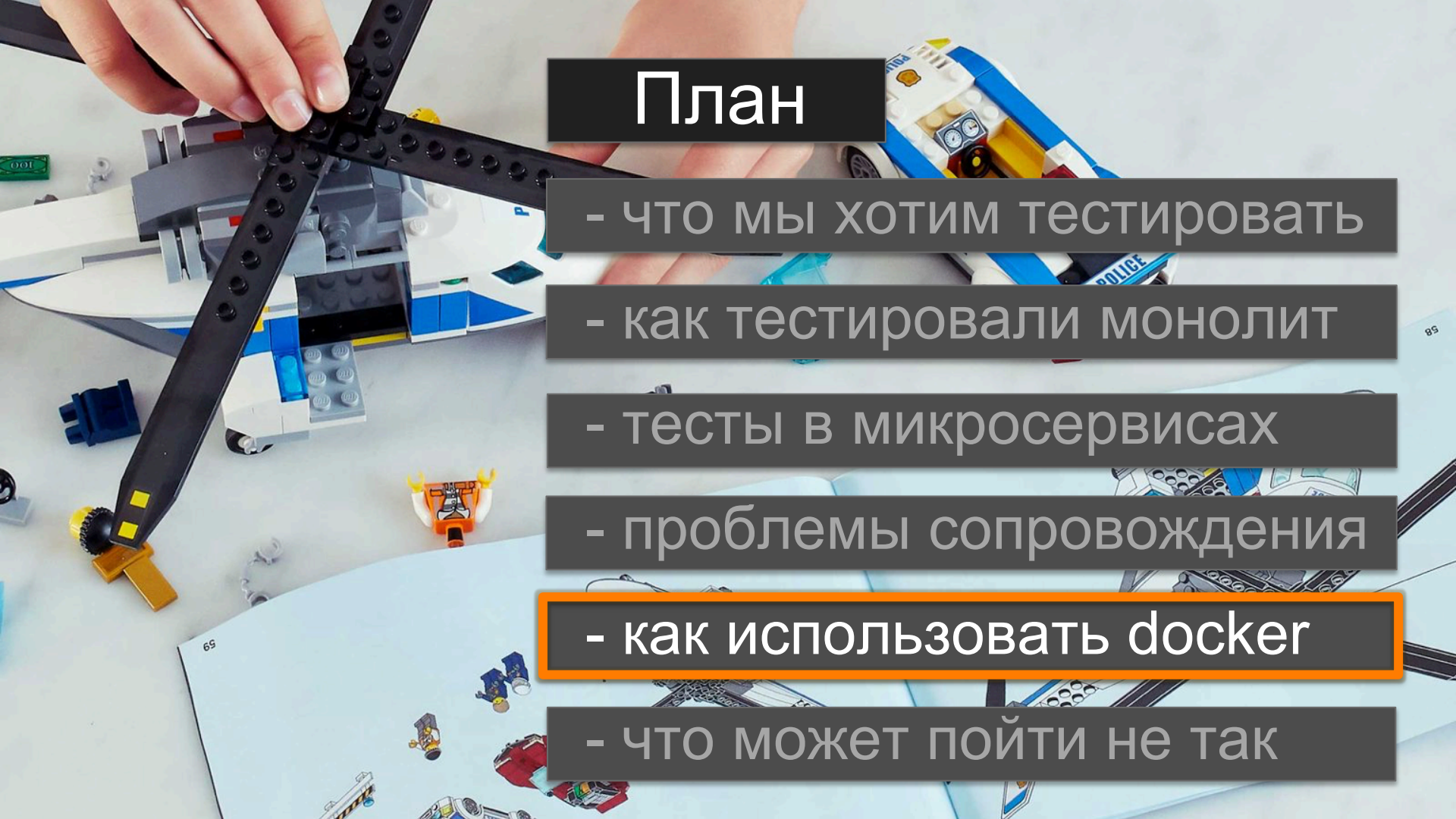
```
@EnableRabbitMqTest
class StarWarsAsyncTest{

  @Autowired
  private AmqpTemplate amqpTemplate;

  @Test
  @ExpectedMessages(queue = ”build-queue",
                    messagesFile = “expected_messages.json")
  void testSendListOfMessages() {

    amqpTemplate.convertAndSend("build-queue",
                                new Dreadnought(20000));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Alpha", 3));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Bravo", 4));
}
```

```
@EnableRabbitMqTest
class StarWarsAsyncTest{

  @Autowired
  private AmqpTemplate amqpTemplate;

  @Test
  @ExpectedMessages(queue = ”build-queue",
                    messagesFile = “expected_messages.json")
  void testSendListOfMessages() {

    amqpTemplate.convertAndSend("build-queue",
                                new Dreadnought(20000));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Alpha", 3));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Bravo", 4));
}
```

```
@EnableRabbitMqTest
class StarWarsAsyncTest{

  @Autowired
  private AmqpTemplate amqpTemplate;

  @Test
  @ExpectedMessages(queue = ”build-queue",
                    messagesFile = “expected_messages.json")
  void testSendListOfMessages() {

    amqpTemplate.convertAndSend("build-queue",
                                new Dreadnought(20000));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Alpha", 3));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Bravo", 4));
}
```

```
{
  "com.antkorwin.starwars.rabbitmq.Dreadnought": [
    {
      "stormtroopers” : "20000"
    }
  ],
  "com.antkorwin.starwars.rabbitmq.XWing":[
    {
      "name” : "Alpha",
      "stormtroopers” : 3
    },
    {
      "name” : "Bravo",
      "stormtroopers” : 4
    }
  ]
}
```

```
@EnableRabbitMqTest
class StarWarsAsyncTest{

  @Autowired
  private AmqpTemplate amqpTemplate;

  @Test
  @ExpectedMessages(queue = ”build-queue",
                    messagesFile = “expected_messages.json")
  void testSendListOfMessages() {

    amqpTemplate.convertAndSend("build-queue",
                                new Dreadnought(20000));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Alpha", 3));

    amqpTemplate.convertAndSend("build-queue",
                                new XWing("Bravo", 4));
}
```


План
- что мы хотим тестировать
- как тестировали монолит
- тесты в микросервисах
- проблемы сопровождения
- как использовать docker
- что может пойти не так

TestContainers

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@ExtendWith(PostgresTcExtension.class)
public @interface EnablePostgresTestContainers {

}
```

```
public class PostgresTcExtension implements Extension {

    static {
        PostgreSQLContainer postgres = new PostgreSQLContainer();
        postgres.start();

        System.setProperty("spring.datasource.driver-class-name",
                           postgres.getDriverClassName());
        System.setProperty("spring.datasource.url",
                           postgres.getJdbcUrl());
        System.setProperty("spring.datasource.username",
                           postgres.getUsername());
        System.setProperty("spring.datasource.password",
                           postgres.getPassword());
    }
}
```

```
public class PostgresTcExtension implements Extension {

    static {
        PostgreSQLContainer postgres = new PostgreSQLContainer();
        postgres.start();

        System.setProperty("spring.datasource.driver-class-name",
                           postgres.getDriverClassName());
        System.setProperty("spring.datasource.url",
                           postgres.getJdbcUrl());
        System.setProperty("spring.datasource.username",
                           postgres.getUsername());
        System.setProperty("spring.datasource.password",
                           postgres.getPassword());
    }
}
```

```
public class PostgresTcExtension implements Extension {

    static {
        PostgreSQLContainer postgres = new PostgreSQLContainer();
        postgres.start();

        System.setProperty("spring.datasource.driver-class-name",
                           postgres.getDriverClassName());
        System.setProperty("spring.datasource.url",
                           postgres.getJdbcUrl());
        System.setProperty("spring.datasource.username",
                           postgres.getUsername());
        System.setProperty("spring.datasource.password",
                           postgres.getPassword());
    }
}
```

// что не так?

```
public class PostgresTcExtension implements Extension {

    static {
        PostgreSQLContainer postgres = new PostgreSQLContainer();
        postgres.start();

        System.setProperty("spring.datasource.driver-class-name",
                           postgres.getDriverClassName());
        System.setProperty("spring.datasource.url",
                           postgres.getJdbcUrl());
        System.setProperty("spring.datasource.username",
                           postgres.getUsername());
        System.setProperty("spring.datasource.password",
                           postgres.getPassword());
    }
}
```

// зачем так?

```
public class PostgresTcExtension implements Extension {

    static {
        PostgreSQLContainer postgres = new PostgreSQLContainer();
        postgres.start();

        System.setProperty("spring.datasource.driver-class-name",
                           postgres.getDriverClassName());
        System.setProperty("spring.datasource.url",
                           postgres.getJdbcUrl());
        System.setProperty("spring.datasource.username",
                           postgres.getUsername());
        System.setProperty("spring.datasource.password",
                           postgres.getPassword());
    }
}
```

// зачем так?

```
public class PostgresTcExtension implements Extension {

    static {
        PostgreSQLContainer postgres = new PostgreSQLContainer();
        postgres.start();

        System.setProperty("spring.datasource.driver-class-name",
                           postgres.getDriverClassName());
        System.setProperty("spring.datasource.url",
                           postgres.getJdbcUrl());
        System.setProperty("spring.datasource.username",
                           postgres.getUsername());
        System.setProperty("spring.datasource.password",
                           postgres.getPassword());
    }
}
```

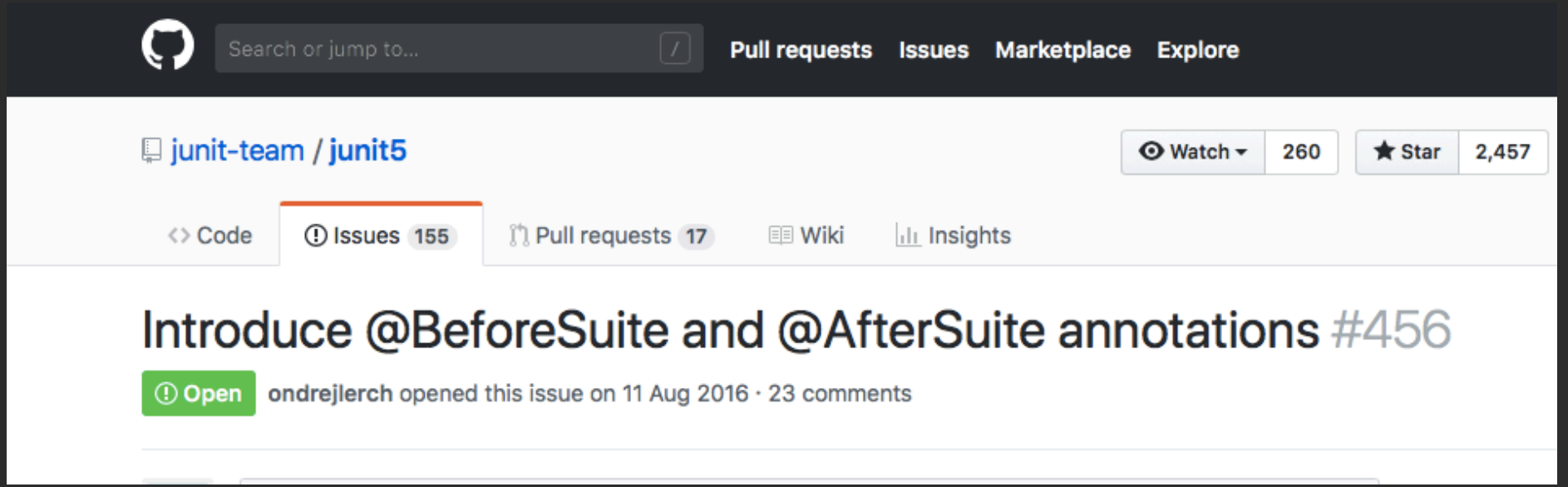
хотим один контейнер
на прогон всех тестов

Нужен BeforeSuite callback !

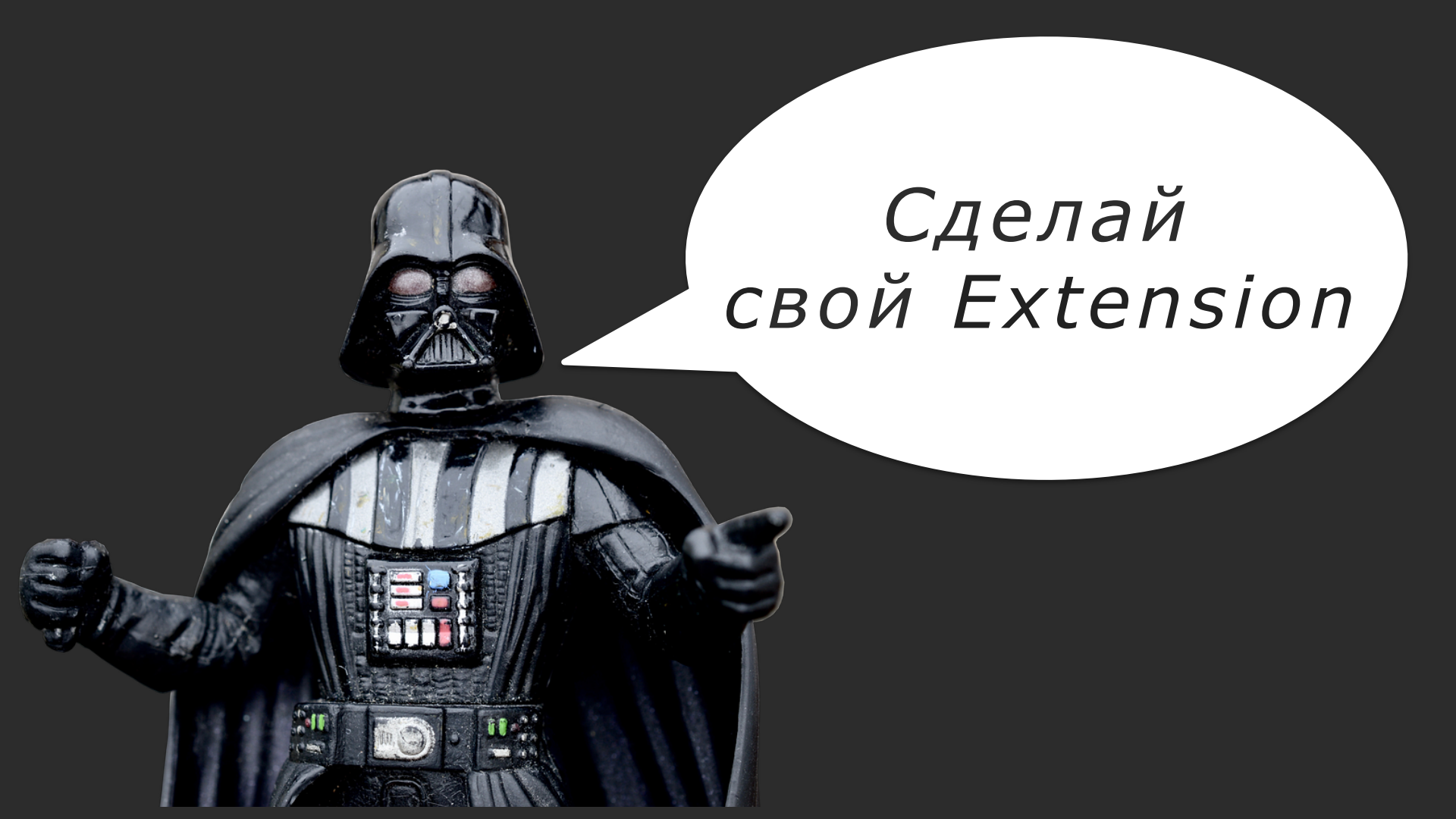
Поищем в JUnit5 ?

// нужен BeforeSuite callback

```
invokeBeforeEachCallbacks(context);
|       if (throwableCollector.isEmpty()) {
|               invokeBeforeEachMethods(context);
|               | if (throwableCollector.isEmpty()) {
|               |         invokeBeforeTestExecutionCallbacks(context);
|               |         | if (throwableCollector.isEmpty()) {
|               |         |       invokeTestMethod(context,
|               |         |                        dynamicTestExecutor);
|               |         | }
|               |         invokeAfterTestExecutionCallbacks(context);
|               | }
|               invokeAfterEachMethods(context);
|       }
invokeAfterEachCallbacks(context);
```



Возможно, скоро будет...


Сделай
свой Extension

```
public class PostgresTestExtension
                            implements BeforeAllCallback {

    @Override
    public void beforeAll(ExtensionContext context)
                                                throws Exception {

        if (containerAlreadyStarted(context)) return;
        if (!findTag(getTestDescriptor(context), TC_TAG)) return;

        startContainer();
        markAsStarted(context);
    }

}
```

```
public class PostgresTestExtension
                        implements BeforeAllCallback {

    @Override
    public void beforeAll(ExtensionContext context)
                                                throws Exception {

        if (containerAlreadyStarted(context)) return;
        if (!findTag(getTestDescriptor(context), TC_TAG)) return;

        startContainer();
        markAsStarted(context);
    }

}
```

```
public class PostgresTestExtension
                            implements BeforeAllCallback {

    @Override
    public void beforeAll(ExtensionContext context)
                                                  throws Exception {

        if (containerAlreadyStarted(context)) return;
        if (!findTag(getTestDescriptor(context), TC_TAG)) return;

        startContainer();
        markAsStarted(context);
    }

}
```

```
public class PostgresTestExtension
                            implements BeforeAllCallback {

    @Override
    public void beforeAll(ExtensionContext context)
                                                  throws Exception {

        if (containerAlreadyStarted(context)) return;
        if (!findTag(getTestDescriptor(context), TC_TAG)) return;

        startContainer();
        markAsStarted(context);
    }

}
```

```
public class PostgresTestExtension
                            implements BeforeAllCallback {

    @Override
    public void beforeAll(ExtensionContext context)
                                                throws Exception {

        if (containerAlreadyStarted(context)) return;
        if (!findTag(getTestDescriptor(context), TC_TAG)) return;

        startContainer();
        markAsStarted(context);
    }

}
```

```
public class PostgresTestExtension
                            implements BeforeAllCallback {

    @Override
    public void beforeAll(ExtensionContext context)
                                                  throws Exception {

        if (containerAlreadyStarted(context)) return;
        if (!findTag(getTestDescriptor(context), TC_TAG)) return;

        startContainer();
        markAsStarted(context);
    }

}
```

// используем ExtensionContext

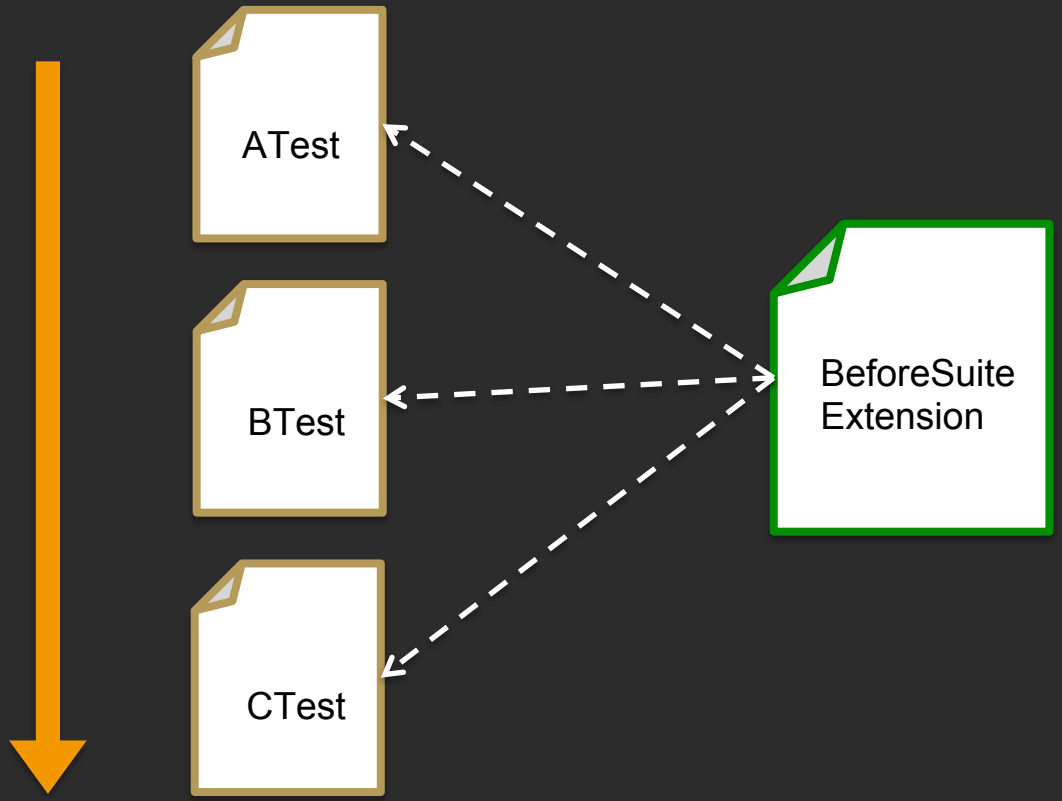
```
private void markAsStarted(ExtensionContext context) {

    context.getRoot()
           .getStore(NAMESPACE)
           .put(TESTCONTAINERS_PG_TAG, true);
}
```

Шарим данные между:

- тестовыми методами
- тестовыми классами

А удалять контейнеры кто будет?

// а как подключить Extension ко всем тестам?



// делаем глобальный Extension

/resources/META-INF/services/
    org.junit.jupiter.api.extension.Extension

    com.....PostgresTestExtension

/resources/
    junit-platform.properties

```
junit.jupiter.extensions.autodetection.enabled = true
```

static init. block

VS

custom before suite

static init. block

✗ less clean code

custom before suite

✓ more clean code

static init. block

- ✗ less clean code
- ✓ junit4 и junit5

custom before suite

- ✓ more clean code
- ✗ только junit5

static init. block

- ✗ less clean code
- ✓ junit4 и junit5
- ✓ прост в написании

custom before suite

- ✓ more clean code
- ✗ только junit5
- ✗ сложная конструкция

static init. block

- ✗ less clean code
- ✓ junit4 и junit5
- ✓ прост в написании
- ✓ легко компоновать

custom before suite

- ✓ more clean code
- ✗ только junit5
- ✗ сложная конструкция
- ✗ переопределение

`junit-platform.properties`


DEMO

// Выводы

С JUnit5 можно уменьшить дублирование кода в микросервисных проектах

Мета-аннотации помогают собрать нужную конфигурацию теста, как из конструктора

Вместе с TestContainers можно использовать production-окружение в тестах

// Выводы

С JUnit5 можно уменьшить дублирование кода в микросервисных проектах

Мета-аннотации помогают собрать нужную конфигурацию теста, как из конструктора

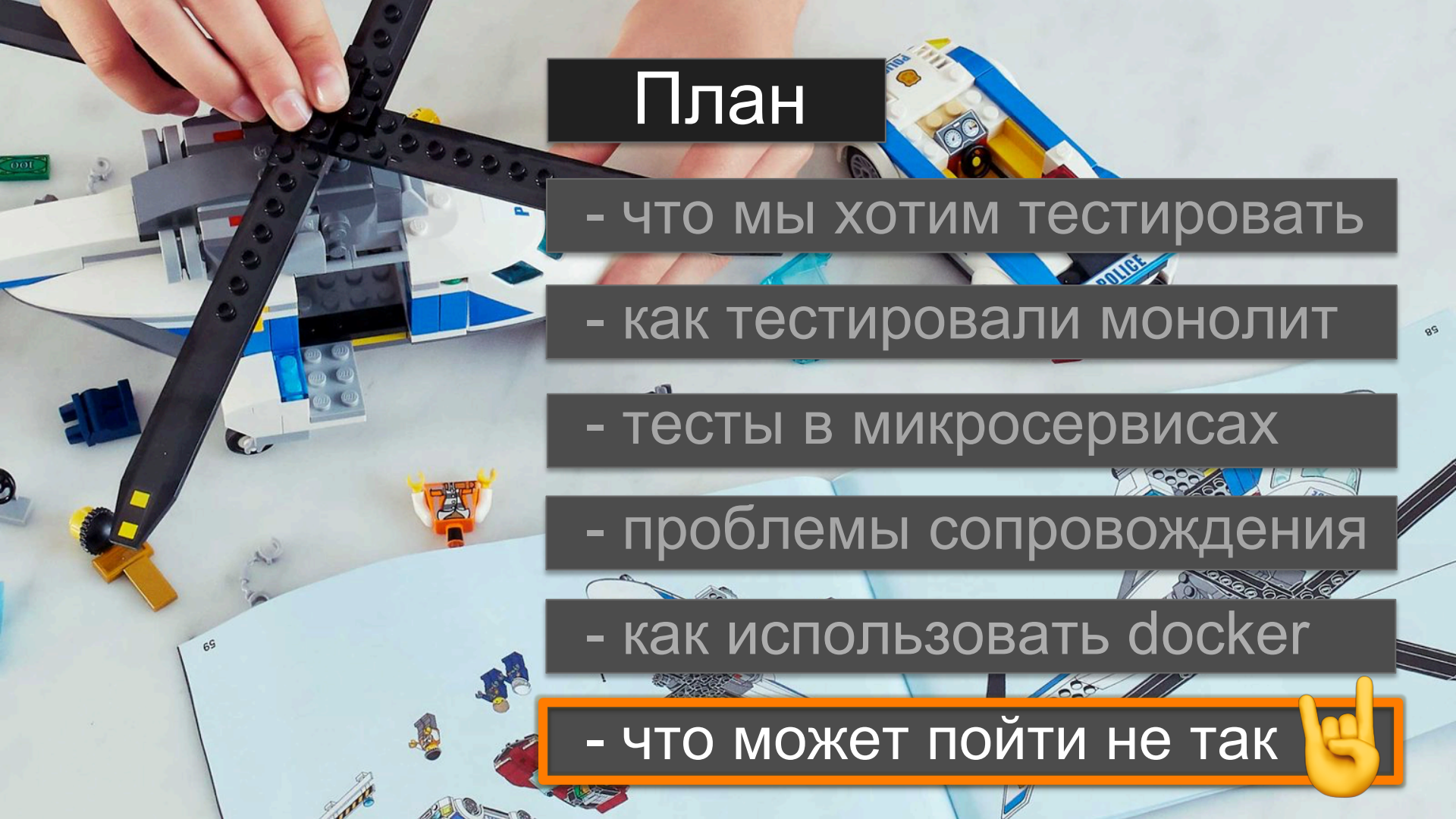
Вместе с TestContainers можно использовать production-окружение в тестах

// Выводы

С JUnit5 можно уменьшить дублирование кода в микросервисных проектах

Мета-аннотации помогают собрать нужную конфигурацию теста, как из конструктора

Вместе с TestContainers можно использовать production-окружение в тестах


План
- что мы хотим тестировать
- как тестировали монолит
- тесты в микросервисах
- проблемы сопровождения
- как использовать docker
- что может пойти не так

Не работает:

Работает:

Темная сторона

Context Caching

ATest

BTest

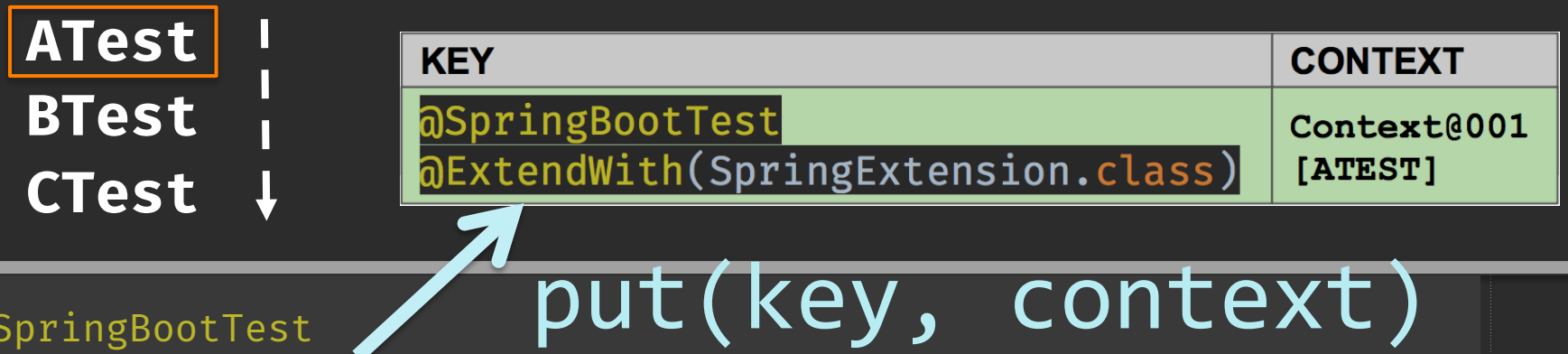
CTest

KEY	CONTEXT
EMPTY_CACHE	

ATest
BTest
CTest

KEY	CONTEXT
EMPTY_CACHE	

```
@SpringBootTest
@ExtendWith(SpringExtension.class)
class ATest {

    @Autowired
    private TemplateService templateService;

    @Test
    void build() {
        String result = templateService.build("%d + %d = %d",
                                              1,   2,   3);

        assertThat(result).isEqualTo("1 + 2 = 3");
    }
```




```
@SpringBootTest
@ExtendWith(SpringExtension.class)
class ATest {

    @Autowired
    private TemplateService templateService;

    @Test
    void build() {
        String result = templateService.build("%d + %d = %d",
                                              1,   2,   3);

        assertThat(result).isEqualTo("1 + 2 = 3");
    }
```

ATest
BTest
CTest

KEY	CONTEXT
@SpringBootTest @ExtendWith(SpringExtension.class)	Context@001 [ATEST]

```
@SpringBootTest
@ExtendWith(SpringExtension.class)
class ATest {

    @Autowired
    private TemplateService templateService;

    @Test
    void build() {
        String result = templateService.build("%d + %d = %d",
                                              1,   2,   3);

        assertThat(result).isEqualTo("1 + 2 = 3");
    }
```

KEY	CONTEXT
@SpringBootTest @ExtendWith(SpringExtension.class)	Context@001 [ATEST]

get(key)

```
@DataJpaTest
@ExtendWith(SpringExtension.class)
class BTest {

    @Test
    void createTest() {
        ...
    }

}
```

KEY	CONTEXT
@SpringBootTest @ExtendWith(SpringExtension.class)	Context@001 [ATEST]
@DataJpaTest @ExtendWith(SpringExtension.class)	**Context@002** **[BTEST]**

put(key, context)

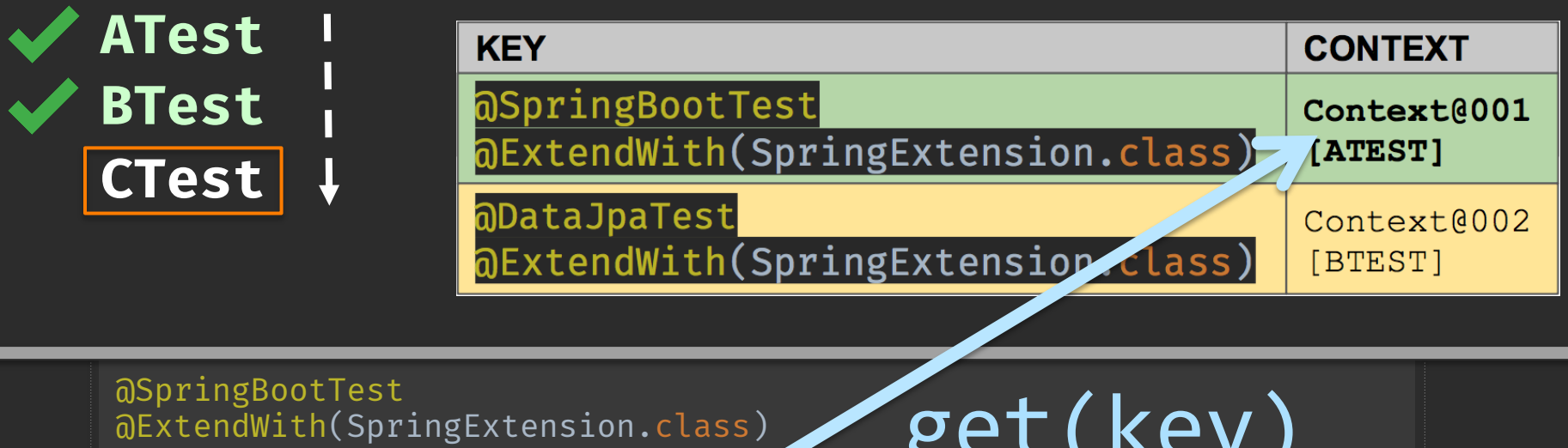
```
@DataJpaTest
@ExtendWith(SpringExtension.class)
class BTest {

    @Test
    void createTest() {
        ...
    }

}
```

KEY	CONTEXT
@SpringBootTest @ExtendWith(SpringExtension.class)	Context@001 [ATEST]
@DataJpaTest @ExtendWith(SpringExtension.class)	**Context@002** **[BTEST]**

```
@DataJpaTest
@ExtendWith(SpringExtension.class)
class BTest {

    @Test
    void createTest() {
        ...
    }

}
```

KEY	CONTEXT
@SpringBootTest @ExtendWith(SpringExtension.class)	Context@001 [ATEST]
@DataJpaTest @ExtendWith(SpringExtension.class)	Context@002 [BTEST]

```java
@SpringBootTest
@ExtendWith(SpringExtension.class)
@EnableRabbitMqTestContainers
class CTest {

    @Autowired
    private DeliveryService deliveryService;

    @Test
    @ExpectedMessage(queue = "test-queue", message = "secret text")
    void sendMessageTest() {
        deliveryService.sendMessage("secret text");
    }
}
```




```
@SpringBootTest
@ExtendWith(SpringExtension.class)
@EnableRabbitMqTestContainers
class CTest {

    @Autowired
    private DeliveryService deliveryService;

    @Test
    @ExpectedMessage(queue = ”test-queue", message = "secret text”)
    void sendMessageTest() {
        deliveryService.sendMessage("secret text");
    }
}
```

KEY	CONTEXT
@SpringBootTest @ExtendWith(SpringExtension.class)	Context@001 [ATEST]
@DataJpaTest @ExtendWith(SpringExtension.class)	Context@002 [BTEST]

```
@SpringBootTest
@ExtendWith(SpringExtension.class)
@EnableRabbitMqTestContainers
class CTest {

    @Autowired
    private DeliveryService deliveryService;

    @Test
    @ExpectedMessage(queue = ”test-queue", message = "secret text”)
    void sendMessageTest() {
        deliveryService.sendMessage("secret text");
    }
}
```

KEY	CONTEXT
@SpringBootTest @ExtendWith(SpringExtension.class)	Context@001 [ATEST]
@DataJpaTest @ExtendWith(SpringExtension.class)	Context@002 [BTEST]

Tests failed: 1, passed: 1 of 2 tests – 210 ms

```
org.springframework.amqp.AmqpConnectException: java.net.ConnectException: Connection refused (Connection refused)

    at org.springframework.amqp.rabbit.support.RabbitExceptionTranslator.convertRabbitAccessException(RabbitExceptionTra
    at org.springframework.amqp.rabbit.connection.AbstractConnectionFactory.createBareConnection(AbstractConnectionFacto
    at org.springframework.amqp.rabbit.connection.CachingConnectionFactory.createConnection(CachingConnectionFactory.jav
    at org.springframework.amqp.rabbit.connection.ConnectionFactoryUtils.createConnection(ConnectionFactoryUtils.java:21
    at org.springframework.amqp.rabbit.core.RabbitTemplate.doExecute(RabbitTemplate.java:1984)
    at org.springframework.amqp.rabbit.core.RabbitTemplate.execute(RabbitTemplate.java:1958)
    at org.springframework.amqp.rabbit.core.RabbitTemplate.send(RabbitTemplate.java:907)
    at org.springframework.amqp.rabbit.core.RabbitTemplate.convertAndSend(RabbitTemplate.java:973)
    at org.springframework.amqp.rabbit.core.RabbitTemplate.convertAndSend(RabbitTemplate.java:955)
    at com.jupiter.tools.demo.contextcache.service.delivery.DeliveryServiceImpl.sendMessage(DeliveryServiceImpl.java:20
    at com.jupiter.tools.demo.contextcache.service.FTest.sendMessageTest(FTest.java:24) <15 internal calls>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <5 internal calls>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <17 internal calls>
    Suppressed: org.springframework.amqp.AmqpConnectException: java.net.ConnectException: Connection refused (Connectio
        at org.springframework.amqp.rabbit.support.RabbitExceptionTranslator.convertRabbitAccessException(RabbitExceptio
```

```
@SpringBootTest
@ExtendWith(SpringExtension.class)
@EnableRabbitMqTestContainers
class CTest {

    ...
```

```
@SpringBootTest
@ExtendWith(SpringExtension.class)
@EnableRabbitMqTestContainers
class CTest {

    ...
```

```java
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@ExtendWith(RabbitMqTcExtension.class)
public @interface EnableRabbitMqTestContainers {
}
```

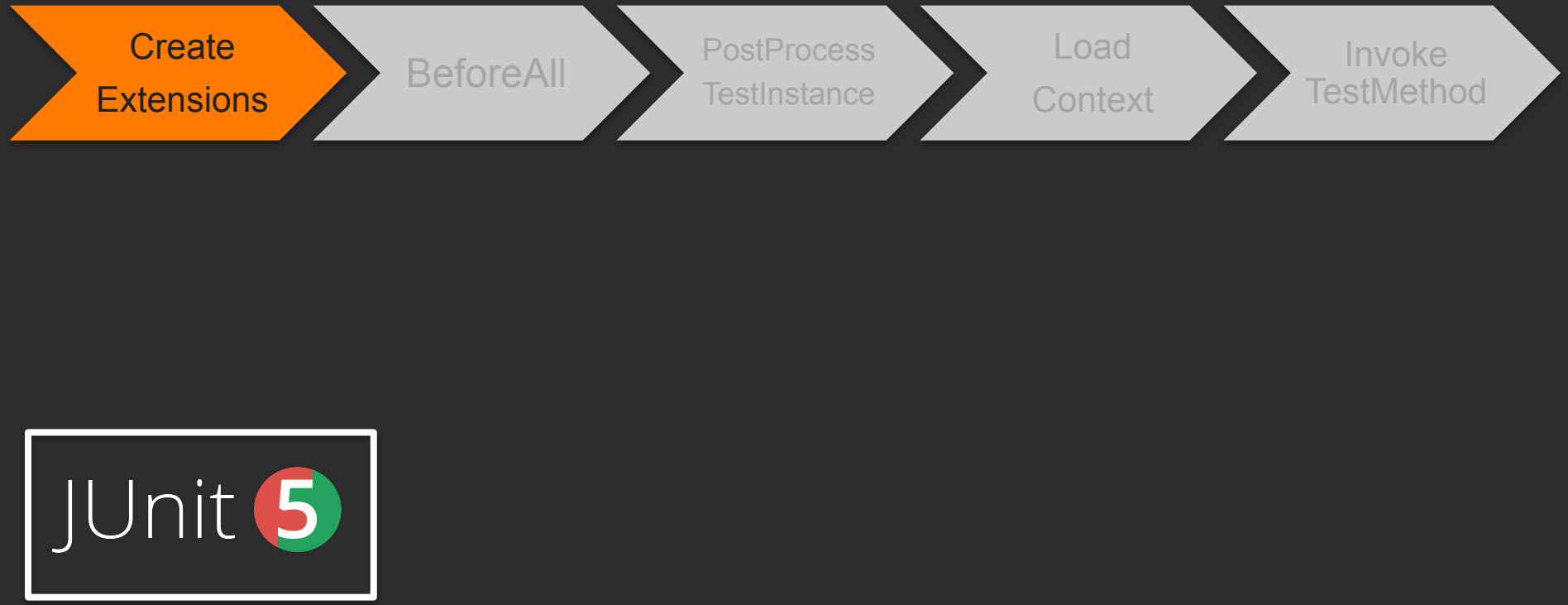
```java
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@ExtendWith(RabbitMqTcExtension.class)
public @interface EnableRabbitMqTestContainers {
}
```

```
public class RabbitMqTcExtension implements Extension {

    private static final Integer RABBIT_PORT = 5672;
    private static GenericContainer rabbitmq =
            new GenericContainer("rabbitmq:management")
                    .withExposedPorts(RABBIT_PORT, 15672);

    static {
        rabbitmq.start();
        System.setProperty("spring.rabbitmq.host",
                           rabbitmq.getContainerIpAddress());
        System.setProperty("spring.rabbitmq.port",
                           rabbitmq.getMappedPort(RABBIT_PORT).toString());
    }
}
```

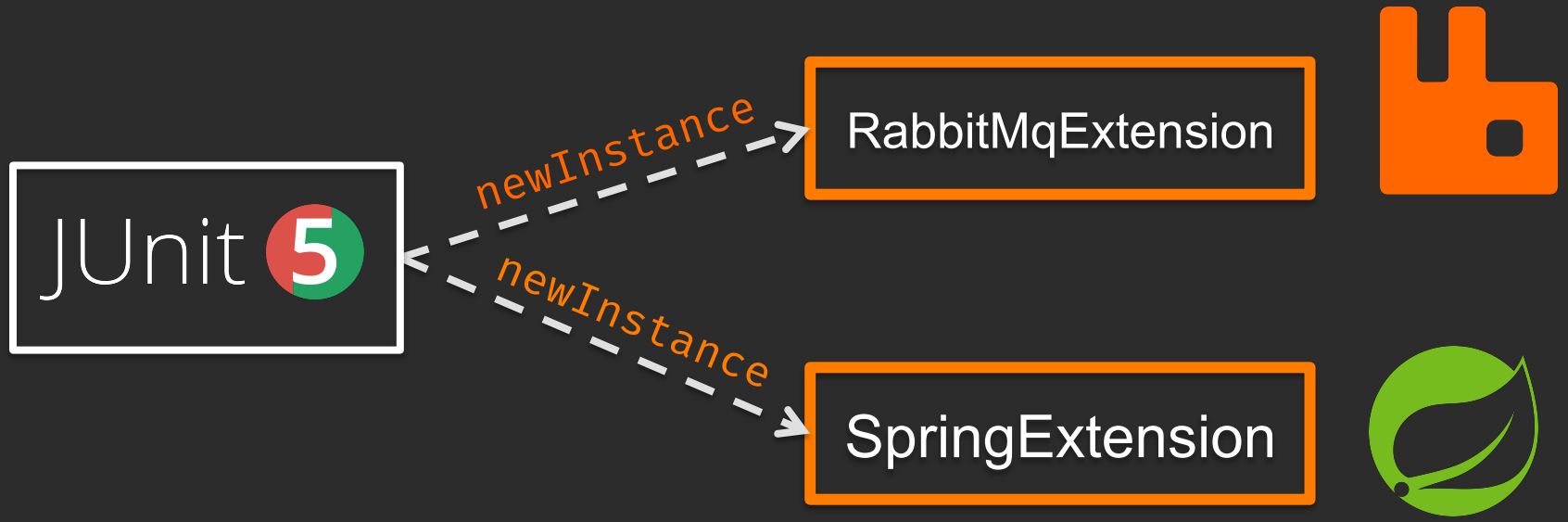
```java
public class RabbitMqTcExtension implements Extension {

    private static final Integer RABBIT_PORT = 5672;
    private static GenericContainer rabbitmq =
            new GenericContainer("rabbitmq:management")
                    .withExposedPorts(RABBIT_PORT, 15672);

    static {
        rabbitmq.start();
        System.setProperty("spring.rabbitmq.host",
                rabbitmq.getContainerIpAddress());
        System.setProperty("spring.rabbitmq.port",
                rabbitmq.getMappedPort(RABBIT_PORT).toString());
    }
}
```

```
public class RabbitMqTcExtension implements Extension {

    private static final Integer RABBIT_PORT = 5672;
    private static GenericContainer rabbitmq =
            new GenericContainer("rabbitmq:management")
                    .withExposedPorts(RABBIT_PORT, 15672);

    static {
        rabbitmq.start();
        System.setProperty("spring.rabbitmq.host",
                           rabbitmq.getContainerIpAddress());
        System.setProperty("spring.rabbitmq.port",
                           rabbitmq.getMappedPort(RABBIT_PORT).toString());
    }
}
```

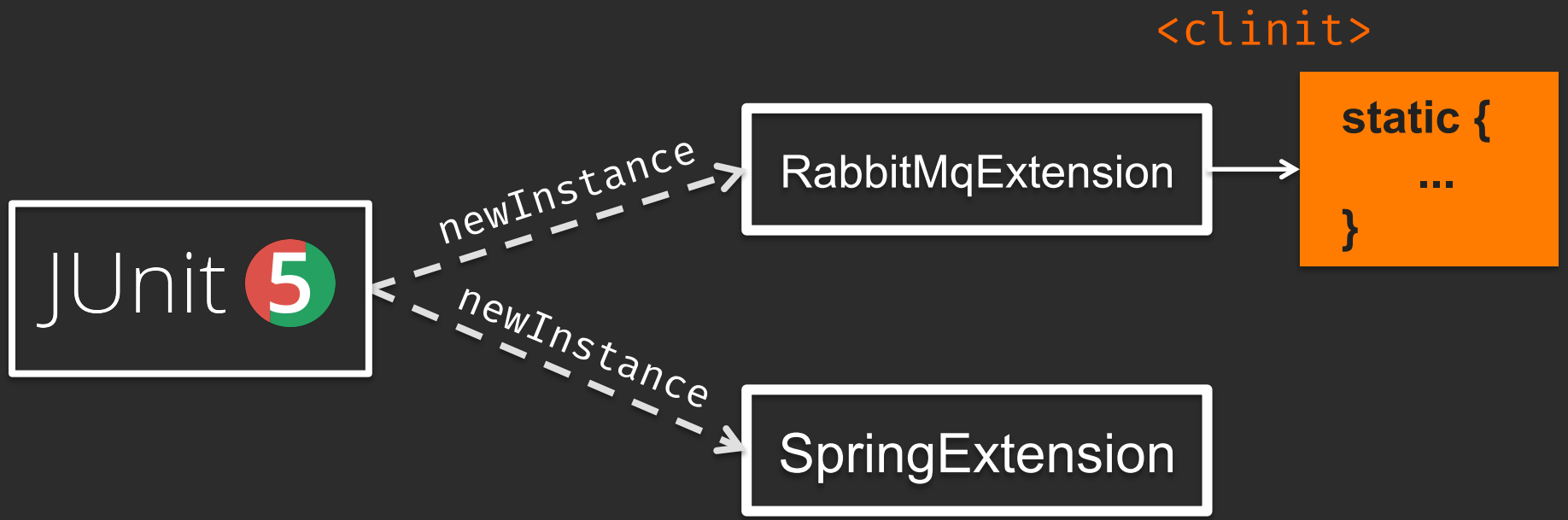
Как работает Spring и JUnit


Create Extensions
BeforeAll
PostProcess TestInstance
Load Context
Invoke TestMethod
JUnit 5

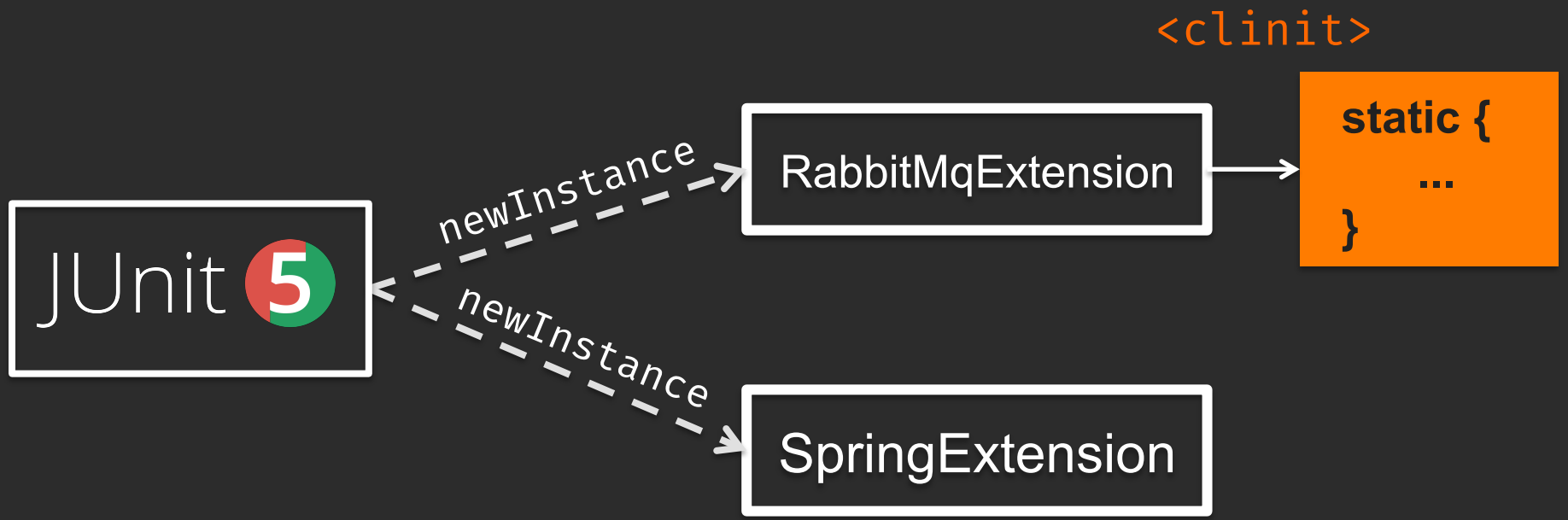
Create Extensions | BeforeAll | PostProcess TestInstance | Load Context | Invoke TestMethod

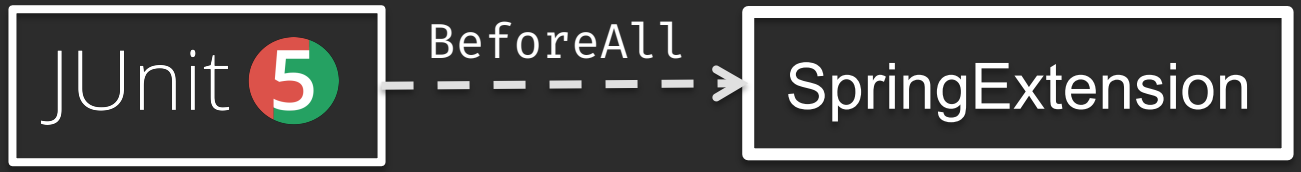


Create Extensions → BeforeAll → PostProcess TestInstance → Load Context → Invoke TestMethod

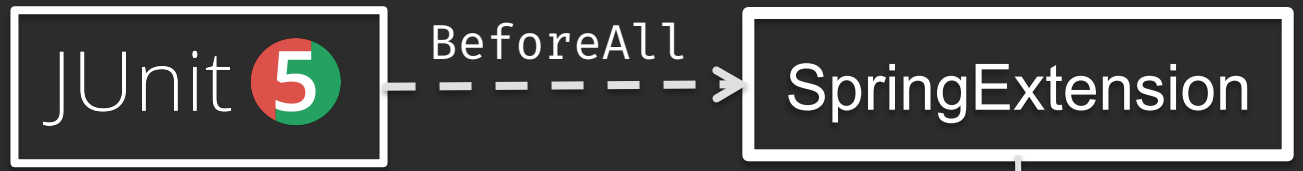


Create Extensions → BeforeAll → PostProcess TestInstance → Load Context → Invoke TestMethod



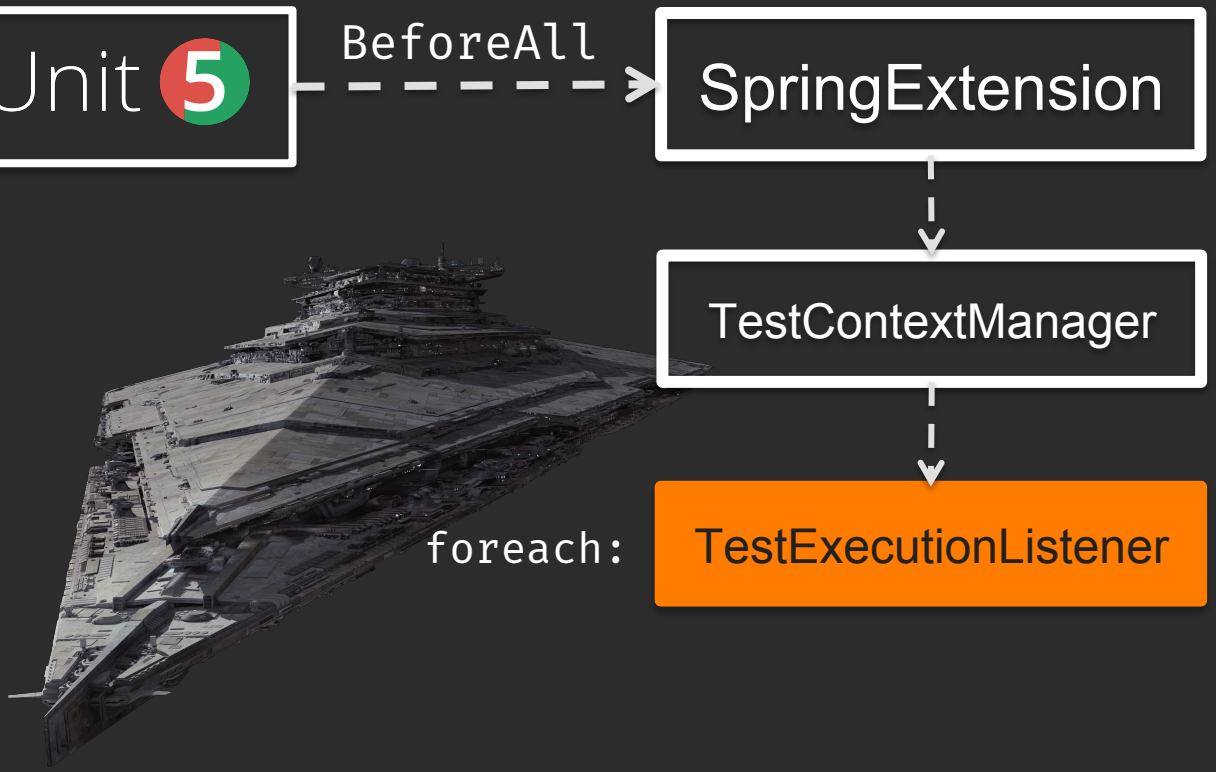

JUnit 5
BeforeAll
SpringExtension

Create Extensions | **BeforeAll** | PostProcess TestInstance | Load Context | Invoke TestMethod



TestContextManager

Create Extensions | **BeforeAll** | PostProcess TestInstance | Load Context | Invoke TestMethod



Create Extensions → **BeforeAll** → PostProcess TestInstance → Load Context → Invoke TestMethod

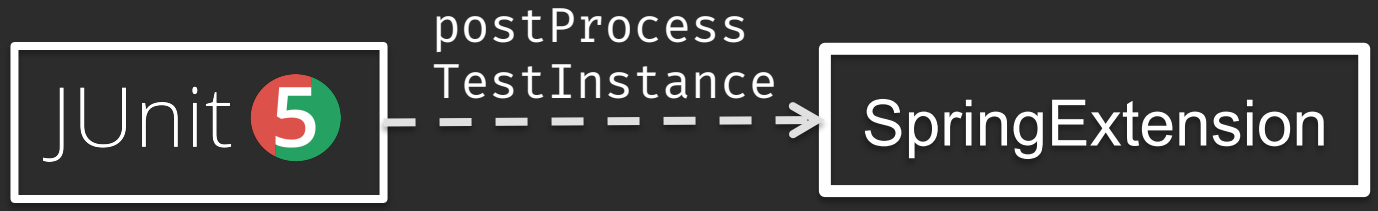
JUnit 5 — `BeforeAll` - - - - - -> SpringExtension

SpringExtension

TestContextManager

`foreach:` TestExecutionListener

`beforeTestClass:` DirtiesContextBeforeModesTestExecutionListener


JUnit 5
postProcess
TestInstance
SpringExtension

Create Extensions → BeforeAll → **PostProcess TestInstance** → Load Context → Invoke TestMethod

JUnit 5 --- `postProcessTestInstance` ---> SpringExtension

SpringExtension ↓

TestContextManager

Create Extensions → BeforeAll → **PostProcess TestInstance** → Load Context → Invoke TestMethod

JUnit 5 --- `postProcessTestInstance` ---> SpringExtension

SpringExtension → TestContextManager → TestExecutionListener

`foreach:`

Create Extensions → BeforeAll → **PostProcess TestInstance** → Load Context → Invoke TestMethod

JUnit 5 --- `postProcessTestInstance` ---> SpringExtension

SpringExtension → TestContextManager → TestExecutionListener → DependencyInjectionTestExecutionListener

`prepareTestInstance:`

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

DependencyInjectionTestExecutionListener

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

DependencyInjectionTestExecutionListener

↓ loadContext

DefaultCacheAwareContextLoaderDelegate

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

DependencyInjectionTestExecutionListener

loadContext

DefaultCacheAwareContextLoaderDelegate

get

loadContext

ContextCache

SpringBootContextLoader

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

DependencyInjectionTestExecutionListener

loadContext

DefaultCacheAwareContextLoaderDelegate

get

loadContext

ContextCache

SpringBootContextLoader

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

DependencyInjectionTestExecutionListener

loadContext

DefaultCacheAwareContextLoaderDelegate

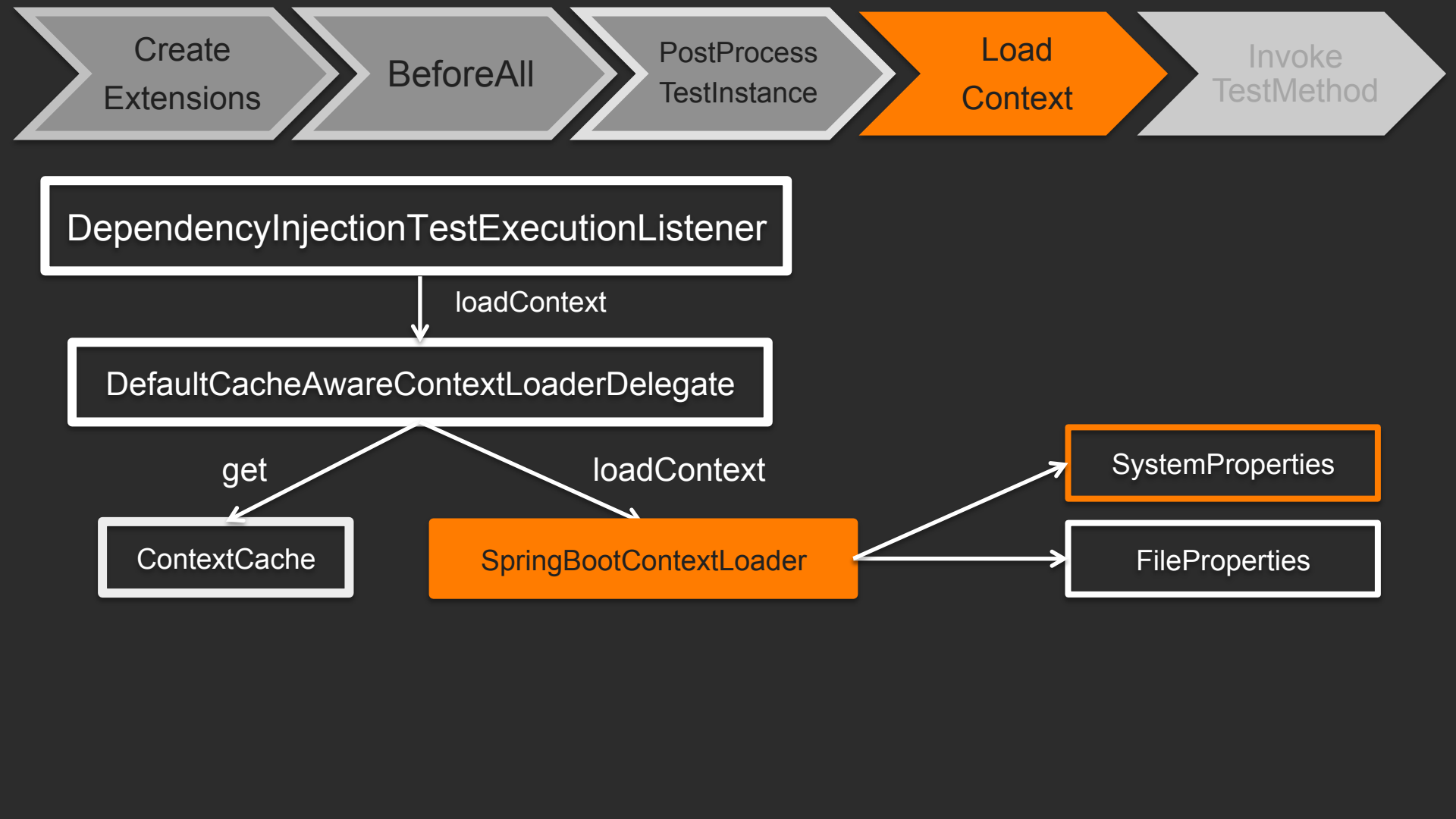
get

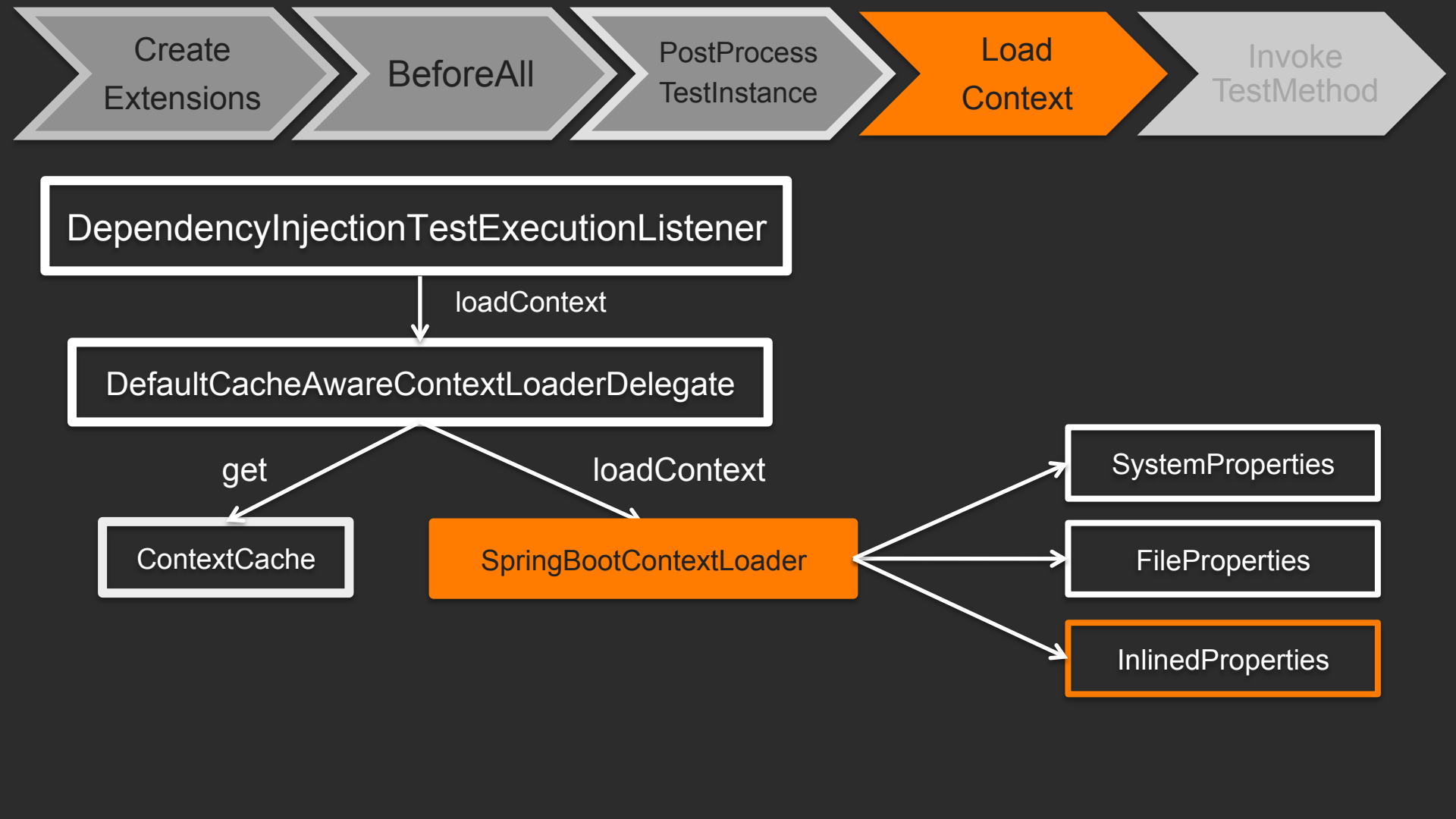
loadContext

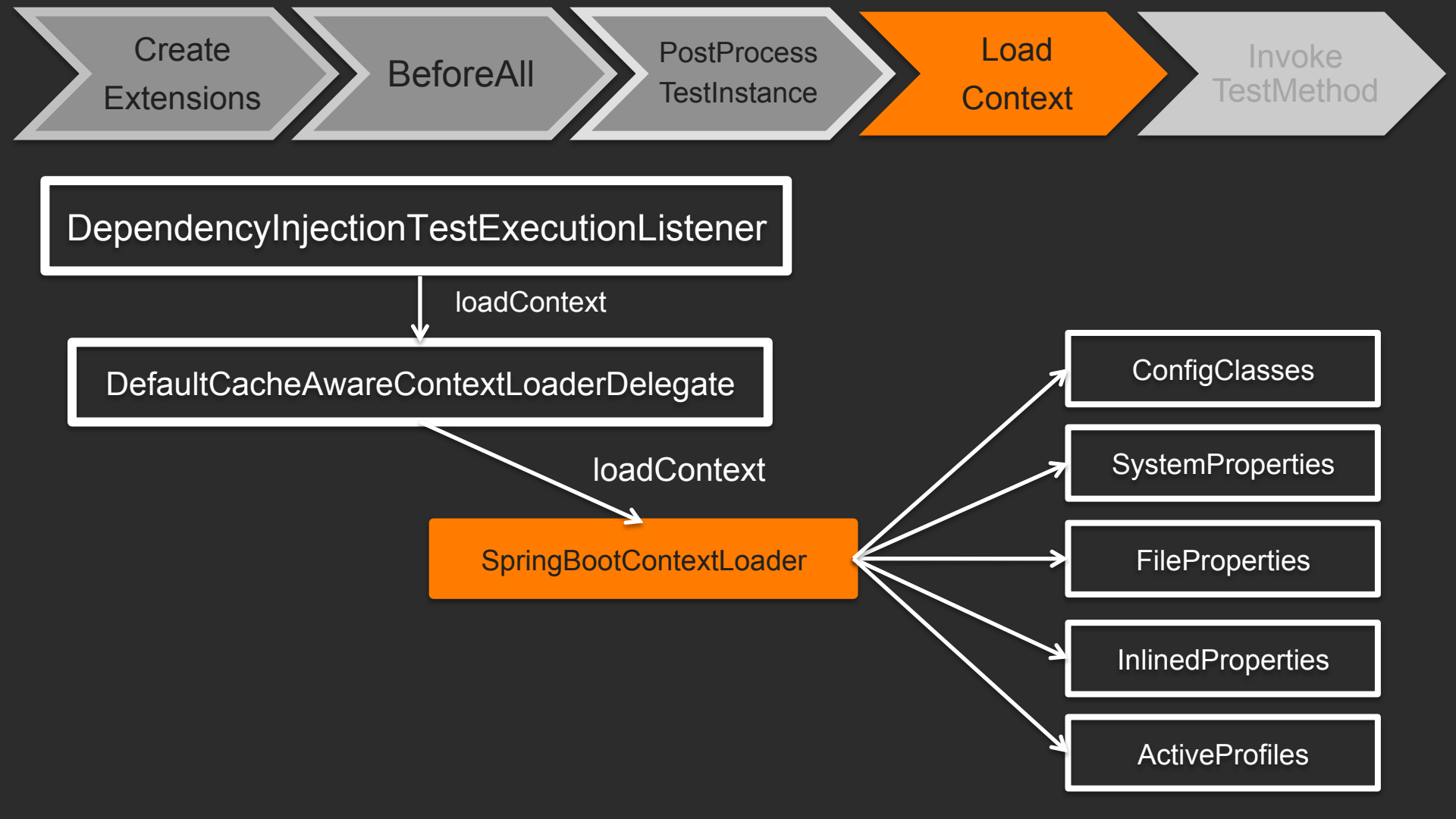
ContextCache

SpringBootContextLoader

FileProperties


Create Extensions
BeforeAll
PostProcess TestInstance
Load Context
Invoke TestMethod
DependencyInjectionTestExecutionListener
loadContext
DefaultCacheAwareContextLoaderDelegate
get
loadContext
ContextCache
SpringBootContextLoader
SystemProperties
FileProperties


Create Extensions
BeforeAll
PostProcess TestInstance
Load Context
Invoke TestMethod
DependencyInjectionTestExecutionListener
loadContext
DefaultCacheAwareContextLoaderDelegate
get
loadContext
ContextCache
SpringBootContextLoader
SystemProperties
FileProperties
InlinedProperties


Create Extensions
BeforeAll
PostProcess TestInstance
Load Context
Invoke TestMethod
DependencyInjectionTestExecutionListener
loadContext
DefaultCacheAwareContextLoaderDelegate
loadContext
SpringBootContextLoader
ConfigClasses
SystemProperties
FileProperties
InlinedProperties
ActiveProfiles

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

DependencyInjectionTestExecutionListener

loadContext

DefaultCacheAwareContextLoaderDelegate

loadContext

SpringBootContextLoader

run

SpringApplication

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

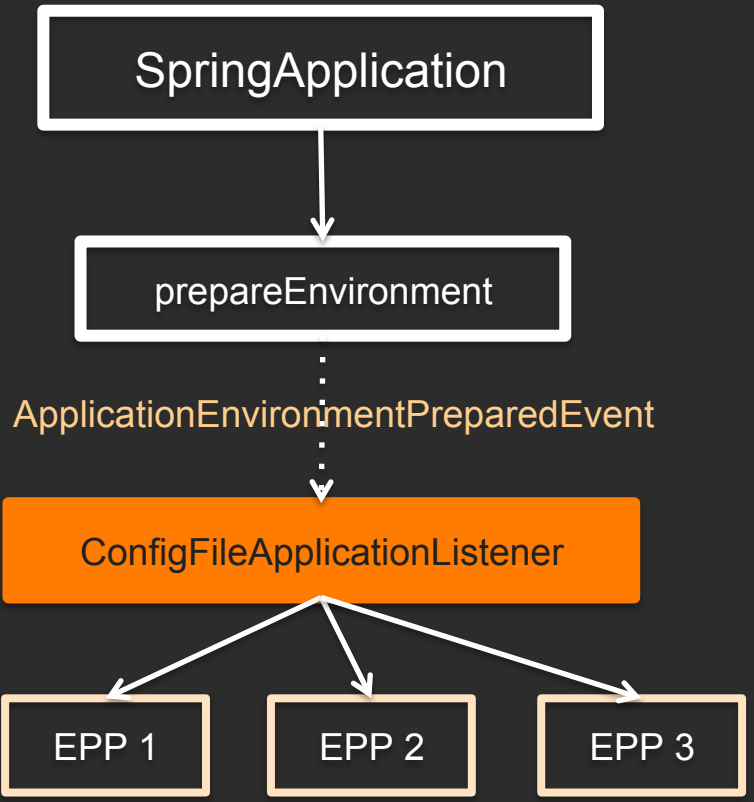
SpringApplication

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

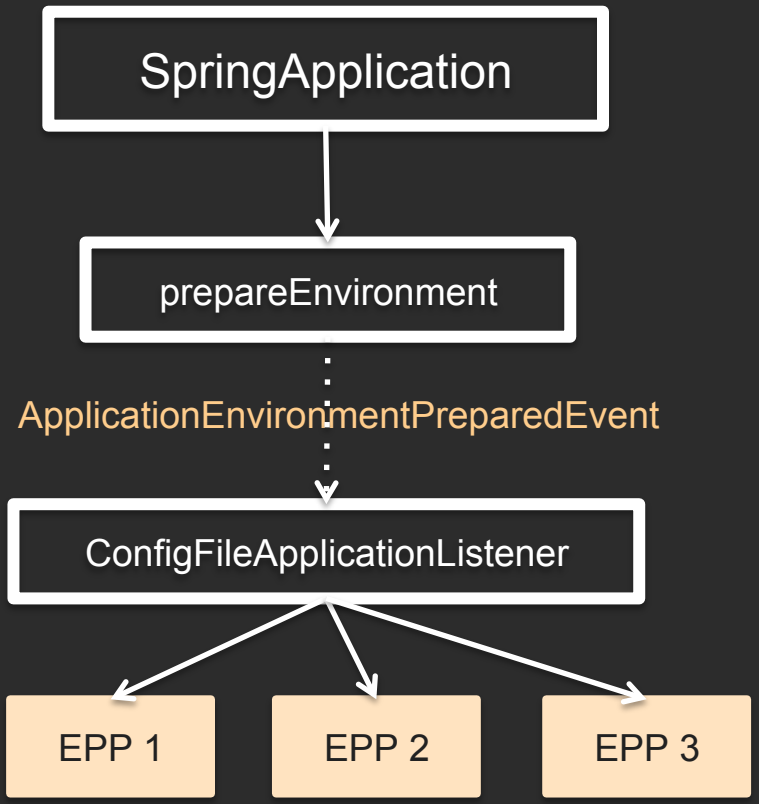
↓

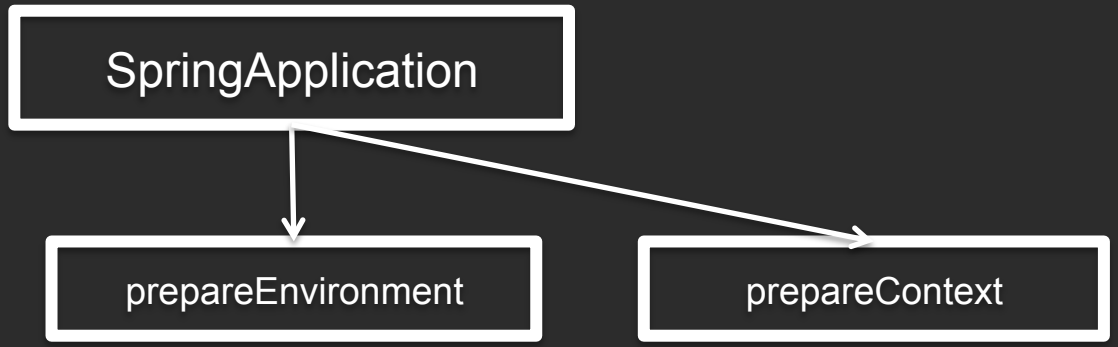
prepareEnvironment


SpringApplication
prepareEnvironment
ApplicationEnvironmentPreparedEvent
ConfigFileApplicationListener
EPP 1
EPP 2
EPP 3

// The Force here:

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod




SpringApplication
prepareEnvironment
prepareContext

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

- prepareEnvironment
- prepareContext

Magic!!!

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

```
2019-03-30 10:05:09.237  INFO   --- [           main] o.s.t.c.support.Ab
10:05:09.508 [main] INFO  o.s.b.t.c.SpringBootTestContextBootstrapper -
2019-03-30 10:05:09.508  INFO   --- [           main] .b.t.c.SpringBootT
10:05:09.680 [main] INFO  o.s.b.t.c.SpringBootTestContextBootstrapper -
2019-03-30 10:05:09.680  INFO   --- [           main] .b.t.c.SpringBootT
10:05:09.710 [main] INFO  o.s.b.t.c.SpringBootTestContextBootstrapper -
2019-03-30 10:05:09.710  INFO   --- [           main] .b.t.c.SpringBootT

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.2.RELEASE)
```

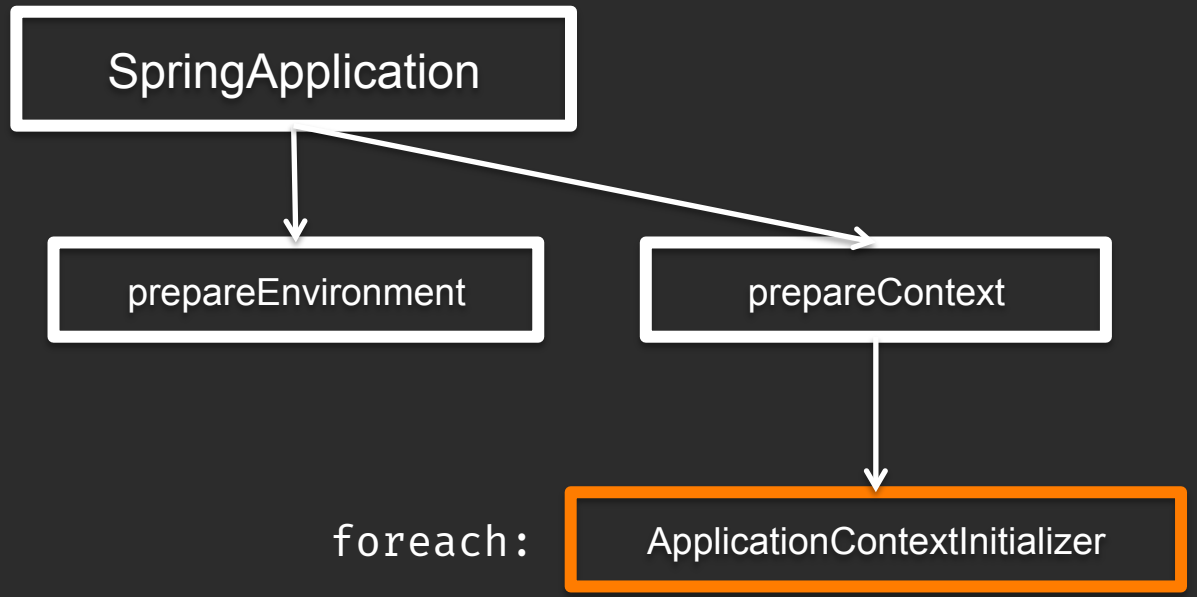
Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

prepareEnvironment

prepareContext

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod



Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

- prepareEnvironment
- prepareContext
- refreshContext

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

- prepareEnvironment
- prepareContext
- refreshContext
  - BeanFactories

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

- prepareEnvironment
- prepareContext
- refreshContext
  - BeanFactories
  - `foreach:` BeanPostProcessors

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

- prepareEnvironment
- prepareContext
- refreshContext
  - BeanFactories
  - `foreach:` BeanPostProcessors
    - ConfigurationPropertiesBindingPostProcessor

AwareContextLoaderDelegate

loadContext

SpringBootContextLoader

- ConfigClasses
- SystemProperties
- FileProperties
- InlinedProperties
- ActiveProfiles

refreshContext

BeanFactories

foreach:

BeanPostProcessors

ConfigurationPropertiesBindingPostProcessor

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

SpringApplication

- prepareEnvironment
- prepareContext
- refreshContext

Create Extensions → BeforeAll → PostProcess TestInstance → **Load Context** → Invoke TestMethod

DependencyInjectionTestExecutionListener

loadContext

DefaultCacheAwareContextLoaderDelegate

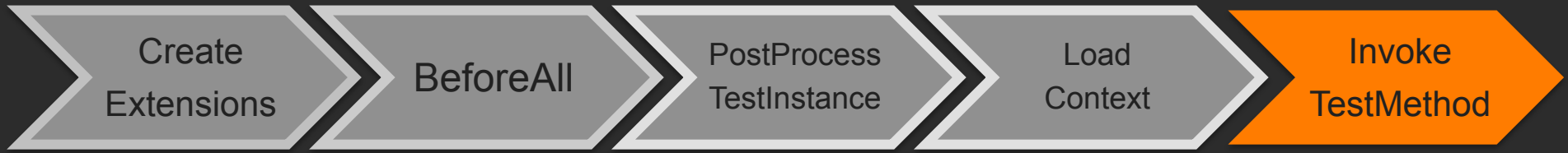
get

loadContext

ContextCache

SpringBootContextLoader

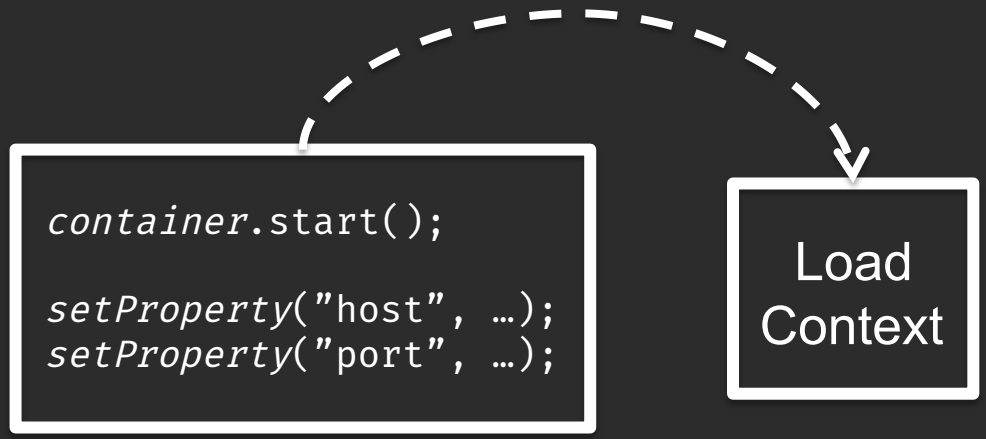
put



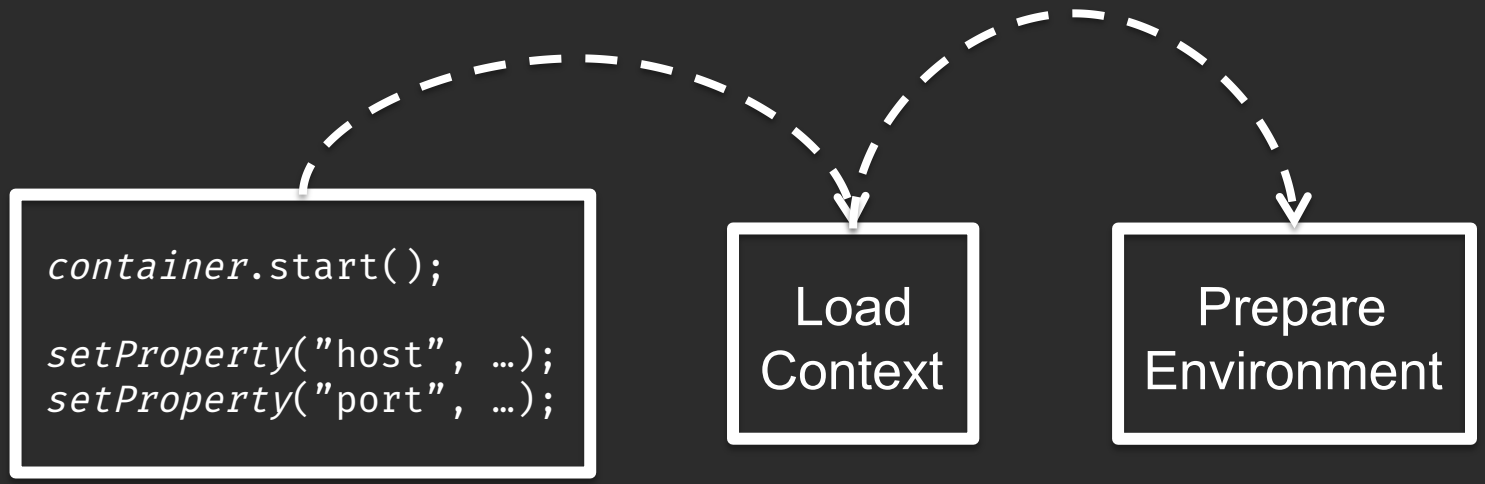

```
@SpringBootTest
@ExtendWith(SpringExtension.class)
@EnableRabbitMqTestContainers
class CTest {

    @Test
    void sendMessageTest() {
        ...
    }

}
```

```
container.start();

setProperty("host", …);
setProperty("port", …);
```

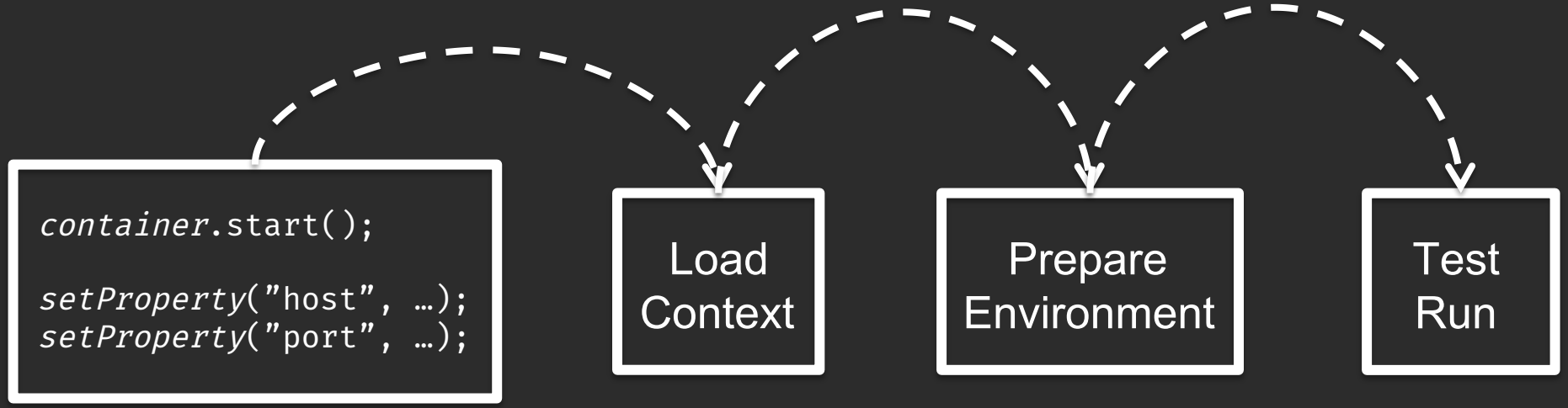
EMPTY CACHE


container.start();
setProperty("host", …);
setProperty("port", …);
Load
Context

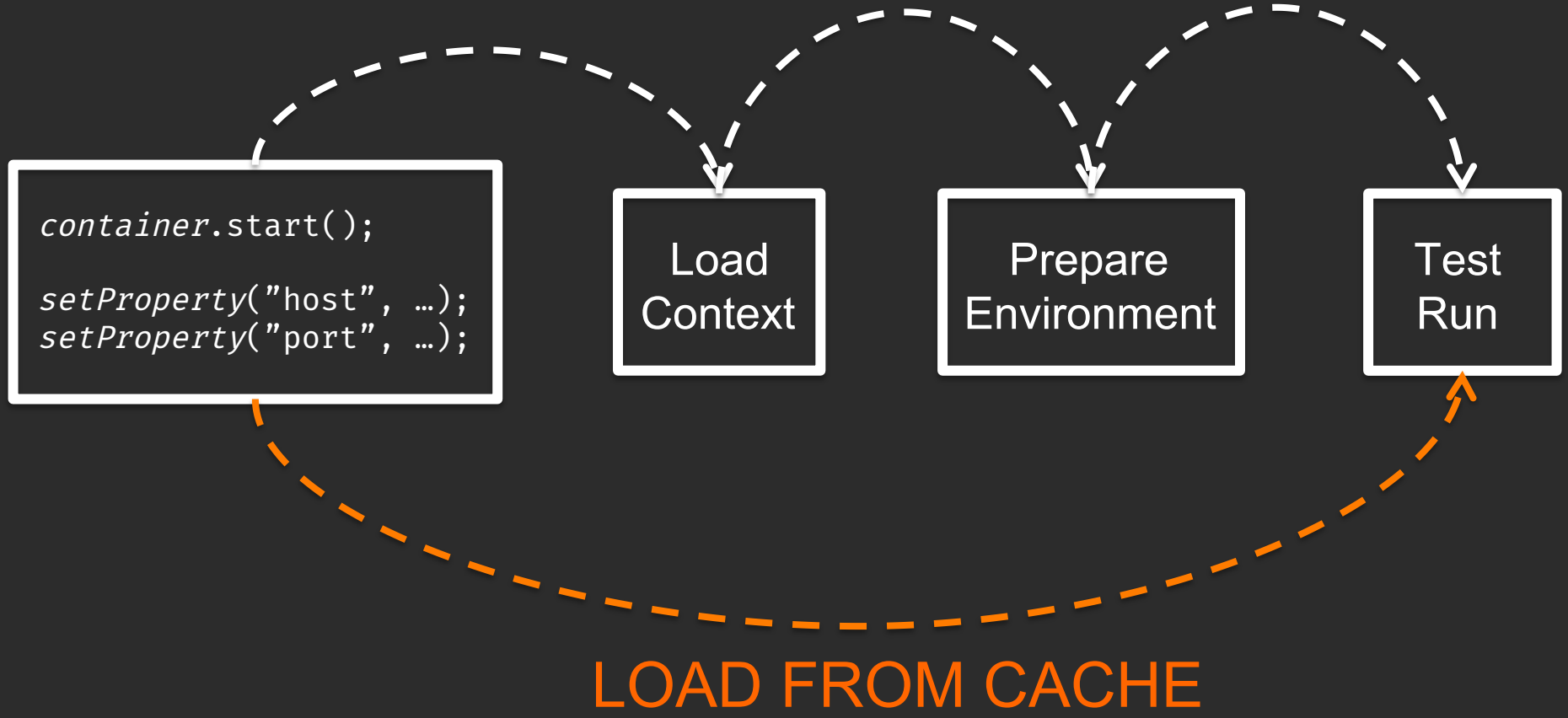
EMPTY CACHE


container.start();
setProperty("host", …);
setProperty("port", …);
Load Context
Prepare Environment

EMPTY CACHE


container.start();
setProperty("host", …);
setProperty("port", …);
Load Context
Prepare Environment
Test Run

EMPTY CACHE


container.start();
setProperty("host", …);
setProperty("port", …);
Load Context
Prepare Environment
Test Run
LOAD FROM CACHE

KEY	CONTEXT
`@SpringBootTest`	SystemTest
`@DataJpaTest`	DataTest
`@WebMvcTest`	MvcTest

KEY	CONTEXT
`@SpringBootTest`	SystemTest
`@DataJpaTest`	DataTest
`@WebMvcTest`	MvcTest
`@ContextConfiguration(classes = TestConfig.class)`	ComponentTest
`@ContextHierarchy`	ComponentTest

KEY	CONTEXT
`@SpringBootTest`	SystemTest
`@DataJpaTest`	DataTest
`@WebMvcTest`	MvcTest
`@ContextConfiguration(classes = TestConfig.class)`	ComponentTest
`@ContextHierarchy`	ComponentTest
`@TestPropertySource(properties = "a=123")`	ComponentTest
`@ActiveProfiles("test")`	ComponentTest

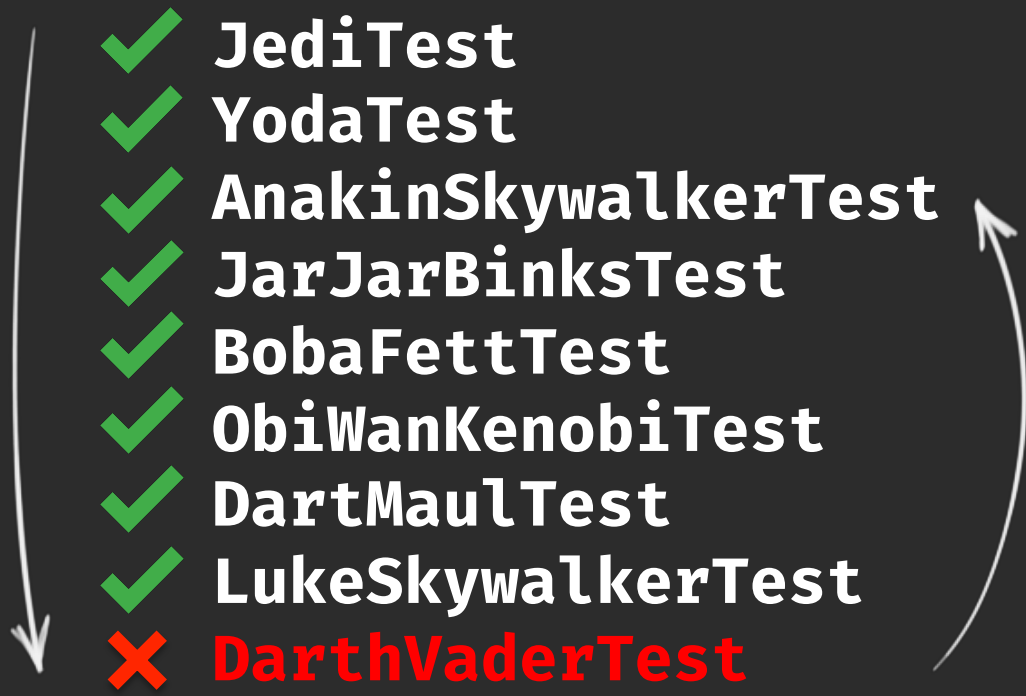
// Что делать?

- ✔ **DarthVaderTest**

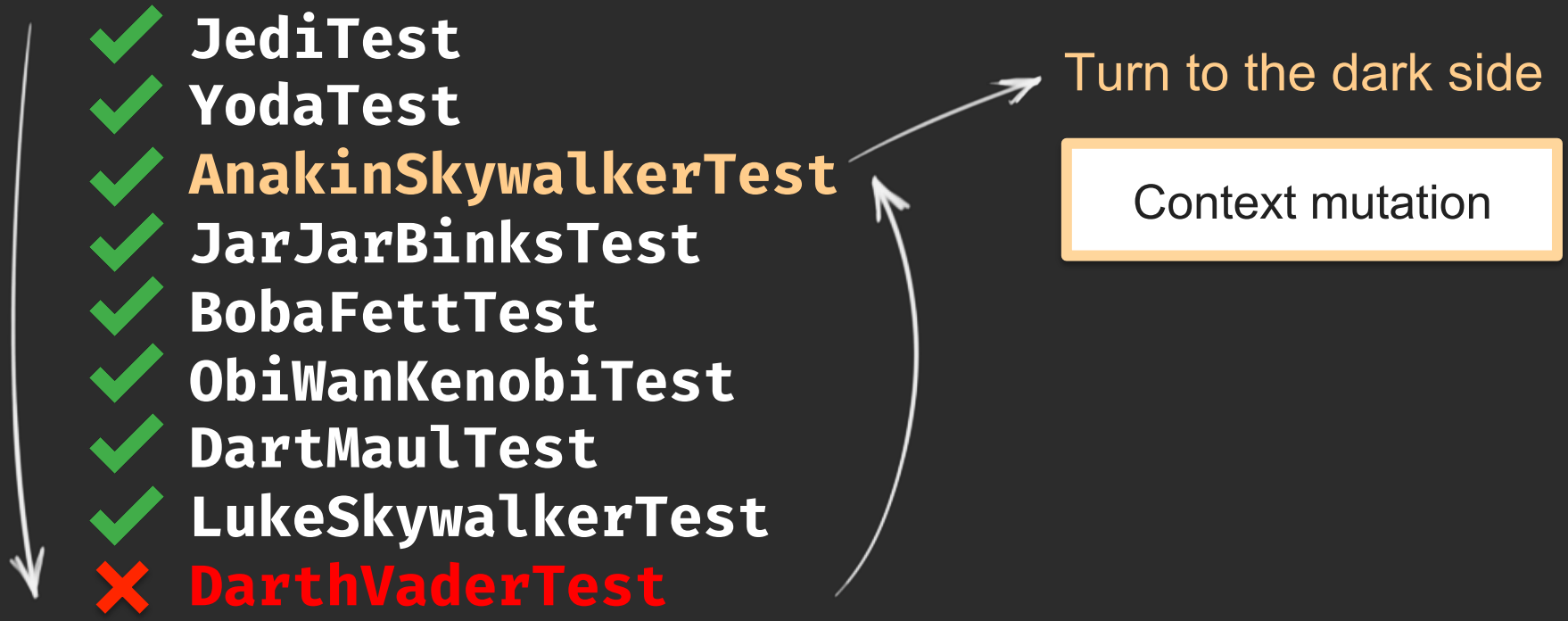
// Что делать?

- ✔ JediTest
- ✔ YodaTest
- ✔ AnakinSkywalkerTest
- ✔ JarJarBinksTest
- ✔ BobaFettTest
- ✔ ObiWanKenobiTest
- ✔ DartMaulTest
- ✔ LukeSkywalkerTest
- ✘ DarthVaderTest

// Что делать?


JediTest
YodaTest
AnakinSkywalkerTest
JarJarBinksTest
BobaFettTest
ObiWanKenobiTest
DartMaulTest
LukeSkywalkerTest
DarthVaderTest

// Что делать?


JediTest
YodaTest
AnakinSkywalkerTest
JarJarBinksTest
BobaFettTest
ObiWanKenobiTest
DartMaulTest
LukeSkywalkerTest
DarthVaderTest
Turn to the dark side
Context mutation

// Может без КЭША обойдемся?

100	интеграционных тестов
10	уникальных контекстов
~12 sec	старт нового контекста
2 min	с кэшем
20 min	без кэша

// Выводы

С JUnit5 можно уменьшить дублирование кода в микросервисных проектах

Мета-аннотации помогают собрать нужную конфигурацию теста, как из конструктора

Вместе с TestContainers можно использовать production-окружение в тестах

Spring Context Caching – сложный механизм, нужно понимать, как он работает

Pull requests & feature requests are welcome!

github.com/jupiter-tools

t.me/test_tools

antkorwin