Architecting for Continuous Delivery with Microservices

Session Three

Testing and Delivering Microservices

Bounded Contexts

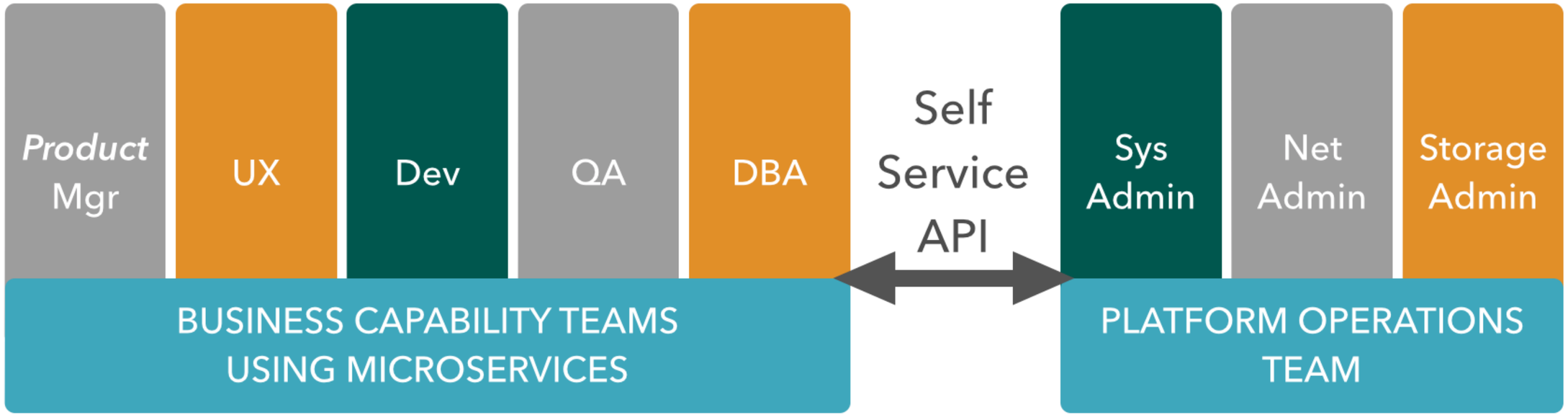
CONWAY'S LAW

Conway's Law

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

— Melvyn Conway, 1967

The Inverse Conway Manuever



Decentralized Autonomous Capability Teams / Services

TESTING

Consumer Driven Contracts

Bounded Contexts

- Crossing Boundaries ➡ Increased Complexity
- Interfaces to Other Contexts
- Availability of Other Services

THIS IS HARD

Consumer to Component =

CONTRACT

Contracts

- Input/Output Data Structure Expectations
- Side Effects
- Performance Characteristics
- Concurrency Characteristics

MULTIPLE CONSUMERS

MULTIPLE CONTRACTS

Contracts

Must Be Met Over 

API =

Contract

Consumer-Driven Contract

I capture my expectations of your service as a suite of automated tests that I provide to you.

You run them in your continuous integration pipeline.

Remember Decentralized Autonomous Capability Teams/Services?

Consumer-Driven Contracts

PACT

https://github.com/realestate-com-au/pact

PACT JVM

https://github.com/DiUS/pact-jvm

Consumer PactFragment

```
@Pact(state = "FortuneState", provider = "FortuneService", consumer = "FortuneUi")
public PactFragment createFragment(ConsumerPactBuilder.PactDslWithProvider.PactDslWithState builder) {
    Map<String, String> headers = new HashMap<>();
    headers.put("Content-Type", "application/json;charset=UTF-8");

    PactDslJsonBody responseBody = new PactDslJsonBody()
            .numberType("id")
            .stringType("text");

    return builder.uponReceiving("a request for a random fortune")
            .path("/random")
            .method("GET")
            .willRespondWith()
            .headers(headers)
            .status(200)
            .body(responseBody).toFragment();
}
```

```
{
  "provider" : {
    "name" : "FortuneService"
  },
  "consumer" : {
    "name" : "FortuneUi"
  },
  "interactions" : [ {
    "providerState" : "FortuneState",
    "description" : "a request for a random fortune",
    "request" : {
      "method" : "GET",
      "path" : "/random"
    },
    "response" : {
      "status" : 200,
      "headers" : {
        "Content-Type" : "application/json;charset=UTF-8"
      },
      "body" : {
        "id" : 8440038105,
        "text" : "JaizxIwBpBfLDObNbYRu"
      },
      "responseMatchingRules" : {
        "$.body.id" : {
          "match" : "type"
        },
        "$.body.text" : {
          "match" : "type"
        }
      }
    }
  } ],
  "metadata" : {
    "pact-specification" : {
      "version" : "2.0.0"
    },
    "pact-jvm" : {
      "version" : "2.1.13"
    }
  }
}
```

Producer Pact Verification

```
<plugin>
    <groupId>au.com.dius</groupId>
    <artifactId>pact-jvm-provider-maven_2.11</artifactId>
    <version>2.1.13</version>
    <configuration>
        <serviceProviders>
            <serviceProvider>
                <name>FortuneService</name>
                <consumers>
                    <consumer>
                        <name>FortuneUi</name>
                        <pactFile>../fortune-teller-ui/target/pacts/FortuneUi-FortuneService.json</pactFile>
                    </consumer>
                </consumers>
            </serviceProvider>
        </serviceProviders>
    </configuration>
</plugin>
```

PACT Broker

https://github.com/bethesque/pact_broker

Pacts

Consumer	Provider
Zoo App	Animal Service

1 pact

A pact between Zoo App and Animal Service

Zoo App version: 1.0.1

Date published: Mon 19 Jan 2015, 5:15pm -05:00

View in HAL Browser

Requests from Zoo App to Animal Service

- A request for an alligator given there is an alligator named Mary
- A request for an alligator given there is not an alligator named Mary
- A request for an alligator given an error occurs retrieving an alligator
- A request for an alligator's sister given there is an alligator named Mary

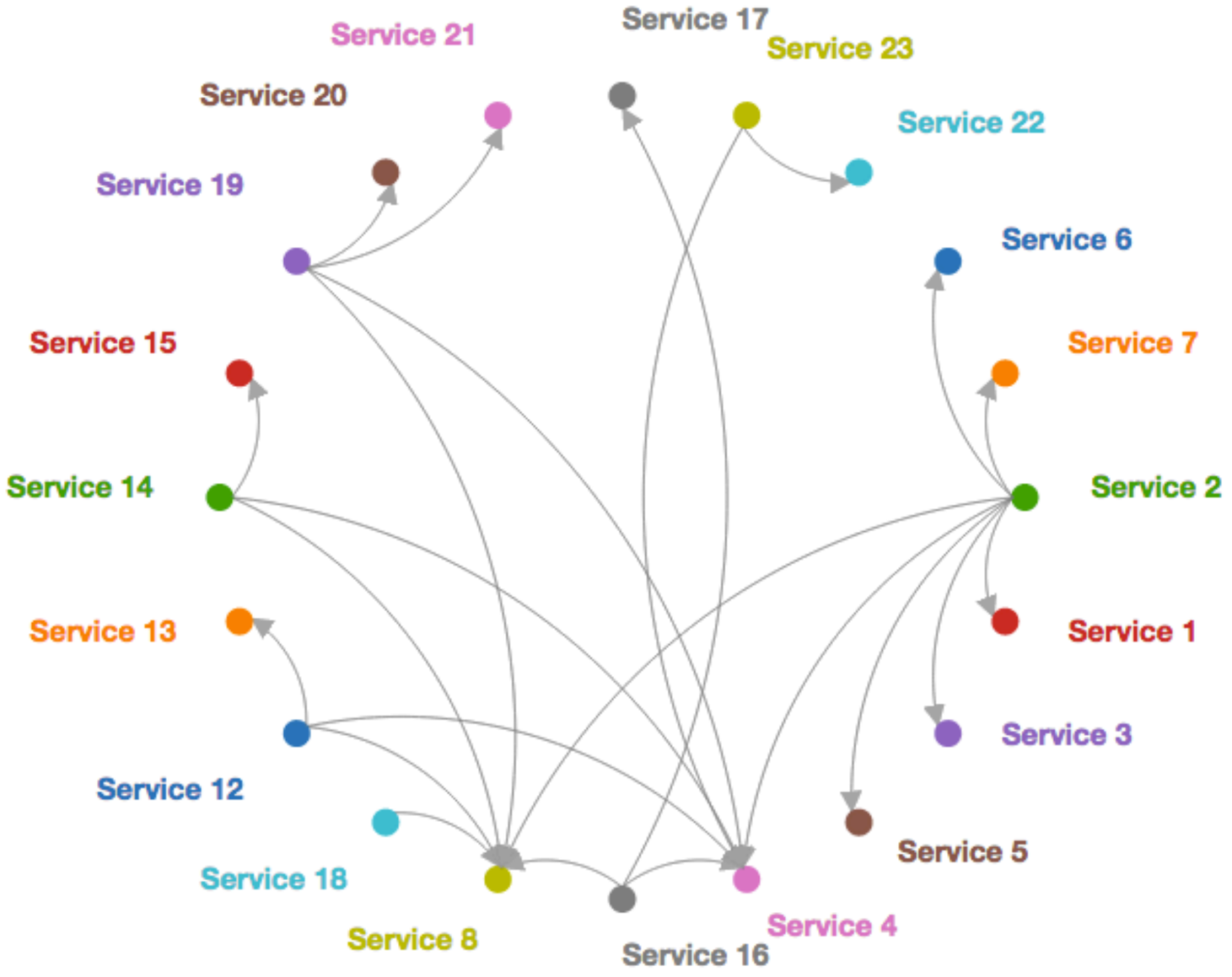
Interactions

Given **there is an alligator named Mary**, upon receiving **a request for an alligator** from Zoo App, with

```
{
  "method": "get",
  "path": "/alligators/Mary",
  "headers": {
    "Accept": "application/json"
  }
}
```

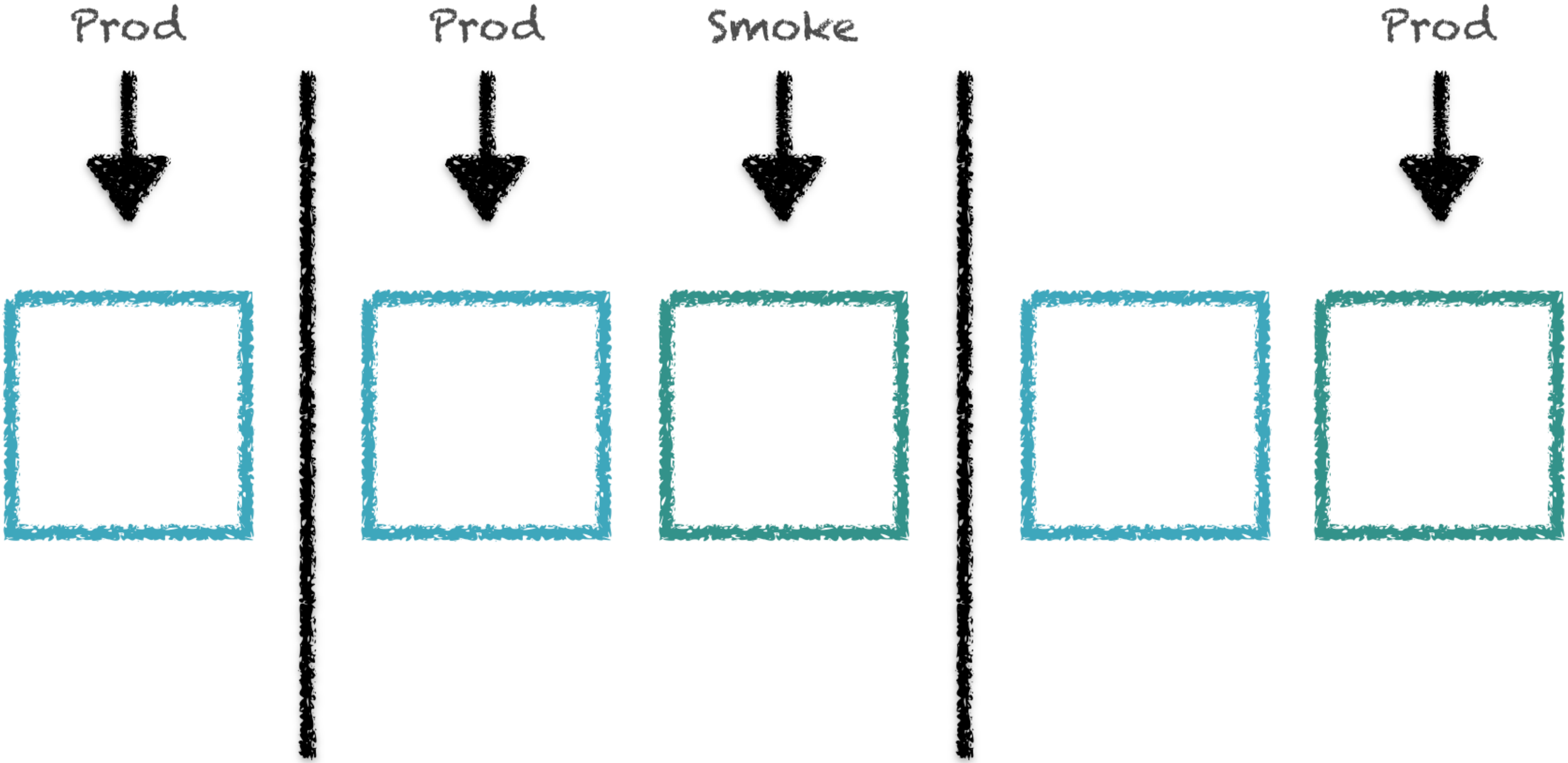
Animal Service will respond with:

```
{
  "status": 200,
  "headers": {
    "Content-Type": "application/json;charset=utf-8"
  },
  "body": {
    "name": "Mary"
  }
}
```

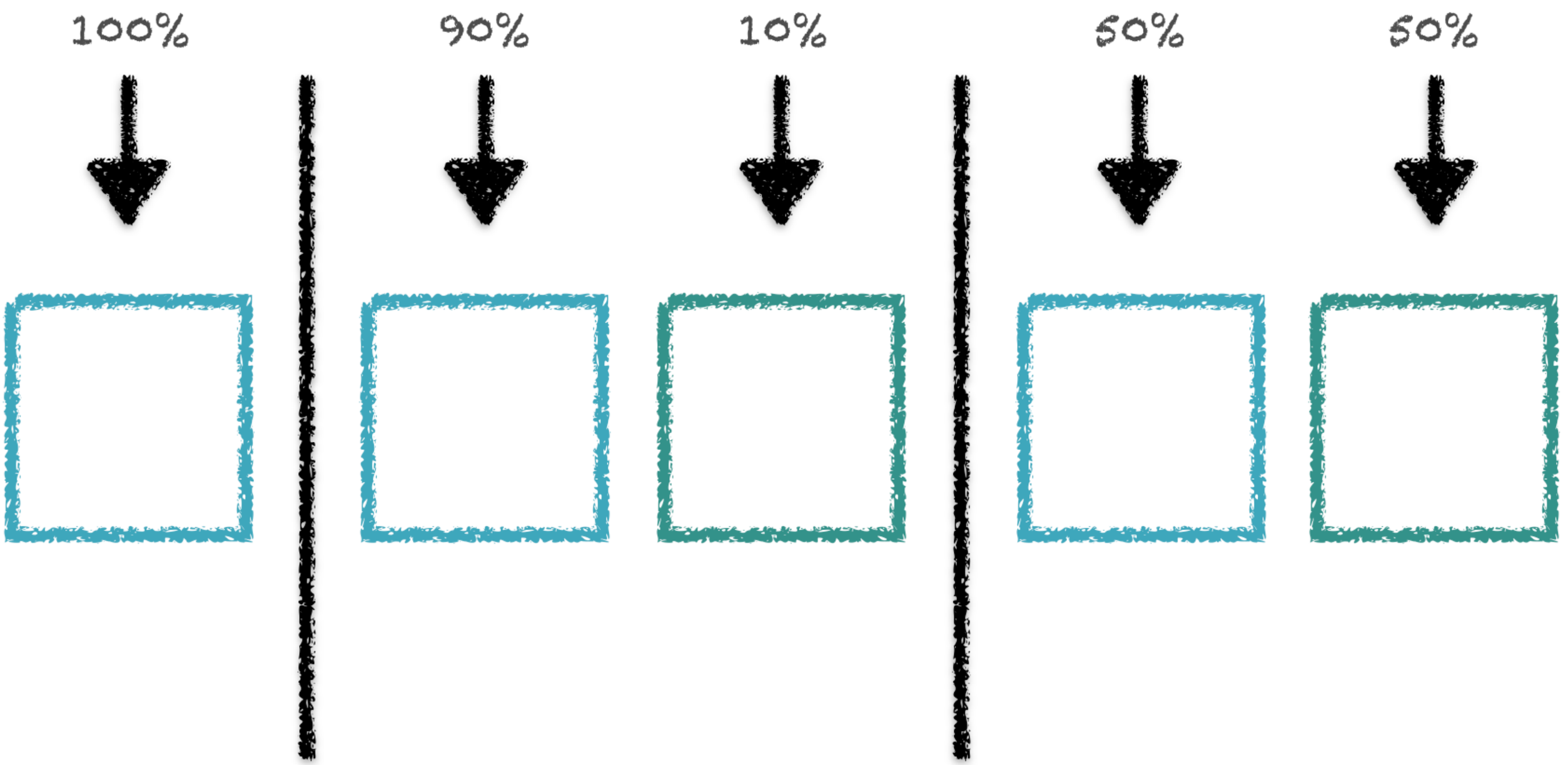

Service 21
Service 17
Service 23
Service 20
Service 22
Service 19
Service 6
Service 15
Service 7
Service 14
Service 2
Service 13
Service 1
Service 3
Service 12
Service 5
Service 18
Service 4
Service 8
Service 16

Testing in Production

Blue Green Deploys


Prod
Prod
Smoke
Prod

Canary Releases


100%
90%
10%
50%
50%

MTBF

vs.

MTTR

Monitor the snot out of production and staging!

Elisabeth Hendrickson

TO THE LABS!