ā'pěks alpe adria

A regional Oracle APEX conference for everyone

# Put your ORDS API to the Acid Test

**APEX Alpe Adria**
**April 17th, 2026, 14:40 - 15:25 | Room B**

**Robert Marz**
DATABEE
Die IT-Architekten

## Primary Role

Senior Technical Architect
with database centric view of the world

Oracle ACE

# Tech Superstars Unite

Get worldwide recognition as an Oracle ACE

Oracle.com profile page


Swag, certification exam credit & event passes

Exclusive content

Networking events

Your own Oracle cloud account

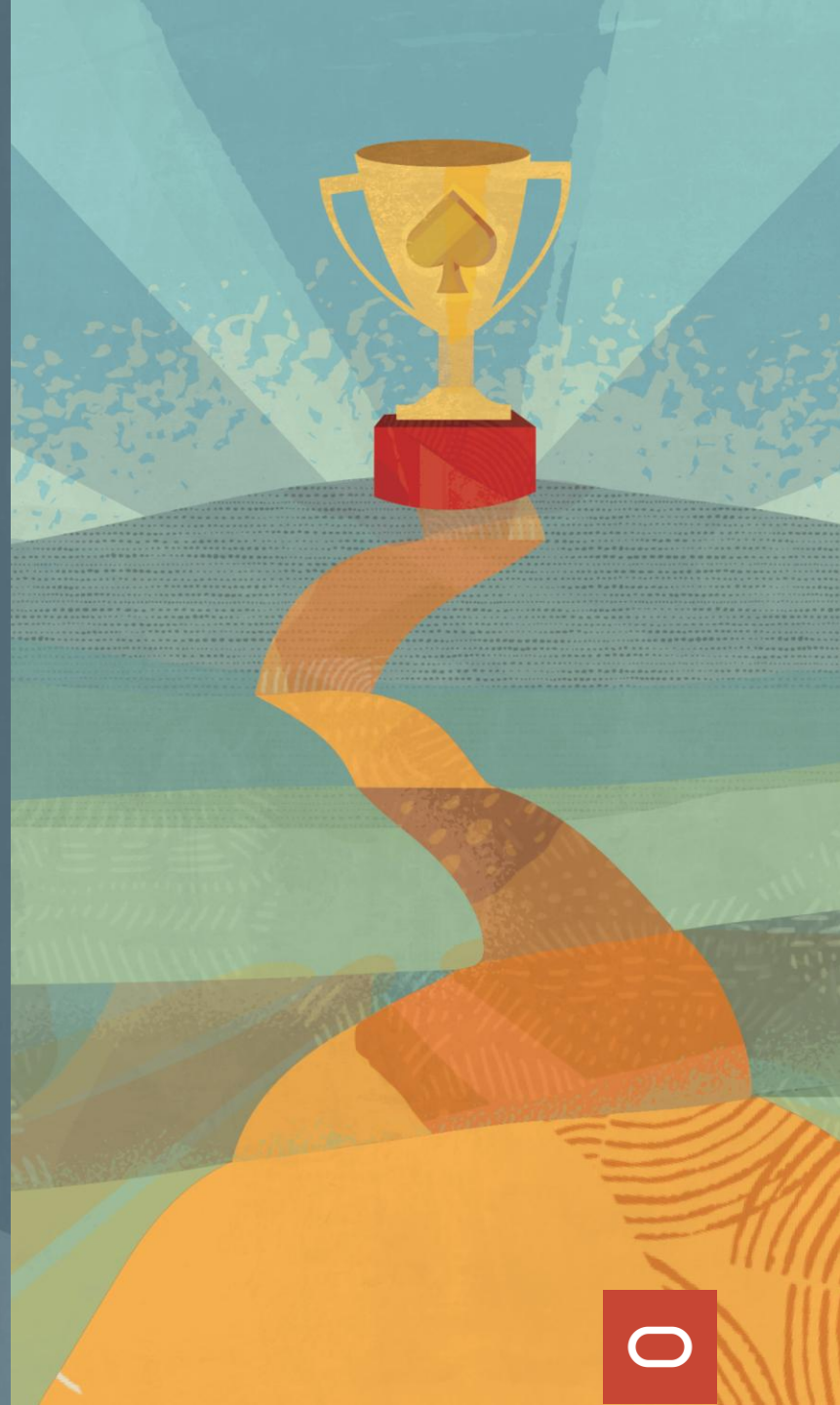
Travel support

@oracleace

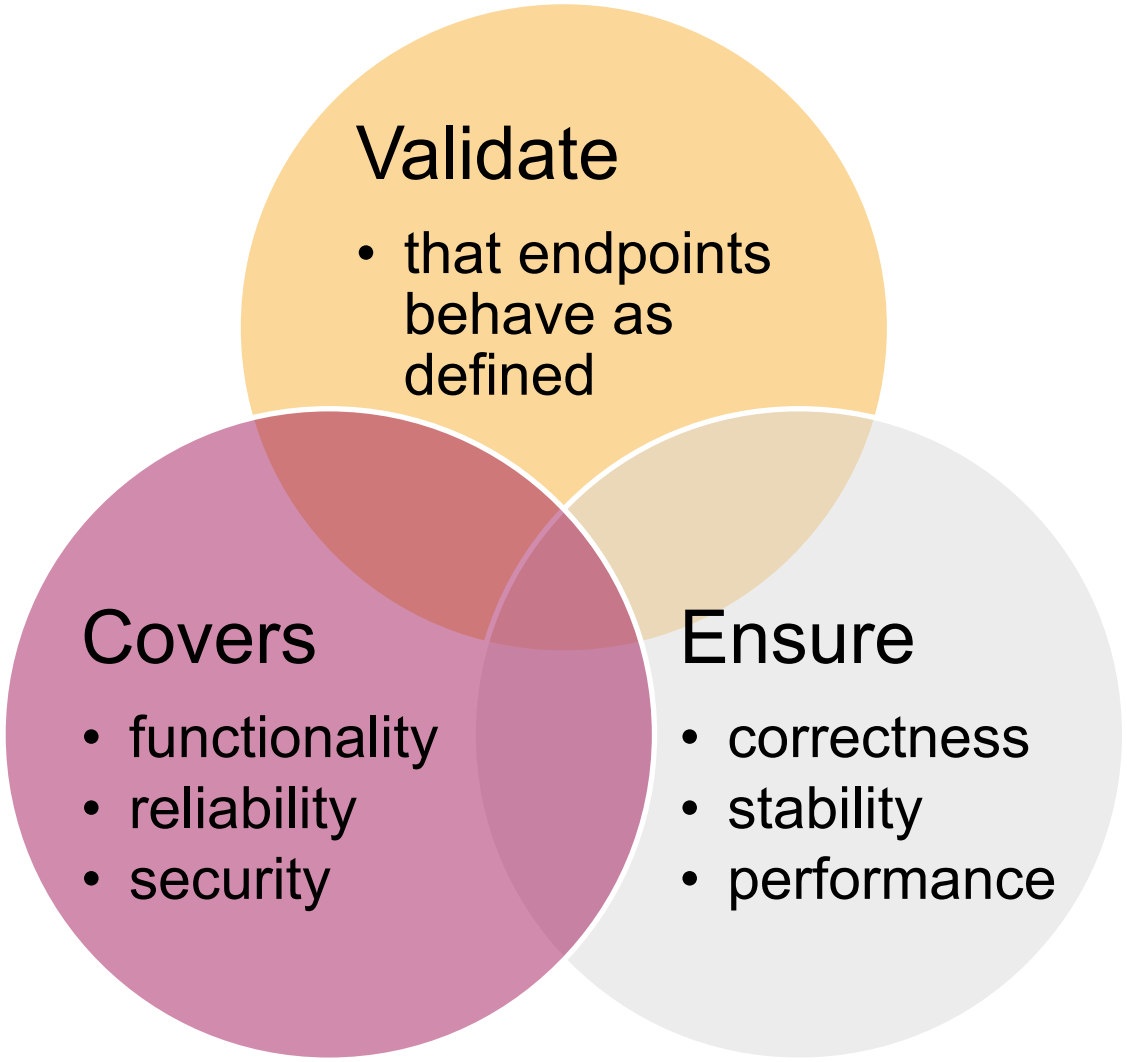
Linkedin.com/groups/72183

@oracleace.bsky.social

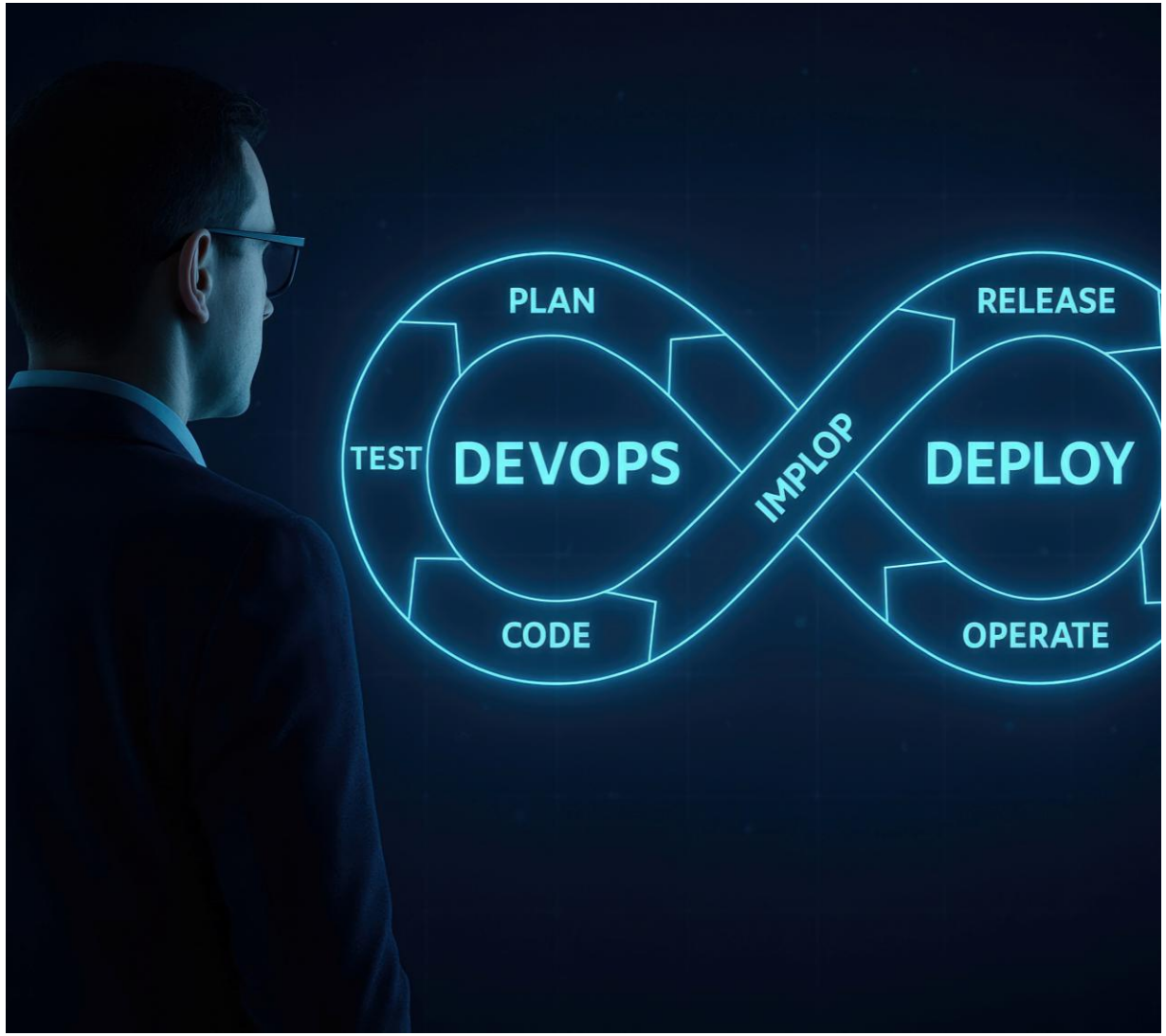
Learn more at:
ace.oracle.com



# What Is API Testing?




# Why Test APIs?




Early detection of logic or integration errors

Confident deployments across environments

Automation-ready (CI/CD)

Supports regression prevention

# How To Test APIs

**Functional tests**
- verify results for given inputs

**Integration tests**
- check API + DB behavior

**Performance tests**
- measure under load

**Contract tests**
- validate against OpenAPI / JSON Schema

# Tools & Context (Preview)

## REST clients

- Postman
- Insomnia

## Unit tests

- utPLSQL

## Load tests

- JMeter
- K6

## Contract validation

- JSON Schema

# Unit Testing

# Why Unit Tests Matter

**Verify**
- PL/SQL logic independently of ORDS

**Catch**
- regressions early

**Improve**
- developer confidence
- code quality

**Enable**
- automated CI/CD quality gates

# What is utPLSQL


Open-source testing framework for PL/SQL

Similar to JUnit, but for Oracle code

Provides assertions, setup/teardown, and reporting

Integrates with SQL Developer, SQLcl, Jenkins, GitHub Actions

# Writing Tests

```
CREATE OR REPLACE PACKAGE test_api AS
  --%suite(Test API Business Logic)
  --%test(should return valid JSON response)
  PROCEDURE test_response;
END;
/
CREATE OR REPLACE PACKAGE BODY test_api AS
  PROCEDURE test_response IS
    l_result CLOB;
  BEGIN
    l_result := api_pkg.get_data;
    ut.expect(l_result).to_match('.*"status":"OK".*');
  END;
END;
/
-- To run the tests, use the following command in SQL*Plus or SQL Developer:
BEGIN
  ut.run('test_api');
END;
/
```

# Unit Tests: Demo


API Tests with REST Clients
400 Bad Request
19.19 s
Visualize

# REST Client Basics

## REST clients

- simulate consumers of your ORDS APIs

## Send

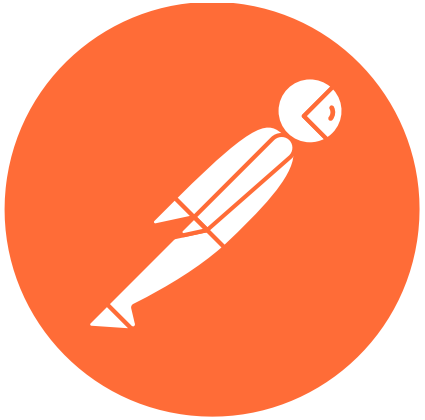
- HTTP requests

## inspect

- Responses

## Ideal for

- manual tests
- Prototyping
- debugging



POSTMAN

Insomnia

# cURL – the mother of all REST Clients

## Lightweight tool for HTTP requests

- CLI tool
- Library

## Installed

- on almost every system

## Perfect for

- quick tests
- automation

## Forms the foundation

- for many REST tools
  - Postman
  - Newman
  - K6

```
curl -s http://localhost:8080/ords/demo/todo/ | jq

curl -s -X POST -H "Content-Type: application/json" \
     -d '{"title":"from curl"}' \
     http://localhost:8080/ords/demo/todo/ | jq
```

# Scripting in REST Clients

```
// pm is the Postman object
pm.test("Status test", function () {
    pm.response.to.have.status(201);
});
// Check if response is valid JSON
pm.test("Response is valid JSON", function () {
    pm.expect(function () {
        JSON.parse(pm.response.text());
    }).not.to.throw();
});
// Stores the jobId in a collection variable
pm.collectionVariables.set("jobId", pm.response.json().jobId);

pm.sendRequest({
      url: url,
      method: 'POST',
      header: {'Content-Type': 'application/x-www-form-urlencoded'},
      body: {mode: 'raw', raw: data}
  },  function(err, response) {
      // Set the environment variable for token
      // [...]
      console.log("new token: ", payload);
  });
```

**Use JavaScript**

- for assertions and automation

**Validate responses**

- status codes
- body content
- headers

**Chain requests**

- for end-to-end workflows

# Managing Environments

**Store variables**

| base URLs | credentials | tokens |
|---|---|---|

**Switch between environments easily**

| dev | test | prod |
|---|---|---|

**Use dynamic values**

| for authentication | or test data |
|---|---|

# Automating Test Runs

## Run

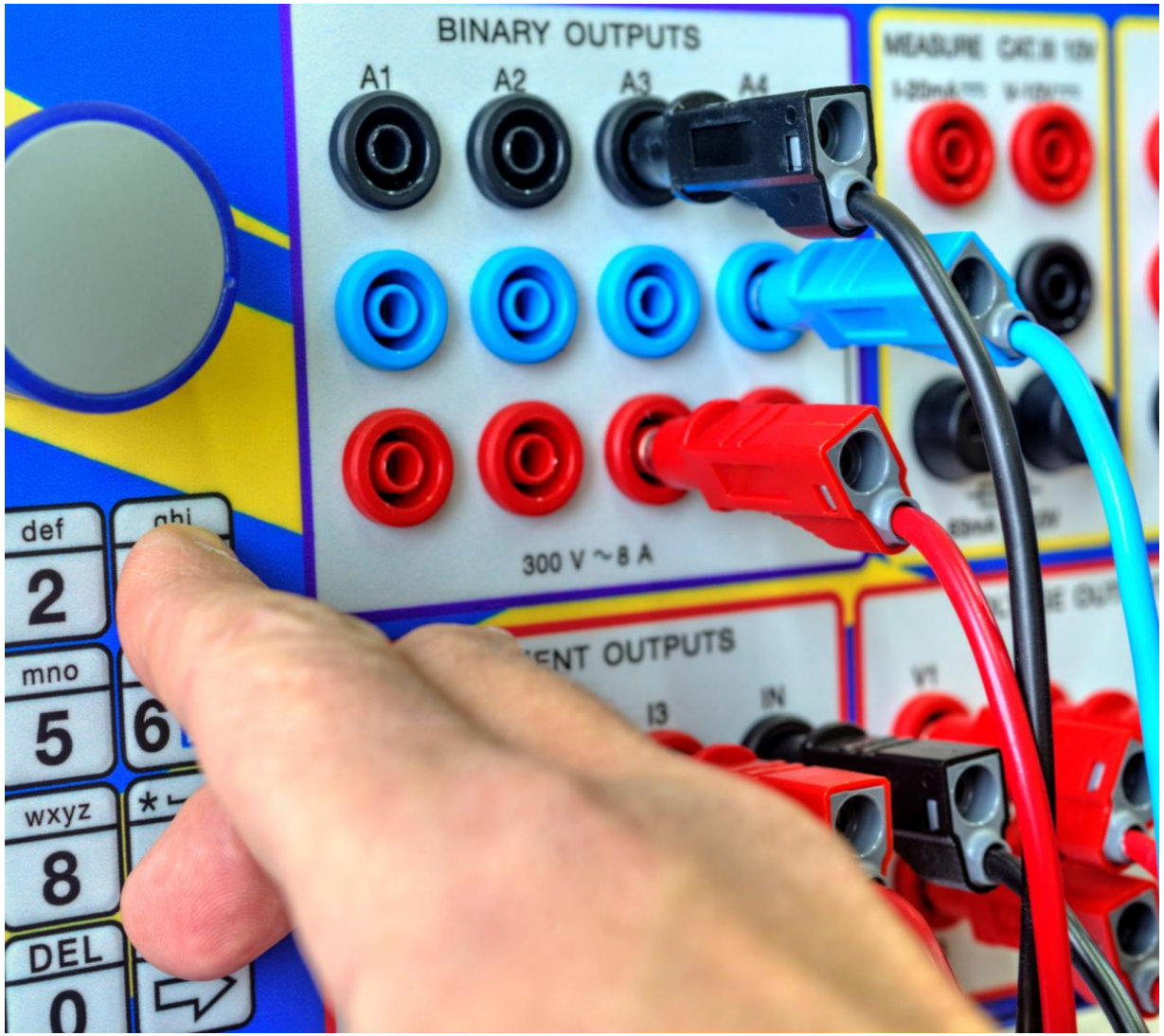
- full collections of tests automatically

## Integrate

- with Newman (Postman CLI runner) for CI/CD pipelines

## Export

- results as reports (HTML, JUnit XML)



# API Testing with Rest Clients

# Stress and Load Tests

# Why Stress and Load Tests

| | |
|---|---|
| Validate | performance |
| | under real traffic |
| Detect | bottlenecks |
| | before users do |
| Ensure | stability |
| | scalability |
| | resilience |
| Essential | before production |
| | deployment |

# Load vs Stress Testing

## Load Test

### Goal

- Verify normal operation under expected users

### Example

- 100 users → stable responses

## Stress Test

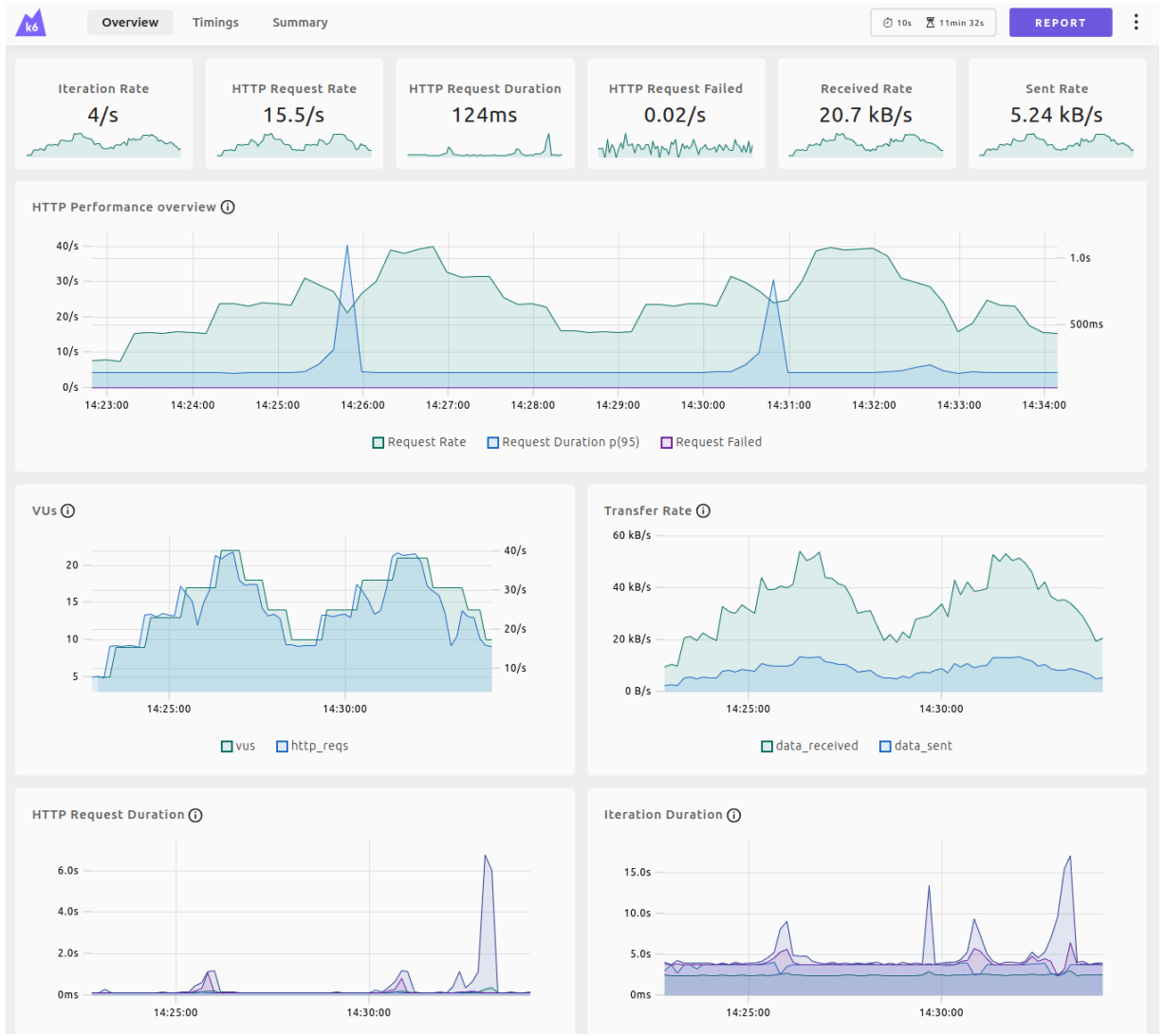
### Goal

- Push beyond limits

### Example

- 500 users → observe failure behavior

# Grafana k6 Overview




Modern, developer-friendly load testing tool

Open Source

Scriptable Engine written in Go

Tests authored in JavaScript or TypeScript

CLI + cloud execution

Great for CI/CD integration

https://grafana.com/docs/k6/latest/results-output/web-dashboard/

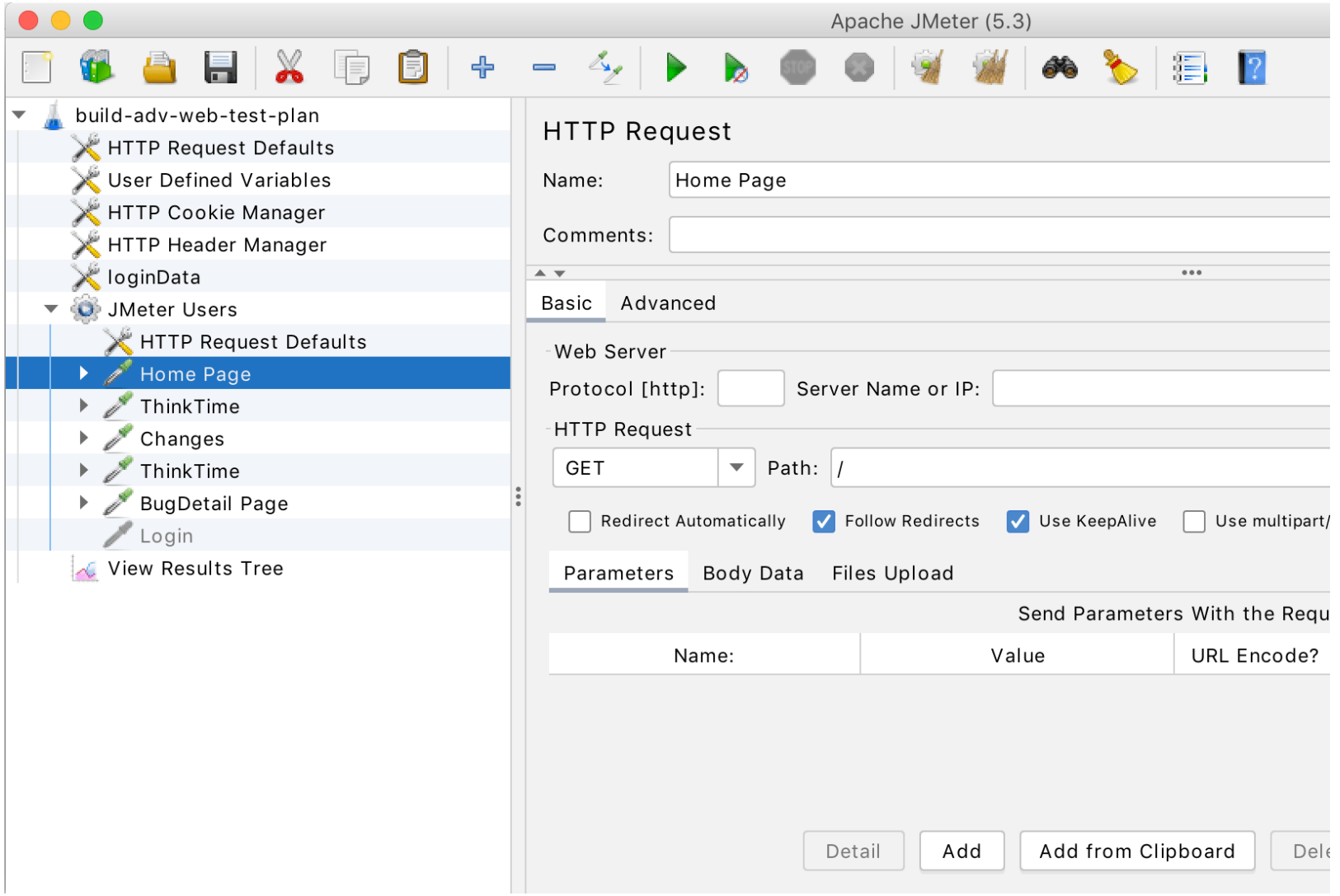
# Apache JMeter Overview

GUI-based open-source load tester

Broad protocol support (HTTP, JDBC, MQ…)

Good for complex test plans and graphs

Produces rich reports and charts



# API-Stress- and Load-Tests


**very soon …**

# JSON Schemas

# Why JSON Schema

## Define

- structure and rules for API payloads

## Validate

- requests and responses automatically

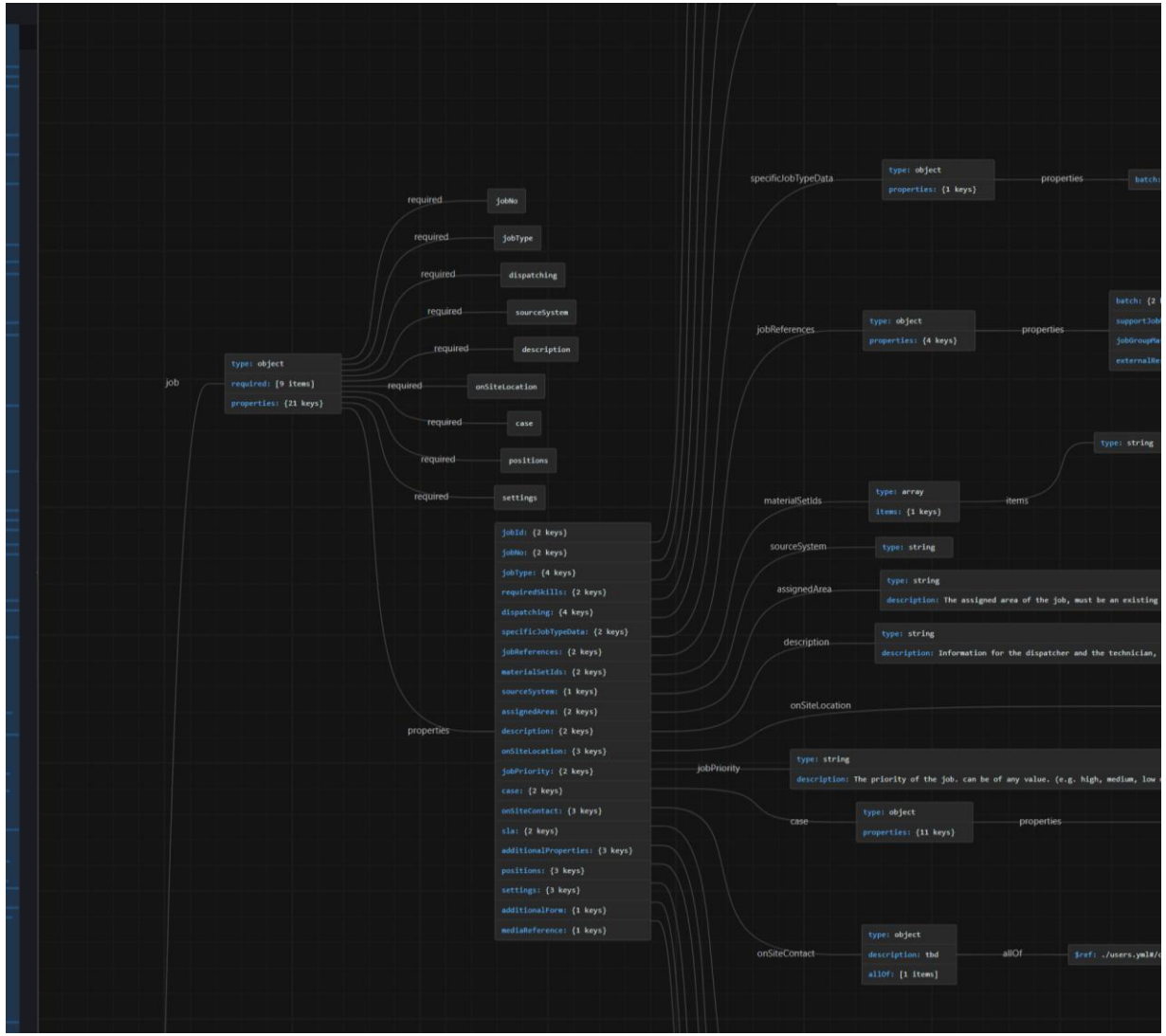
## Detect

- breaking changes early

## Enable

- contract-based development

# Testing JSON Payloads




**Validate incoming**

- payloads against schema

**Validate outgoing**

- responses for consistency

**Prevent**

- Nulls
- type errors
- missing fields

**Use schema**

- as single source of truth

# Automate Schema Extraction

## Extract JSON Schemas

- automatically from openapi.json

## Keep schema

- and API in sync

## Use scripts or CI steps

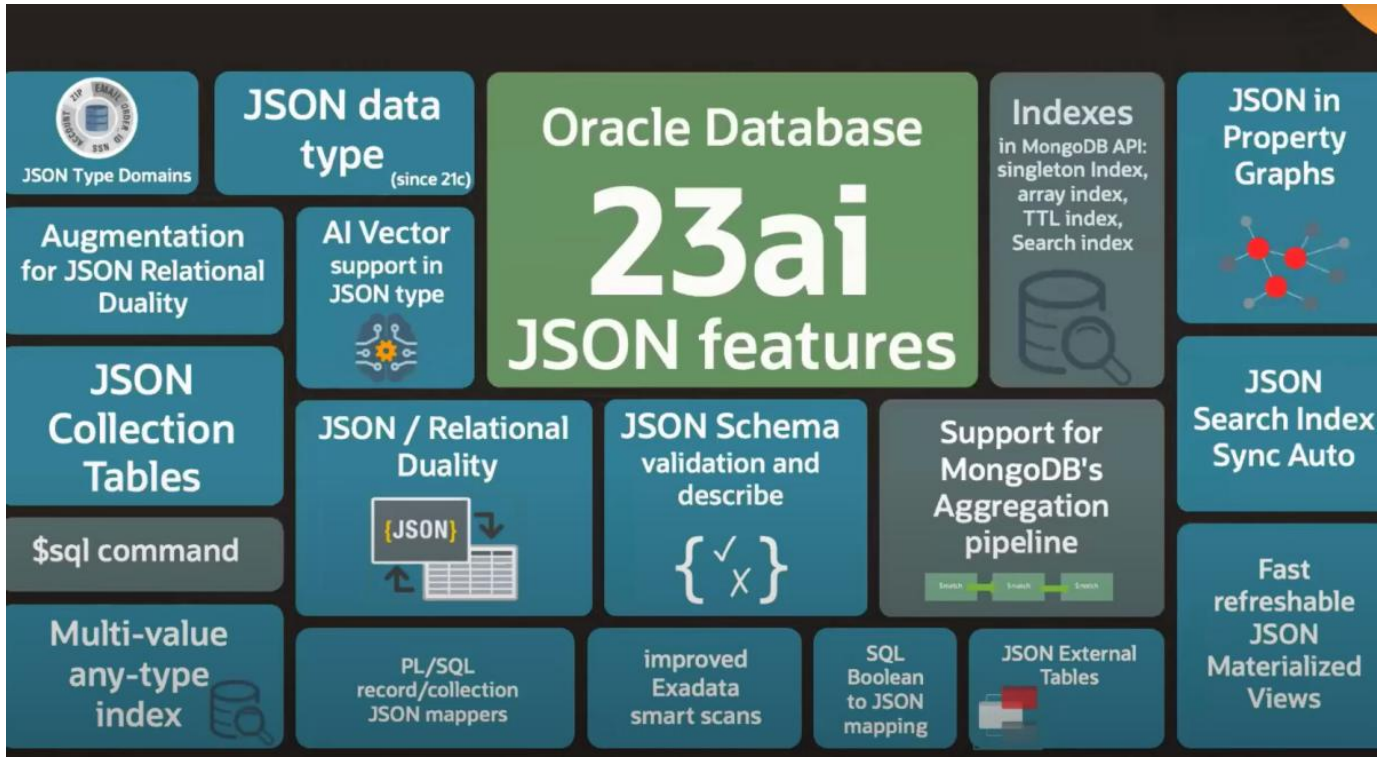
- for regeneration

## Commit schemas

- to repo for version control

```bash
#!/bin/bash
# Extract all JSON Schemas from OpenAPI spec
# Requires: jq (https://stedolan.github.io/jq/)

API_FILE="openapi.json"
OUT_DIR="./schemas"

mkdir -p "$OUTPUT_DIR"

# Extract each schema from the OpenAPI spec
jq -r '.components.schemas | keys[]' "$API_FILE" | while read -r SCHEMA
do
  jq ".components.schemas[\"$SCHEMA\"]" "$API_FILE" > "$OUT_DIR/${SCHEMA}.json"
  echo "Extracted schema: $SCHEMA"
done

echo "✅ All schemas exported to $OUT_DIR/"
```

# JSON Schema in Oracle 23ai / 26ai




**Built-in support**
- for JSON Schema validation
- IS JSON with VALIDATE USING <schema>
- for column constraints

**Use**
- DBMS_JSON_SCHEMA.VALIDATE_REPORT
- for runtime validation

**Store and manage schemas**
- in DB tables

# JSON Schema Validation with API Gateways

Oracle 19c has no JSON Schema Support

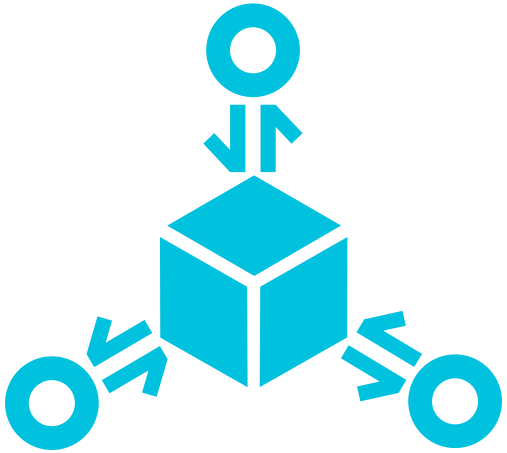
API Management Systems to the the Rescue

Schema policies derived from OpenAPI spec

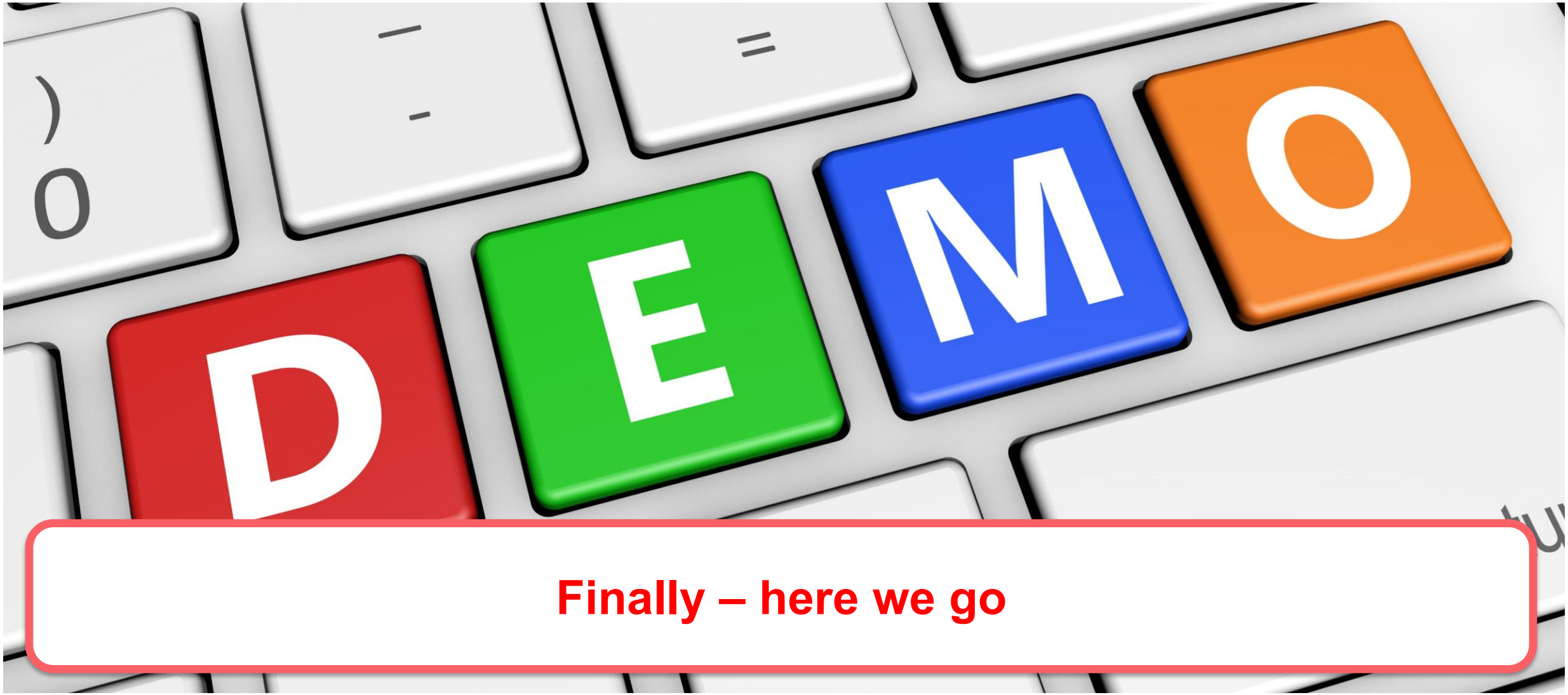
Validate requests/responses before reaching backend

Central enforcement for multiple APIs

Combine with monitoring and alerts

# JSON Schema Validation




**Finally – here we go**

# Put your ORDS API to the Acid Test

- Validate your ORDS APIs on every layer
- Use **utPLSQL** for backend unit tests
- Test API endpoints via **REST clients**
- Simulate real-world usage with **K6** or **JMeter**
- Enforce structure, data integrity and API contracts with **JSON Schema**