Leveraging Istio for Creating API Tests

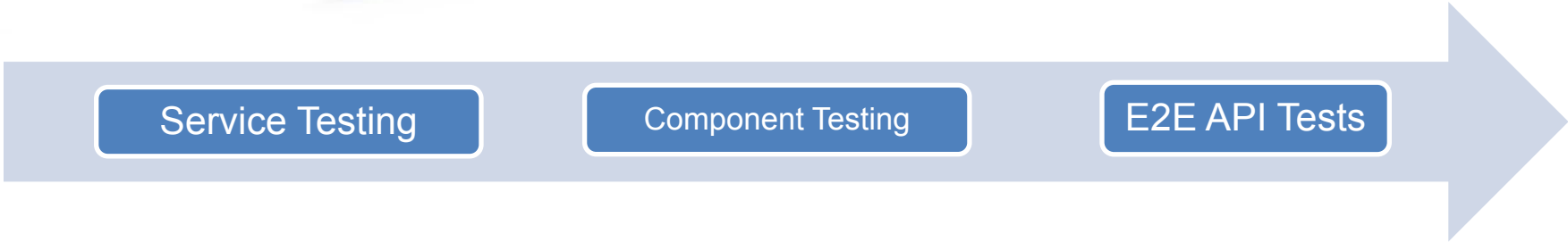
Low Effort API Testing for Microservices

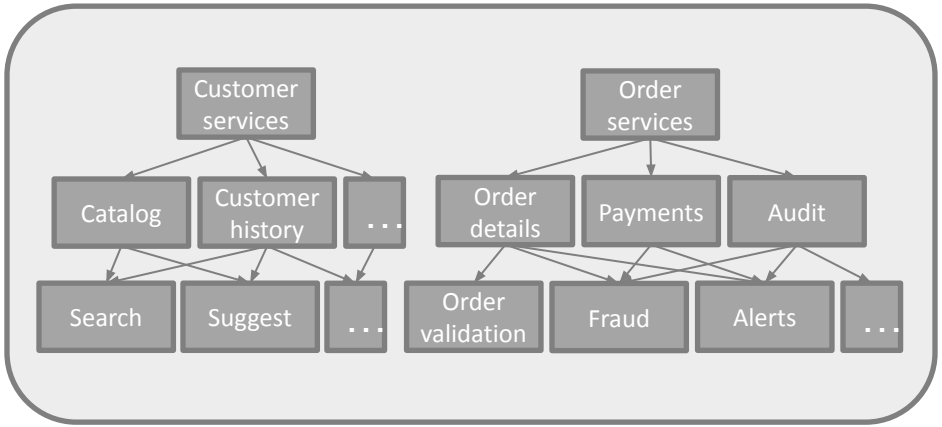
mesh dynamics

Structure

- What has changed?
  - Migration to microservices triggering need for extensive API tests
- Problem:
  - Creating API tests is effort intensive
  - Creating + maintainting E2E, service tests, component tests adds up very quickly
- What happens if you do not address the problem?
  - Thorough test coverage can take a lot of time and effort
  - Realistic outcome: Just create E2E tests
- What is our solution?
  - Leverage Istio sidecar to listen to API traffic data and create tests from the data
  - 10x speed in creating API tests
    - Can also be sped up by just navigating the application UI
  - Create E2E tests, component tests and service tests from the same data
- Key product benefits (#releases, #rollbacks, MTTR, #bugs-in-production, Reduced eng effort for testing, velocity)
  - Early testing of services components auto-generated from end-to-end tests
  - Significantly reduced time and cost for API testing for microservices architectures with Istio
  - Fewer failures higher up the test pyramid as a result of improved API tests
- Istio benefits
  - Venky / Prasad – point here
- Demo
- Questions

API-driven applications exploding



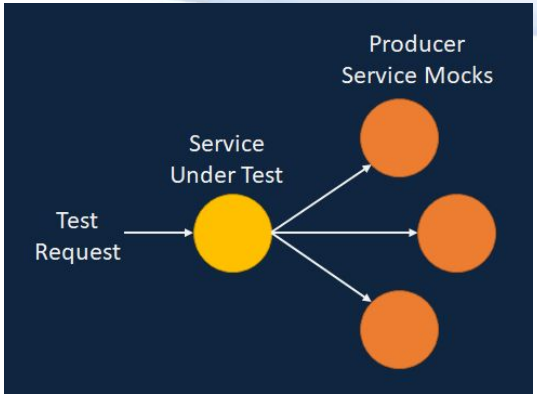


Engineering effort grows superlinearly as #APIs grow

Terminology

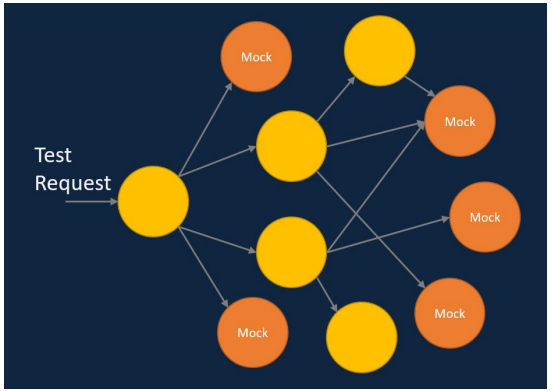
Service testing

Test a single service in isolation. All producer services are mocked.

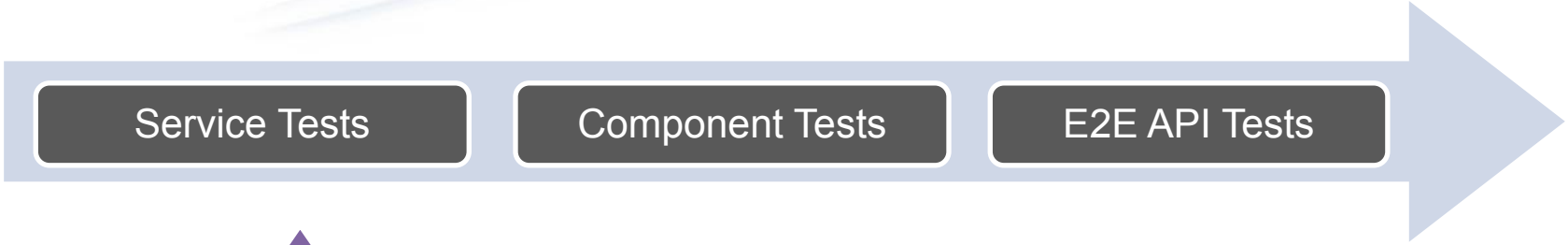


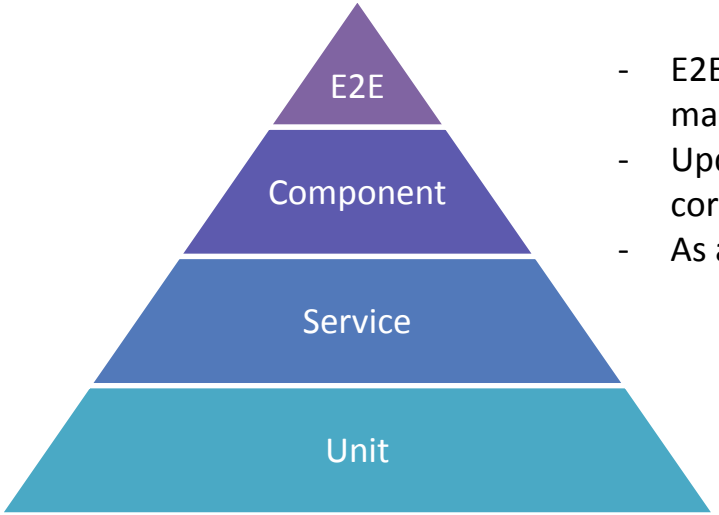
Component testing

Test a set of services as a single sub-system while isolating them from other services, for example payment processing system



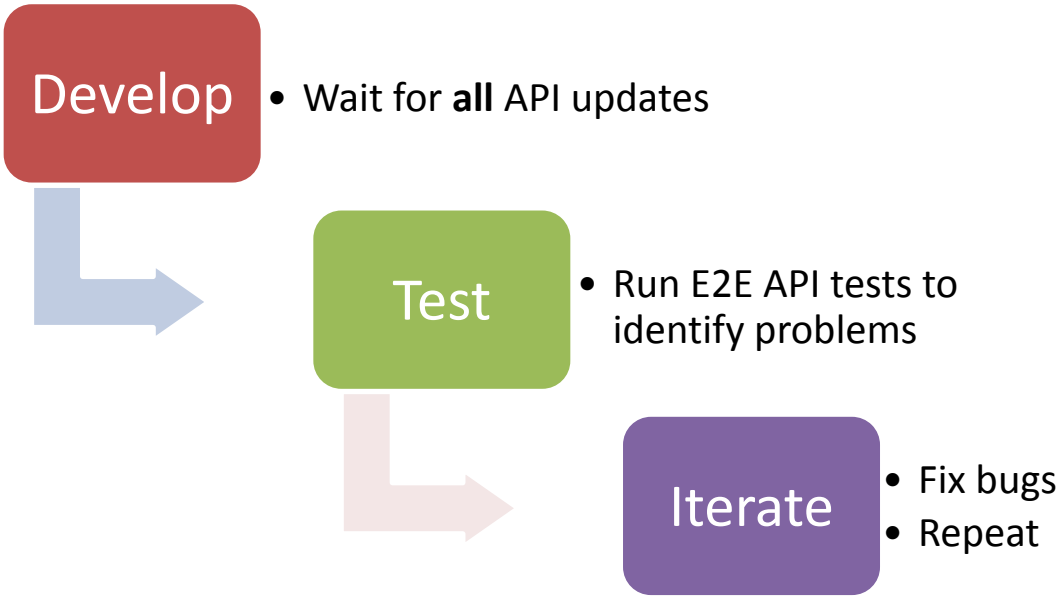
Current approaches do not scale with #APIs





- E2E, component and service tests created manually are often created independently
- Updates to an API require updating corresponding Service and Component tests
- As a result, teams would go for just E2E tests

Teams often focus on End-to-End tests (besides unit tests)



Testing starts late in the API development process.

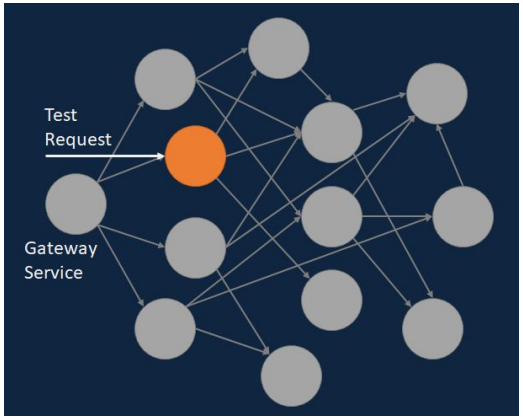
That's not good!!

What we need…

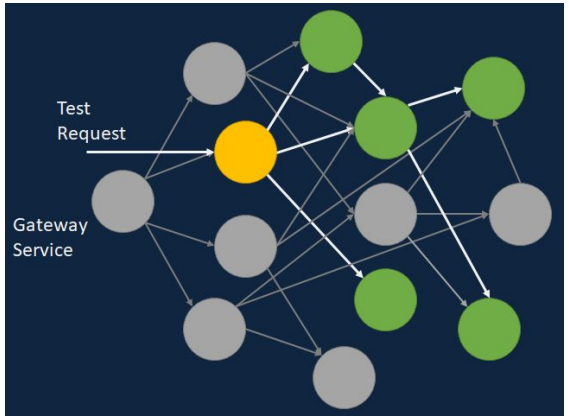
Start testing earlier

Create and maintain a balanced test pyramid

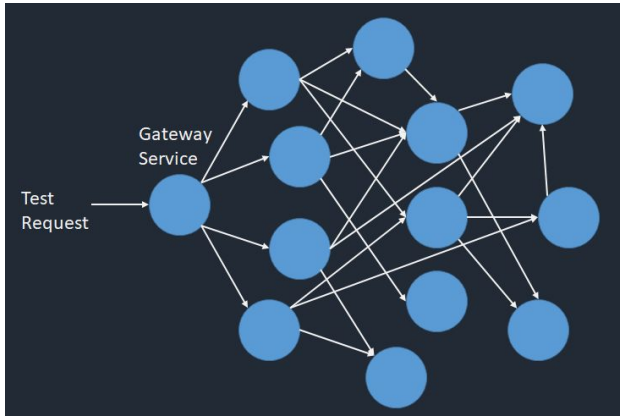
Create different types of tests with low effort



Service

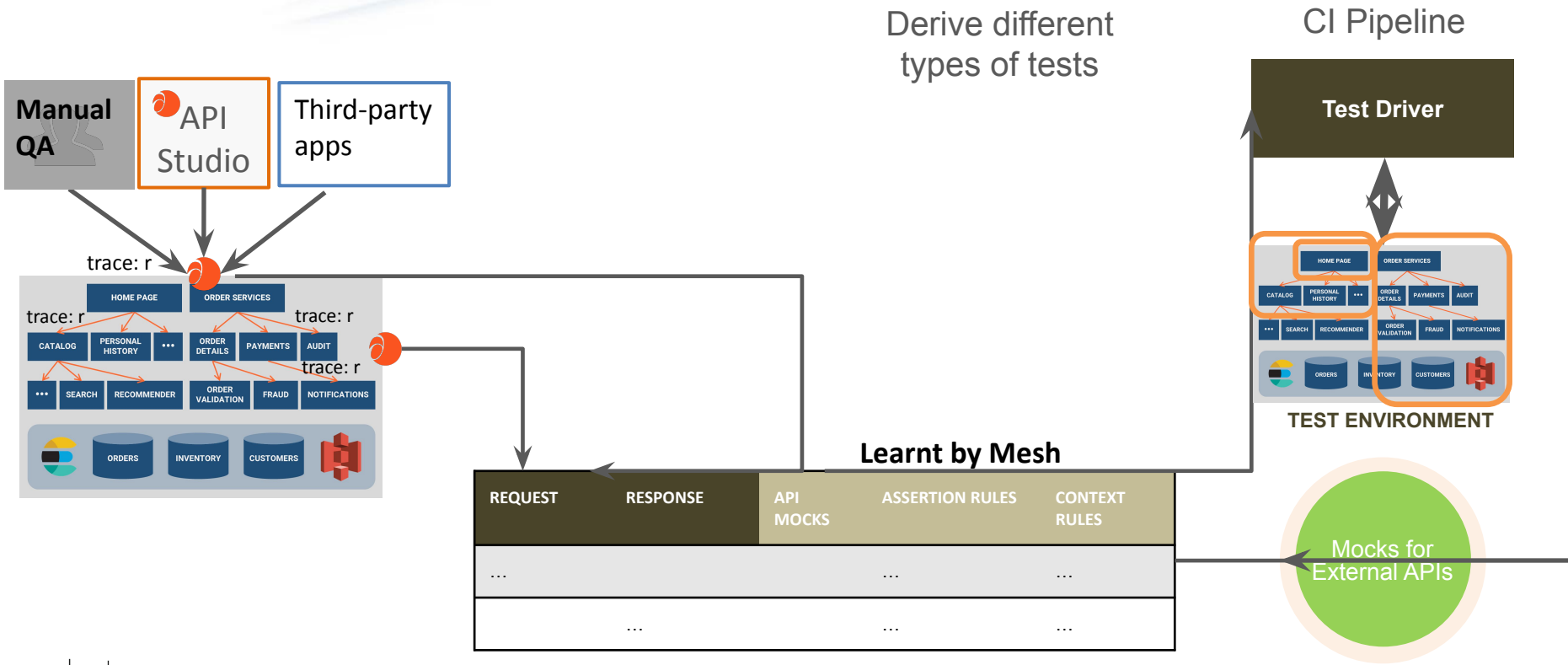


Component

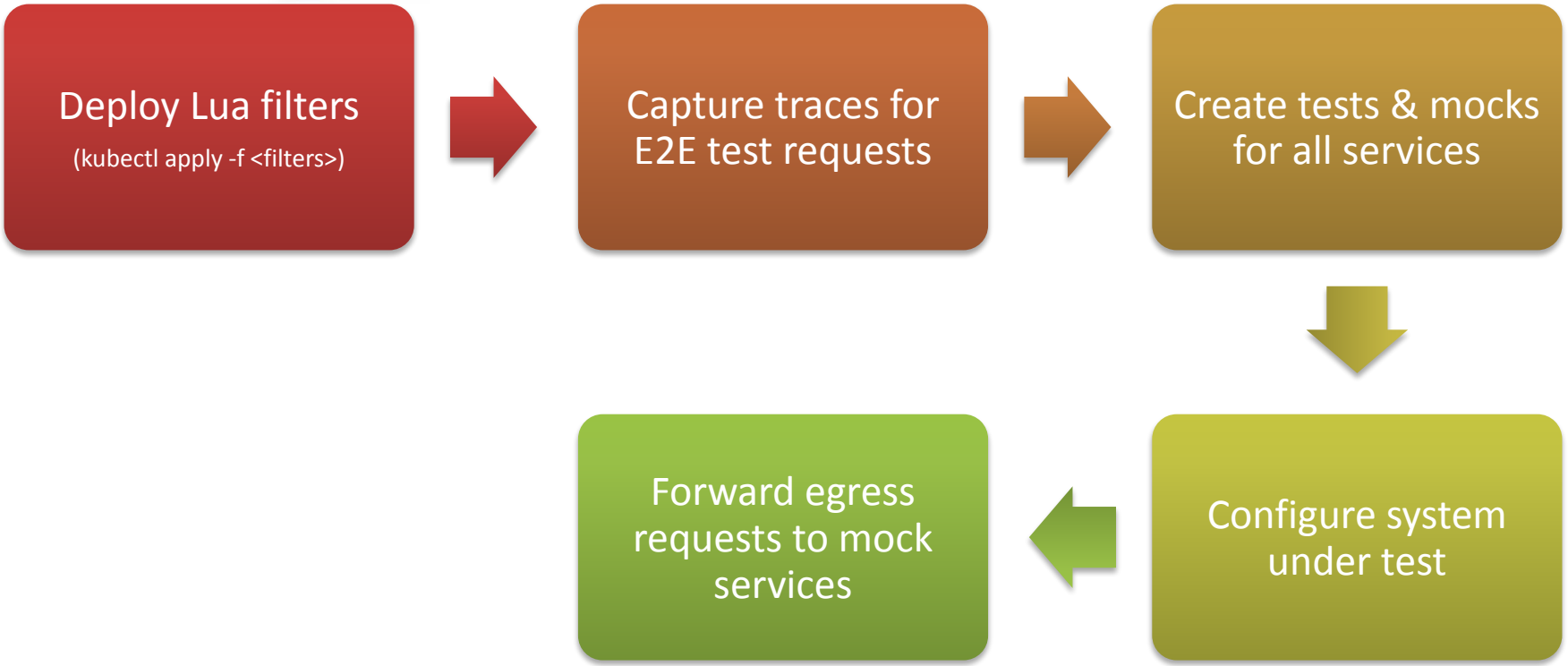


End-to-end

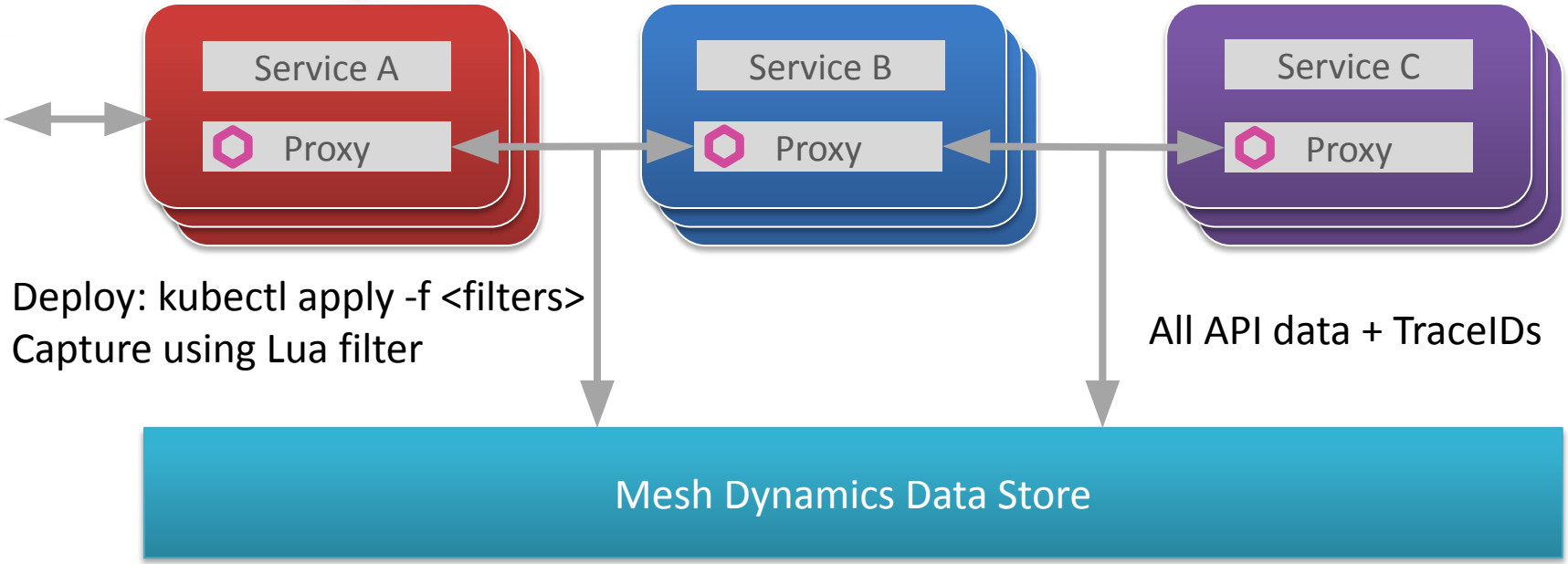
Istio enables learning tests from API usage



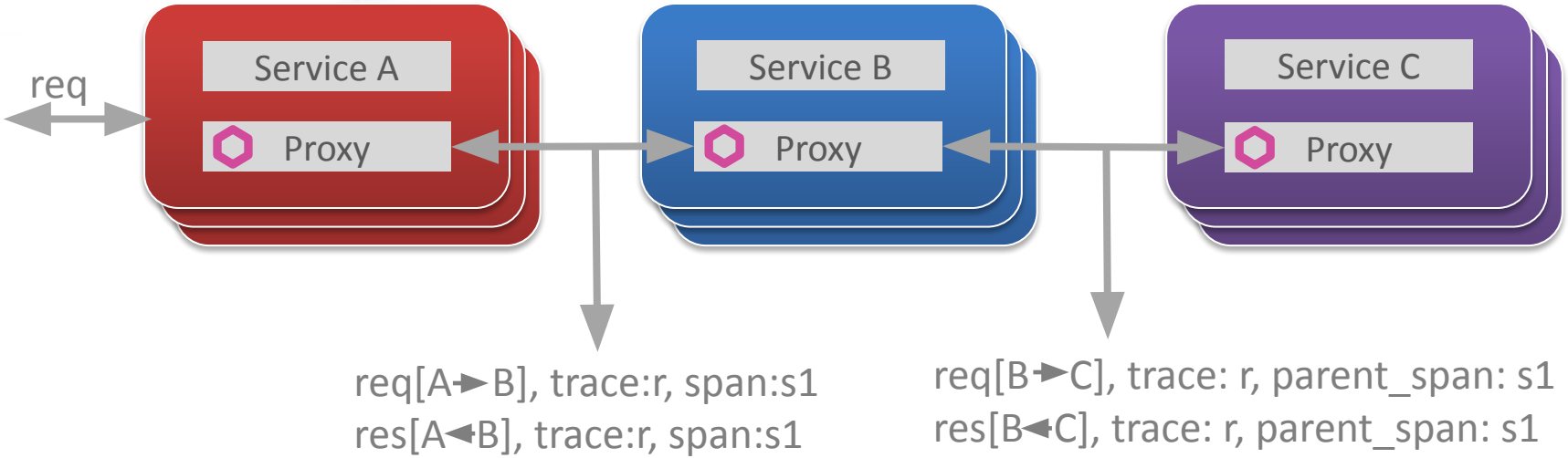
Process flow using Istio



Capture API interactions with lua filters



Assemble API request traces

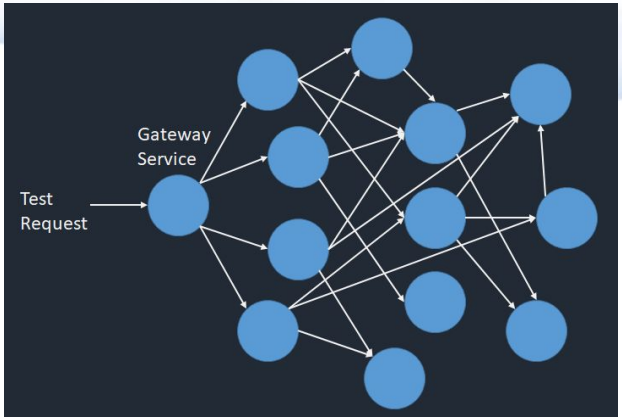


Construct request trace

req
→ req[A->B]
→ req[B->C]

At this point, we have:

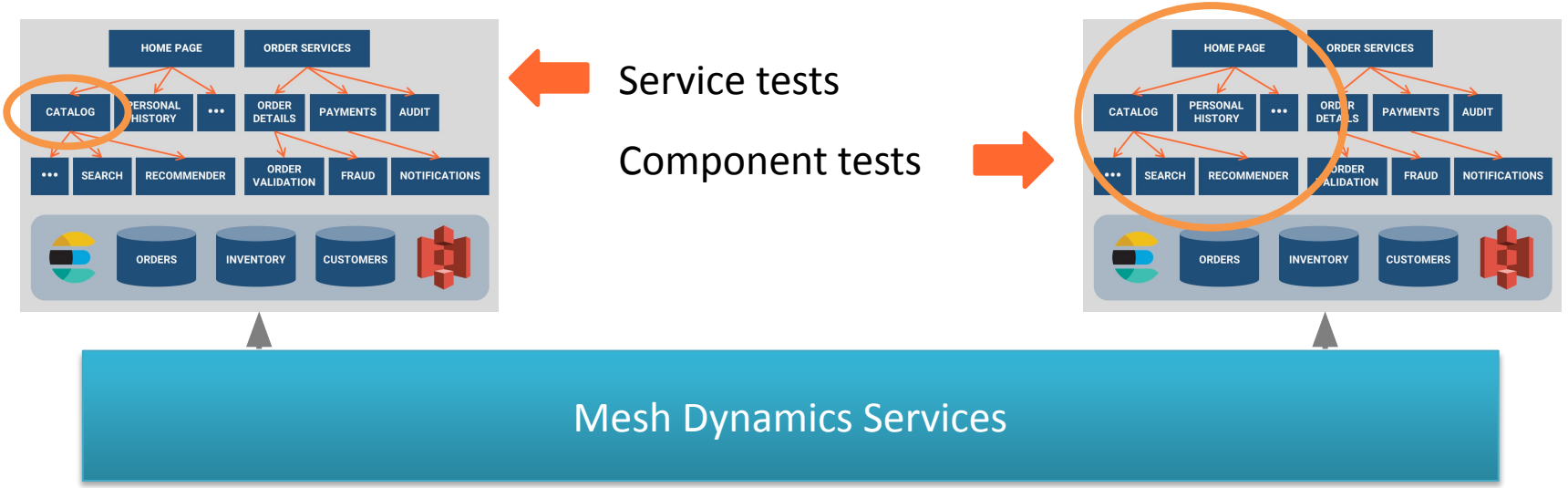
- Full trace of every request from the gateway
- Complete request and response data for every API request in a trace



From this data, we can:

- Drive test requests to any of the endpoints
- Create precise mocks for any of the endpoints

Leverage API data for different types of tests

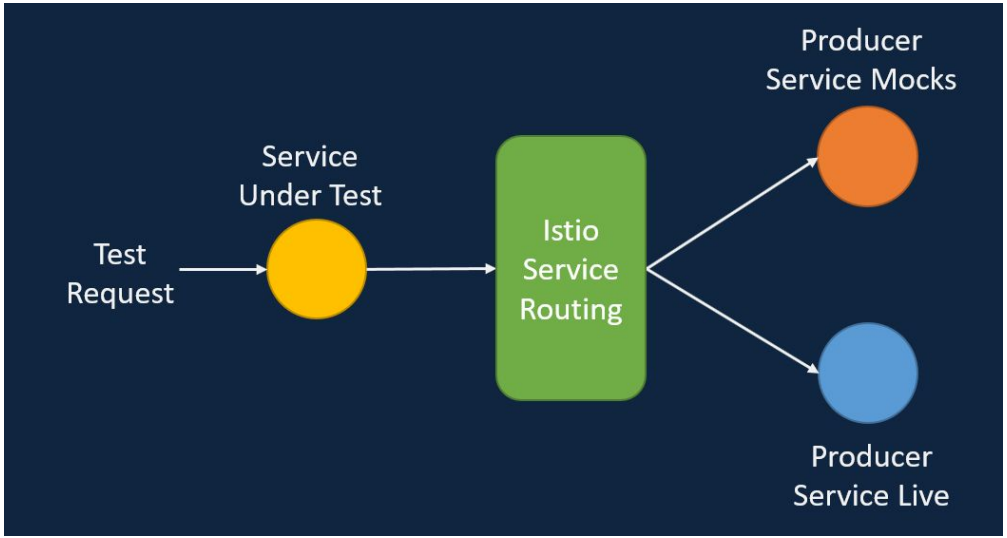


Configure mocks with Istio virtual service

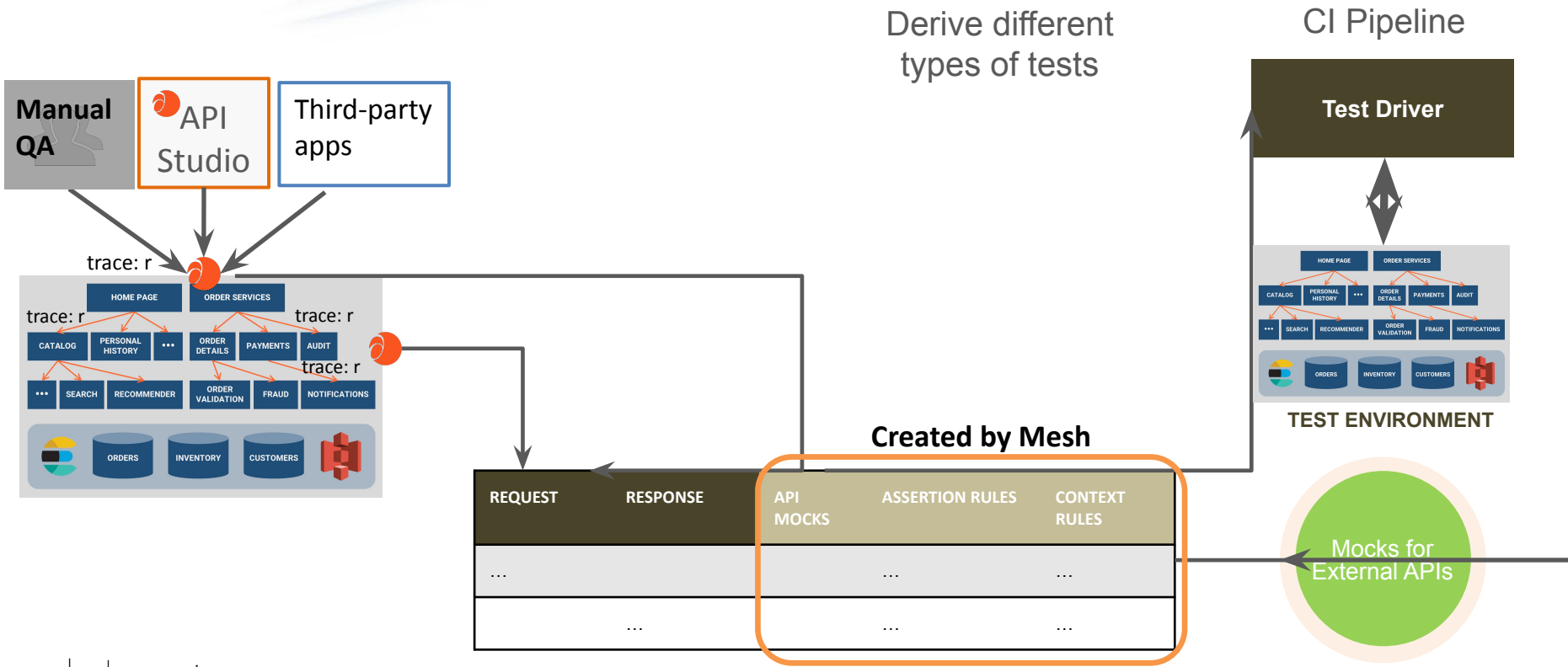
Route requests to mock svc with a virtual service

```
- match:
    - uri:
        prefix: /reviews
    rewrite:
      uri:
/api/ms/CubeCorp/MovieInfo/test/reviews/reviews
```

On-demand configuration to test any component/service



Creating test suites from API traffic



ML-assisted Context Rule Learning

Test execution sequence

Problem

- Test uses outcome of a previous API request
- Context propagation rarely obvious

Challenge

- Dependencies require lot of time to code
- Many dependencies in a test suite
- Dependency maintenance is effort intensive

Solution

- ML-driven identification of candidate relationships
- Supervised system to accept true positives
- No code!

ML-assisted Assertion Rule Learning

Reference data

createOrder Response: Recording

```
{
    "orderId": "ORDR1890675",
    "orderValue": "58.75"
}
```

Test data

createOrder Response: Test

```
{
    "orderId": "ORDR1892533",
    "orderValue": "28.00"
}
```

Problem

- Not all differences are errors

Challenge

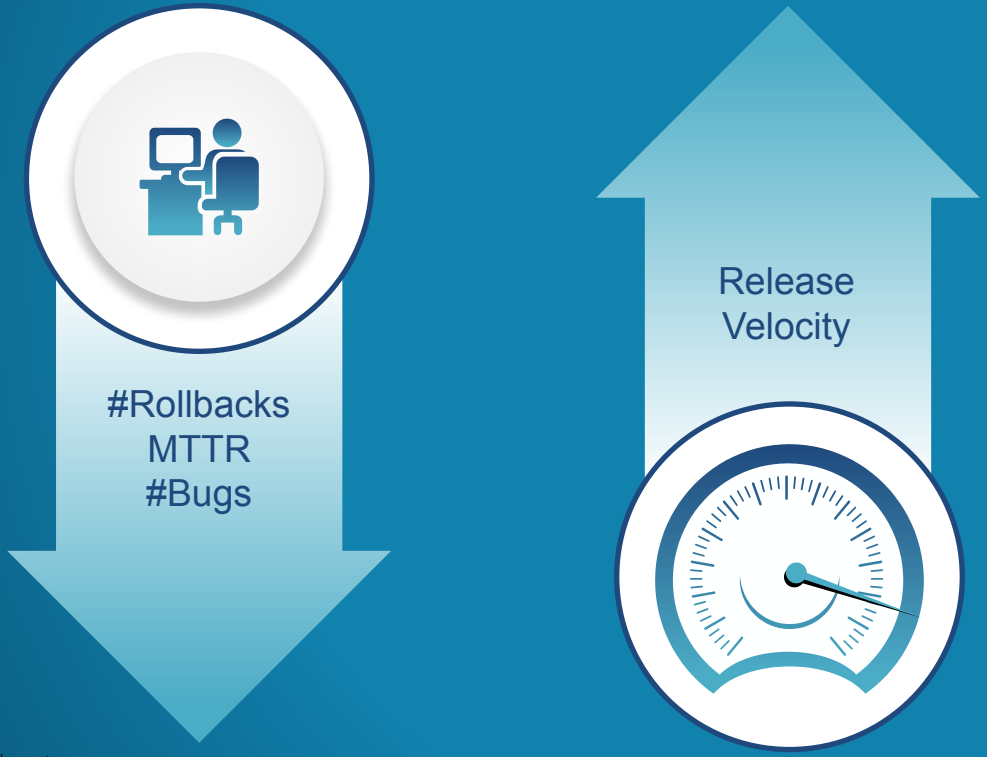
- Assertion creation/maintenance is effort intensive

Solution

- Comprehensive comparison of results
- ML-driven identification of decision rules
- Human review to accept the learned rules
- No code!

Summary: create different types of tests efficiently by learning from API traffic

Scale API Functional & Integration Testing



Improve productivity of each of your developers

10x API test and mock creation speed

DEMO

Download MeshD and use

http://www.meshdynamics.io