INDEX
SAN FRANCISCO

Discover. Collaborate. Deploy.

Testing Cloud Native from the ground up

Erin Schnabel

@ebullientworks

What is in ...

Cloud Native and microservices architectures are growing in popularity every day, but once you start writing production systems, **how do you make sure they are fully tested?**

This session will give an overview of the trials and tribulations when testing cloud native applications, drawing on our experiences from writing the text-based microservice adventure Game On! and developing the polyglot microservice system that generates code for IBM Cloud users.

Starting from the ground up, it will cover **how to structure your application**, the **different types of tests** you should write and **how to test a system of microservices when working across different teams**.

Finally, it will introduce code examples of how to utilise tools such as JMockit to create mock objects that can be used during testing.

What is out ...

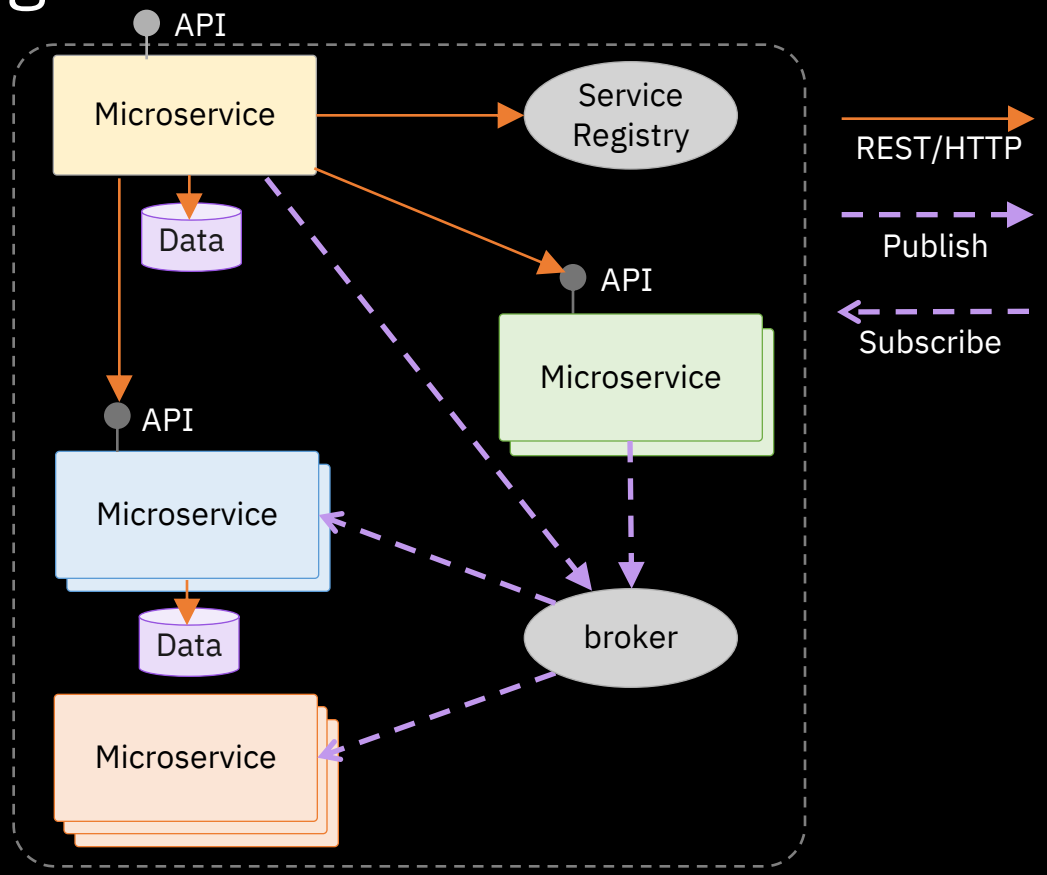
- Performance / Scalability Testing
- Security Testing
- Monitoring with Synthetic Transactions

Cloud Native

An application architecture designed to leverage the strengths and accommodate the challenges of a standardized cloud environment, including concepts such as elastic scaling, immutable deployment, disposable instances, and less predictable infrastructure.

Microservices are used to...

- compose a complex application using
  - “small”
  - independent (autonomous)
  - replaceable
  - processes
- that communicate via
  - language-agnostic APIs



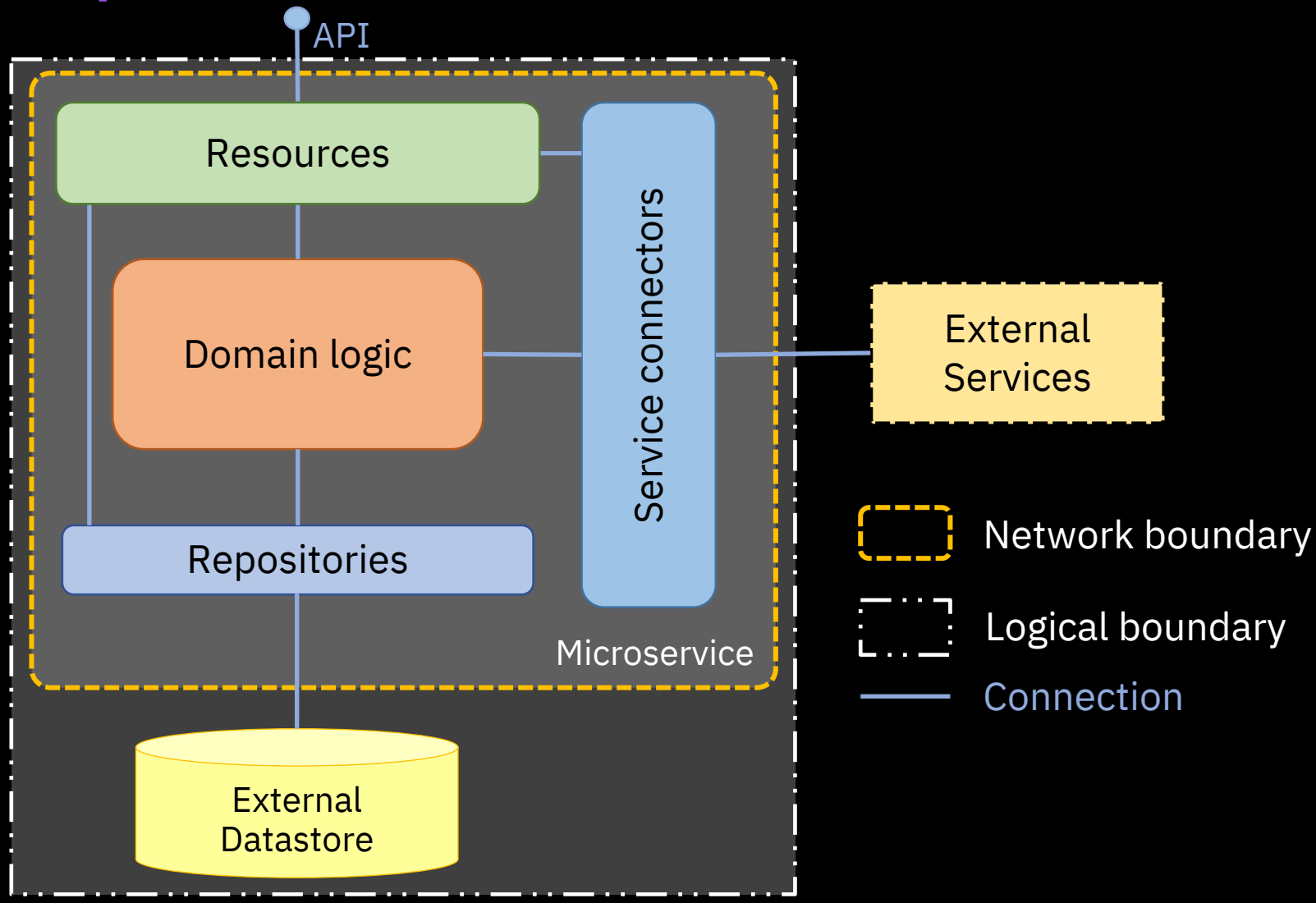
Types of testing

Unit	Single class or module: focus on internal behaviors
Component (Module)	Verify component as a whole. Calls to external services are mocked
Contract (Interface)	Does the service uphold its published contract?
Integration	Verify communication between a components (mocks for others)
End to End	Verify behavior of the entire system as a whole

Build for test! Separation of concerns

- Anti-corruption layer as tool for testing
  - Domain Logic
  - Resources
    - Exposed APIs
  - Service connectors
    - Consumed APIs
  - Repositories
    - Persisted data

CHOOSE ONE



Testing Domain Logic

- Heart of your service
  - No understanding of external dependencies
  - Unit Test #FTW!

```
 */
public Site updateRoom(String authenticatedId, String id, RoomInfo roomInfo) {
    Log.log(Level.FINER, this, "Update site: {0} {1}", id, roomInfo);

    if ( authenticatedId == null ) {
        throw new MapModificationException(Response.Status.FORBIDDEN,
                "Room could not be updated",
                "User was not specified (unauthenticated)");
    }

    Site result = sites.updateRoom(authenticatedId, id, roomInfo);

    //publish event.
    if ( kafka != null )
        kafka.publishSiteEvent(SiteEvent.UPDATE, result);

    return result;
}
```

/*

Testing Resources (similar for Service Connectors or Repositories)

- APIs exposed to other services
  - Might do basic validation per API definition
  - No understanding of domain logic

```java
/**
 * POST /map/v1/sites
 * @throws JsonProcessingException
 */
@POST
@SignedRequest
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response createRoom( RoomInfo newRoom) {

    // NOTE: Thrown exeptions are mapped (see MapModificationException)
    Site mappedRoom = mapRepository.connectRoom(getAuthenticatedId(AuthMode.AUTHENTICATION_REQUIRED), newRoom);

    return Response.created(URI.create("/map/v1/sites/" + mappedRoom.getId())).entity(mappedRoom).build();
}
```

Unit Tests

- Choose a good automated framework
  - JUnit, Karma, ...
- Use Mocks, Stubs, and Fakes
  - Jmockit, Mockito, Angular, ...
- Test for ___ the way the framework wants you to..
  - E.g. for JUnit use an annotation and let the method throw!
    `@Test(expected=MapModificationException.class)`

Mocking frameworks

```java
@Test
public void testException(@Mocked Response response, @Mocked ResponseBuilder builder) {
    exMapper.toResponse(new Exception("TestException")); // 500

    new Verifications() {{
        ResponseBuilder rb;
        Status s;
        String json;

        rb = Response.status(s = withCapture()); times = 1;
        rb.entity(json = withCapture()); times = 1;

        Assert.assertEquals(Response.Status.INTERNAL_SERVER_ERROR, s);
        Assert.assertTrue("Stringified json should include status 500: [" + json + "]",
                json.contains("\"status\":500"));
        Assert.assertTrue("Stringified json should include message containing exception's message: [" + json + "]",
                json.contains("\"message\":\"TestException\""));
        Assert.assertFalse("Stringified json should not include info",
                json.contains("more_info"));
    }};
}
```

```java
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.ResponseBuilder;
import javax.ws.rs.core.Response.Status;
```

Unit Tests

- Erin’s tip: Don’t go for 100% coverage
- Write tests to cover key behaviors
- Balance the value of a test against maintenance cost
  - But watch for inheritance creep – DRY can trip you
  - JMockit: `Deencapsulation.invoke`
- Be kind to future you!

Component Tests

- *Test doubles* fill the role of external dependencies
  - Maintaining behavior of the test double becomes the problem

- Frameworks for Test doubles
  - Spring Cloud Contract + WireMock
    - Contract Definition Language (DSL)

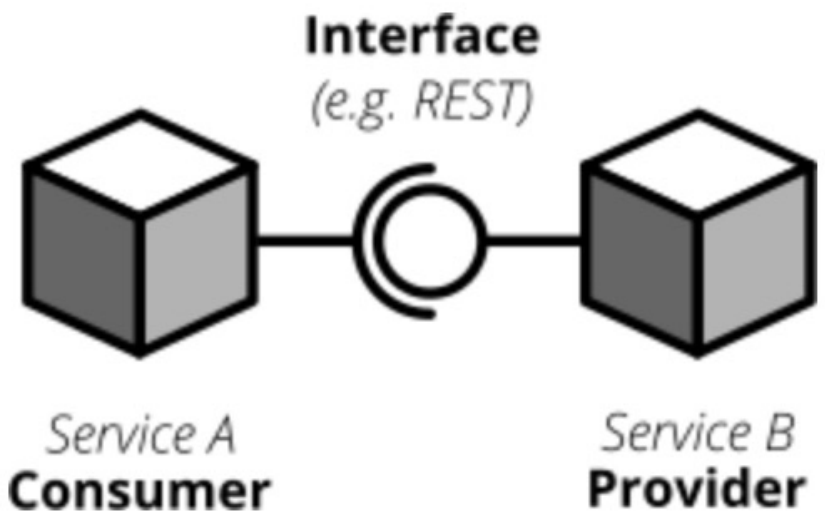
  - Smart Bear: Auto API Mocking
    - Swagger/Open API based!

Where does test data come from?

- For some test doubles, right from Open API documentation!

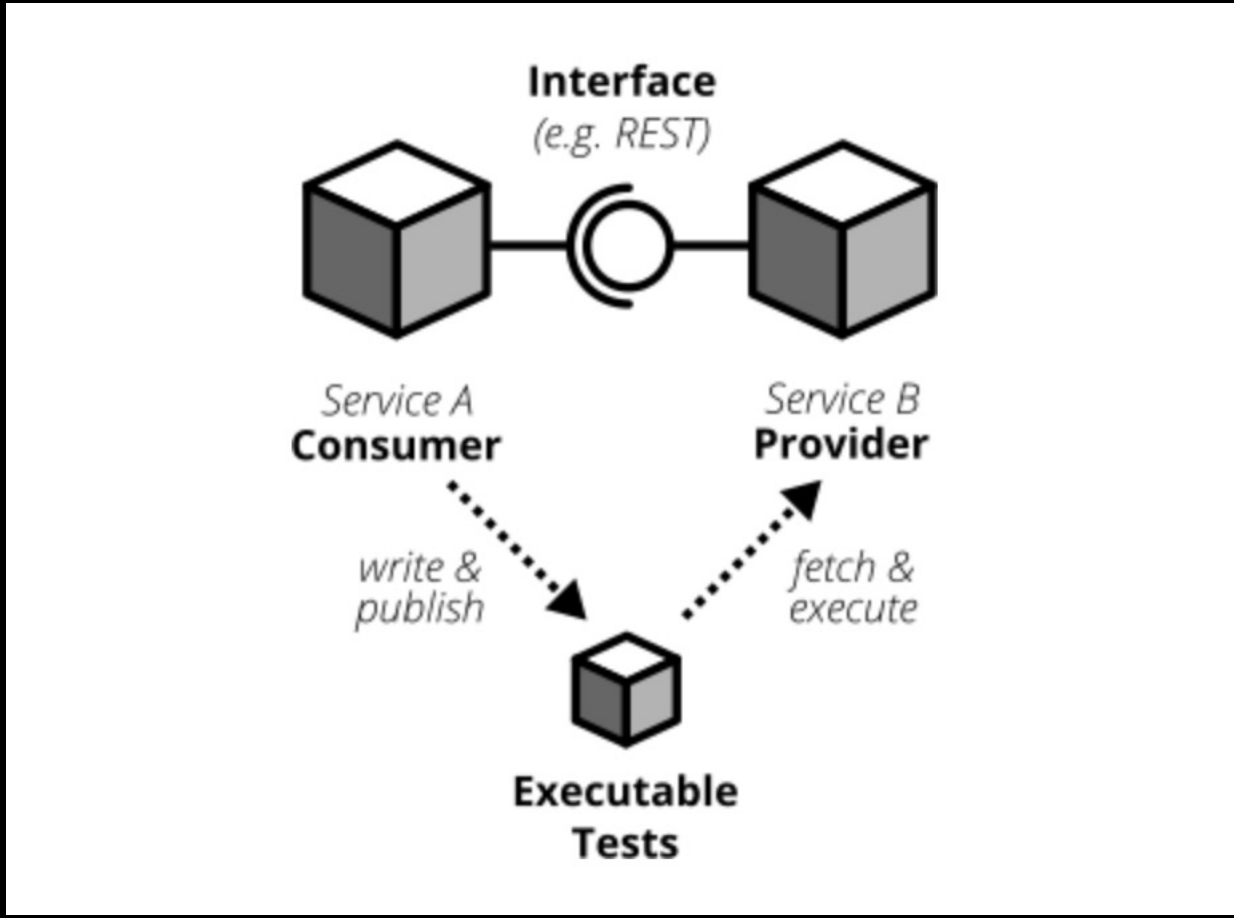
- Pre-test / Post-test data population and cleanup
  - Some NoSQL databases can be populated w/ curl + json

Contracts



Each interface has a providing (or publishing) and a consuming (or subscribing) party. The specification of an interface can be considered a contract.

Consumer Driven Contract tests



Contract Tests – Consumer Driven

- Contract testing is not functional testing
  - Consumer and Provider have a shared understanding of API
  - HOW not WHY

- Pact – Consumer Driven Contracts, https://docs.pact.io/
  - Drives API from “consumers” POV

- Swagger + Pact
  - https://bitbucket.org/atlassian/swagger-mock-validator (node)
  - https://bitbucket.org/atlassian/swagger-request-validator (java)

Contract Tests -- Provider

- Only write what is required by the contract! (YAGNI)
- Run tests *provided by the consumer*

- Compare published swagger with generated swagger
  - https://github.com/RobWin/assertj-swagger

- Generate client stub from Swagger definition
  - Use that client to run tests against provider

Integration Tests

- A **service** and near-neighbors:
  - Data store, cache, or other external dependency
- Goal: Verify **service** can communicate with other services
  - Not trying to verify the other service
  - Test where you *serialize* or *deserialize* data
- Try to target a real instance
- Don’t target a production server

End to end testing?

Medium

Cindy Sridharan Follow
@copyconstruct on Twitter. views expressed on this blog are solely mine and NOT those of my employer.
Dec 30, 2017 · 39 min read

Testing Microservices, the sane way

There's no dearth of information or best-practices or books about how best to test software. This post, however, focuses solely on testing backend *services* and not desktop software or safety critical systems or GUI tools or frontend applications and what have you.

Cindy Sridharan
@copyconstruct

Yep! The whole point of microservices is to enable teams to develop, deploy and scale independently.

Yet when it comes to testing, we insist on testing *everything* together by spinning up *identical* environments, contradicting the mainspring of why we even do microservices.
twitter.com/thramp/status/…

4:16 PM - Dec 14, 2017

112 61 people are talking about this

”Proposal: rename ‘staging’ to ’theory’: “It works in theory, but not on production” – Najaf Ali

No substitute for experimentation in real production environment

Containers *will not fix* your broken culture

(and other hard truths)

Automated, Manual, Chaos, Canary

- Automated tests in CI/CD
- Manual, intentional or exploratory tests catch a lot
- Chaos monkey: well-timed failures in production
- Canary testing: gradual, selective rollout of new features
- A/B testing: concurrent versions, compare behavior

References

- IBM Redbooks: Microservices Best Practices for Java
  http://www.redbooks.ibm.com/abstracts/sg248357.html?Open
- Toby Clemson(ThoughtWorks): Testing Strategies in a Microservice Architecture
  https://martinfowler.com/articles/microservice-testing/#anatomy-modules
- Testing Microservices, the sane way – Cindy Sridharan – Medium
- https://dzone.com/articles/keep-your-promises-contract-based-testing-for-jax
- Bridget Kromhout: Containers will not fix your broken culture
  https://queue.acm.org/detail.cfm?id=3185224
- Ham Vocke (ThoughtWorks): Testing Microservices
  http://www.hamvocke.com/blog/testing-microservices/

Notices and disclaimers

Notices and disclaimers continued

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**
- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.