App Connect v12

Unit testing with a Pipeline Example

—

Trevor Dolby
Architect, IBM App Connect Enterprise
tdolby@uk.ibm.com

Contents

App Connect Enterprise (ACE) v12
Background

Testing integration solutions
Challenges and techniques

ACE in pipelines
DevOps and containers

Demo pipeline
GitHub, Tekton, and CI

ACE v12

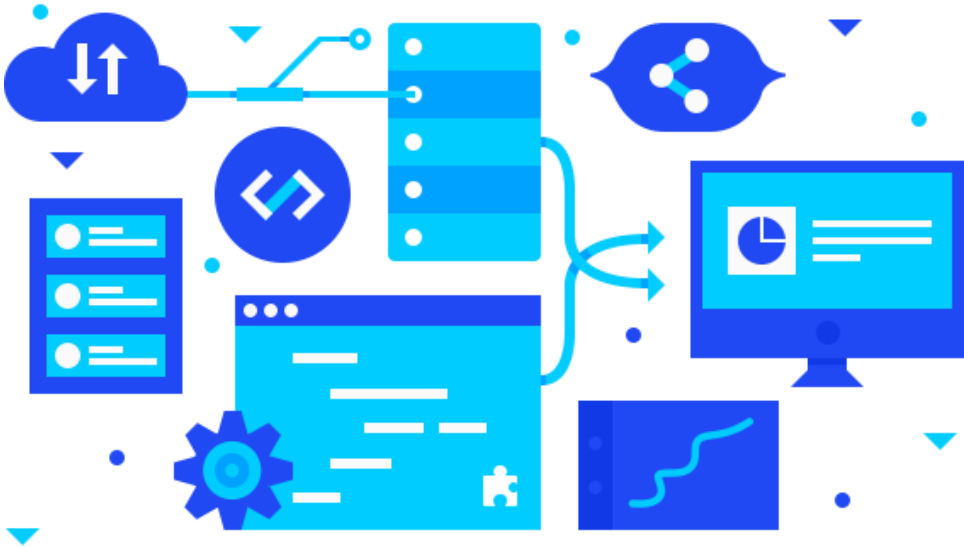
What is integration?

Integration is how systems and applications are connected to enable the systems and applications to exchange data with each other or with other connected and integrated applications and systems no matter what the data and/or protocols are being used between them.

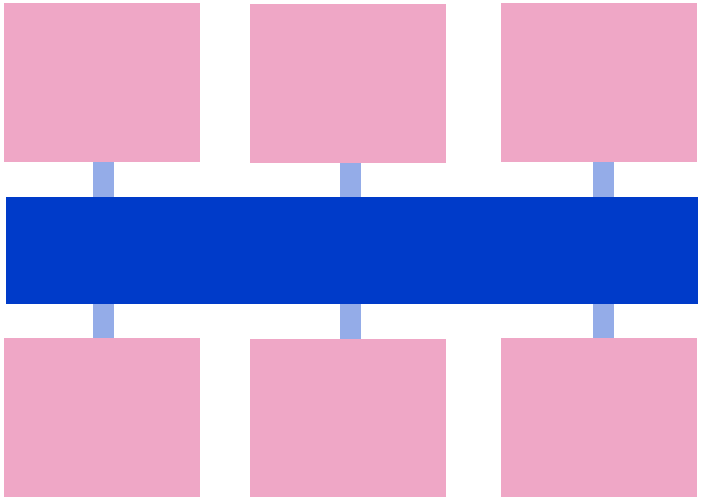
Provides:

1. Connectivity
2. Routing
3. Transformation

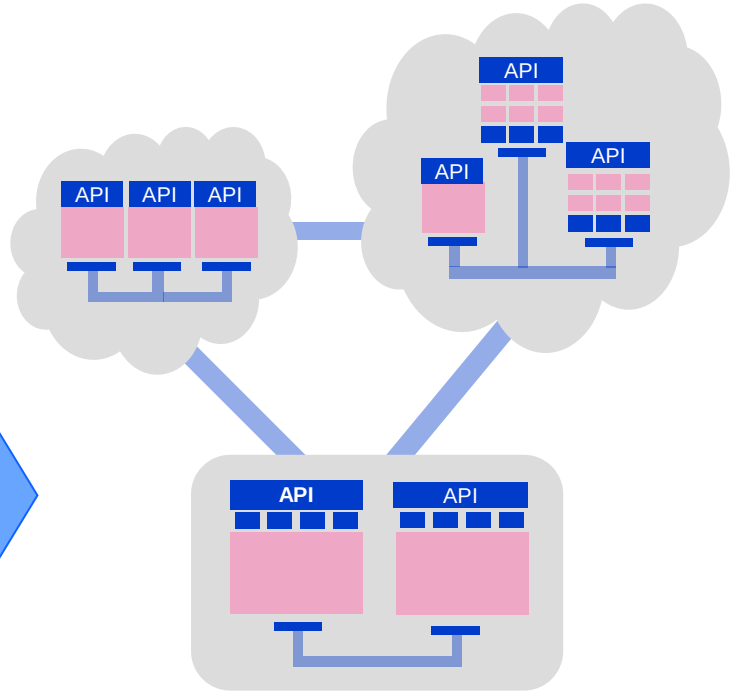
to enable sharing of data between applications and services securely irrespective of the data format and protocol that these system and services use between them



Agile Integration Architecture



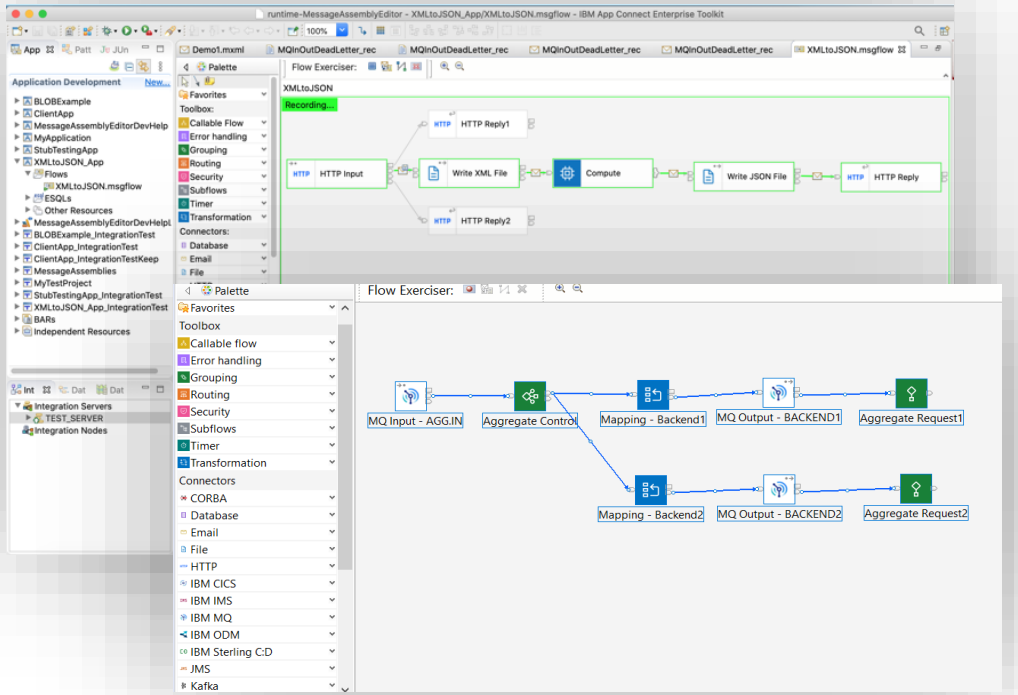
Traditional application integration
(centrally provisioned and administered silo)



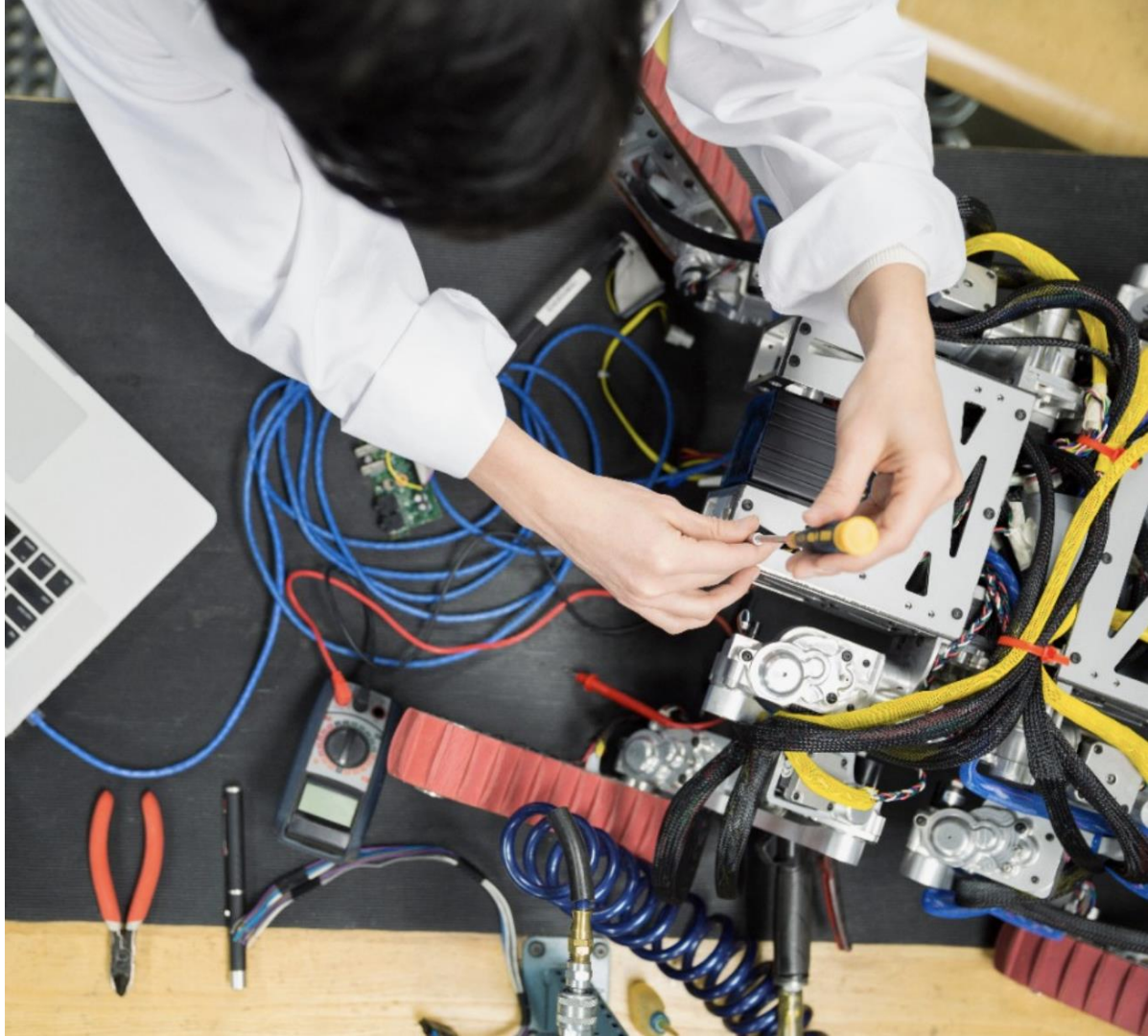
A. **Fine-grained deployment**
B. **Decentralized ownership**
C. **Cloud native infrastructure**

App Connect Enterprise v12

- Built-in nodes encapsulate transports, technologies and applications:
  - Connecting with client applications
  - Routing messages
  - Transforming and enriching messages
  - Processing events
  - Handling errors in message flows
  - Reduce the amount of custom code required to implement solutions
  - Built-in facilities like activity log and resource statistics
- High performance and scalability
- Feature rich Message Modelling



Testing integration solutions



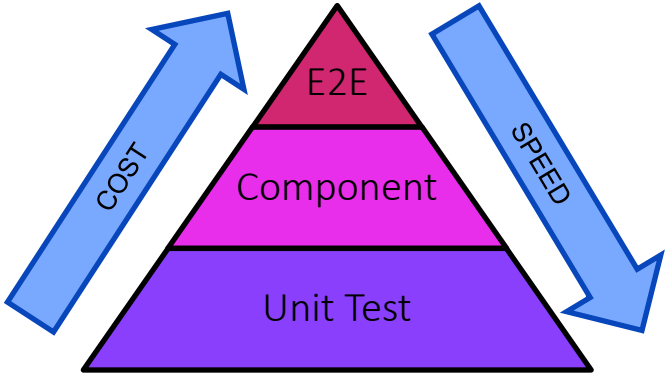
Testing: prove business requirements met

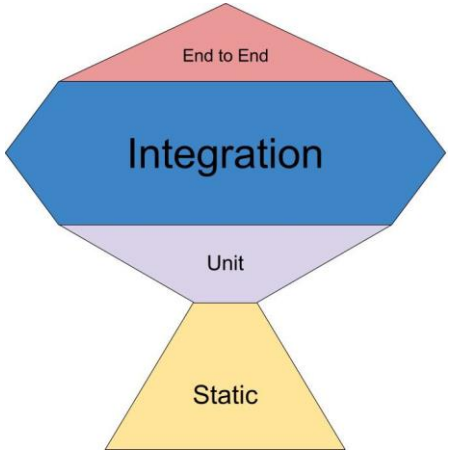
Functionality, quality, reliability, serviceability, etc

Integration has more external interactions
- More infrastructure (hardware, network, etc) changes
- External service upgraded
- Back-end service changes

Requires more test approaches than ordinary application development, but similar techniques apply

No one test shape fits everyone
- Traditional test pyramid principles still valid
- Realistic testing looks less like an ideal pyramid and more like a test trophy or microservice honeycomb





Test a single message flow node

- Test one node
  - Use simple or recorded messages
  - No down flow propagation at test time
  - No interaction with the outside world
- Check map or ESQL changes haven’t broken behaviour
- Identifying issues & analysis scoped to a single node
- Spy & assert correct behaviour

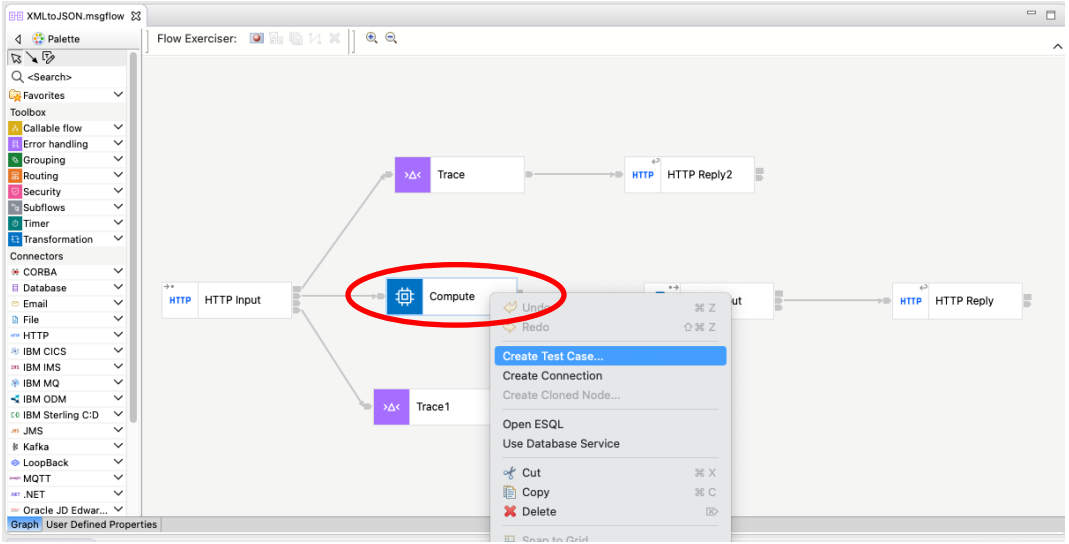
Test external transport link

- Test one node
  - Interacts with the outside world
- Check DB connection works, and that schemas are compatible
  - Not dependent on REST service
- Could run a similar test on the REST call without needing a DB

Create and run tests during development

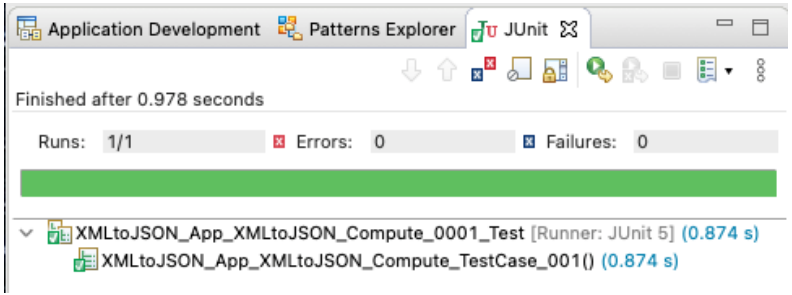
Create tests for message flows and nodes:

- Generated automatically from message models associated with a flow
- Generated using actual messages recorded during development
- Written by developers starting from sample tests, tutorials, etc.

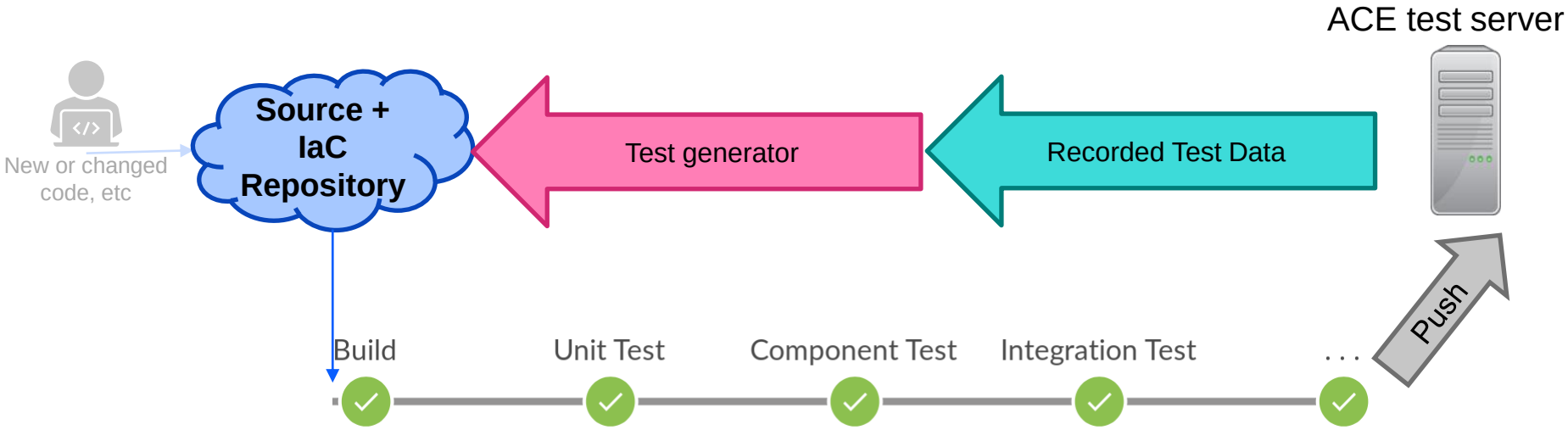


Run tests without needing infrastructure:

- Eclipse toolkit JUnit runner
- Scripted runs from command line
- No need to create and maintain a server
  - Similar to function-as-a-service or serverless, where the server starts only when needed.



Capture existing messages to create tests



- Server setting allows “bulk” recording of messages through running flows
- **`ibmint generate tests`** command creates unit tests based on recorded messages

```
Gabriels-MBP:server0 gabrielroels$ ibmint generate tests --recorded-messages recordedMessages  --java-class com.ibm.MyTests --outp
ut-test-project generatedTests/LotsOfTests
BIP15215I: Found '7542' recorded messages in directory '/Users/gabrielroels/Dev/SISworkdirs/server0/recordedMessages'.
BIP15216I: Generating test project from recorded messages.
BIP15183I: Successfully generated test project containing '5615' test-cases and written it to '/Users/gabrielroels/Dev/SISworkdirs/
server0/generatedTests/LotsOfTests'.
BIP8071I: Successful command completion.
```

ACE in pipelines

Where does ACE fit?

Key goal is to fit in with any pipeline technology

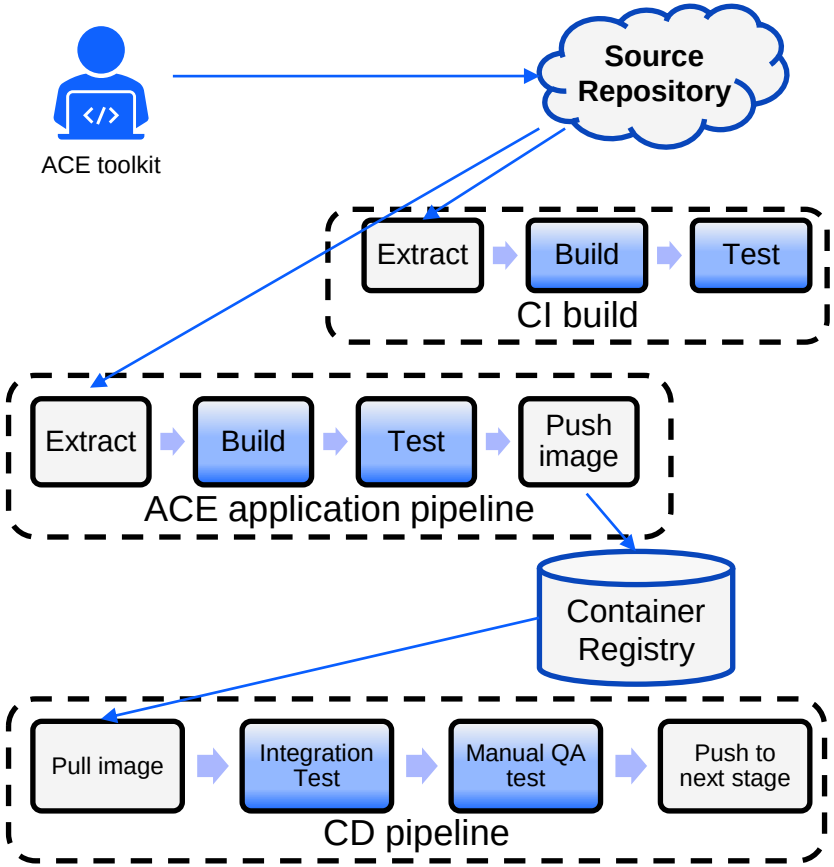
- Not trying to replace github, Jenkins, Tekton, ArgoCD, gitlab, etc, but will work with all of them

Stages that use ACE have lots of other tools in place also

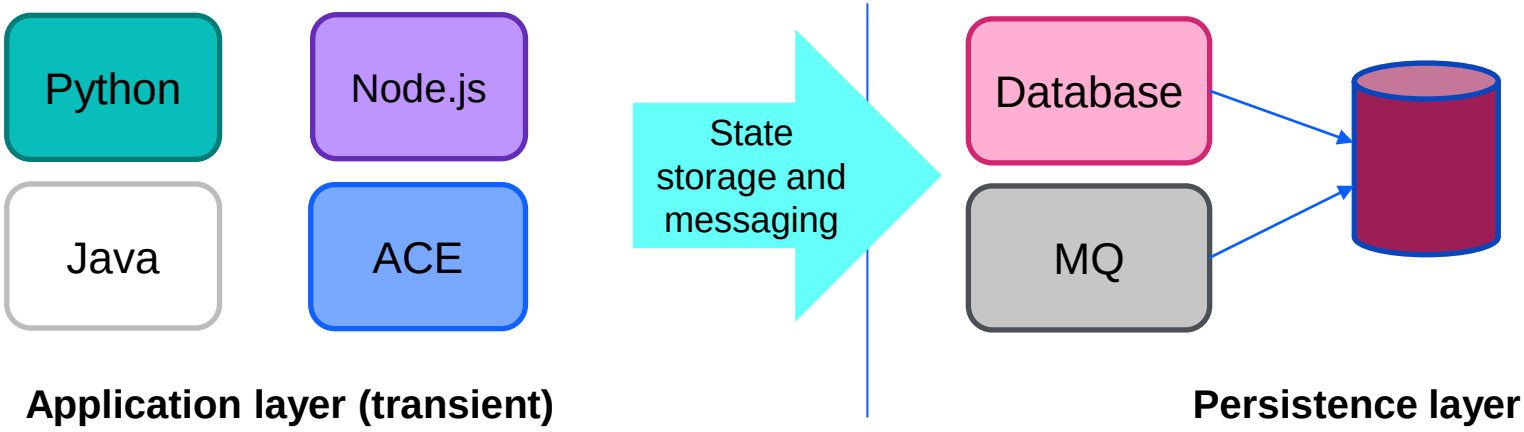
- Ant/Maven/Gradle/etc
- ACE can be run from anywhere
- IntegrationServer-based tests easy to run

ACE testing in multiple places

- Automated unit and integration tests



ACE and MQ infrastructure layers



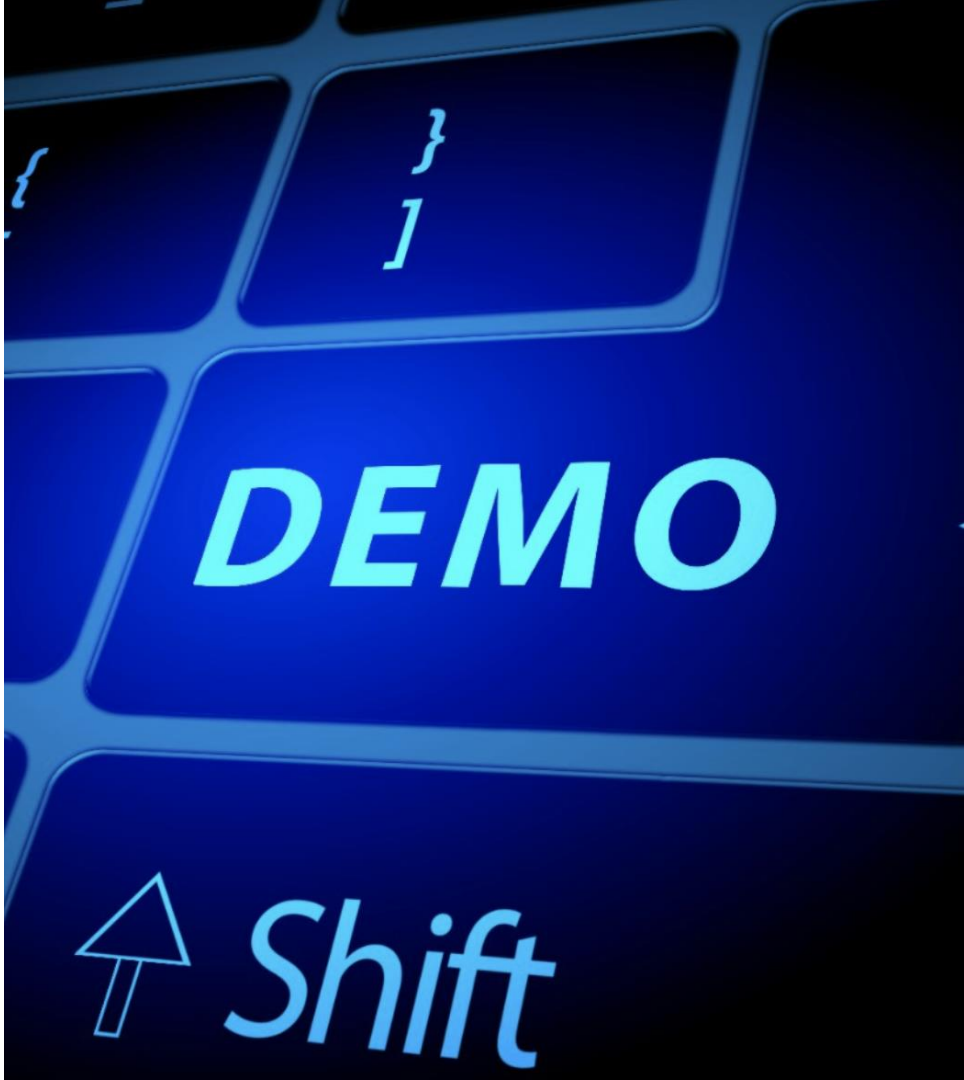
ACE does not store persistent data, and is best seen as application layer

- MQSI (long ago) would have fit on the other side

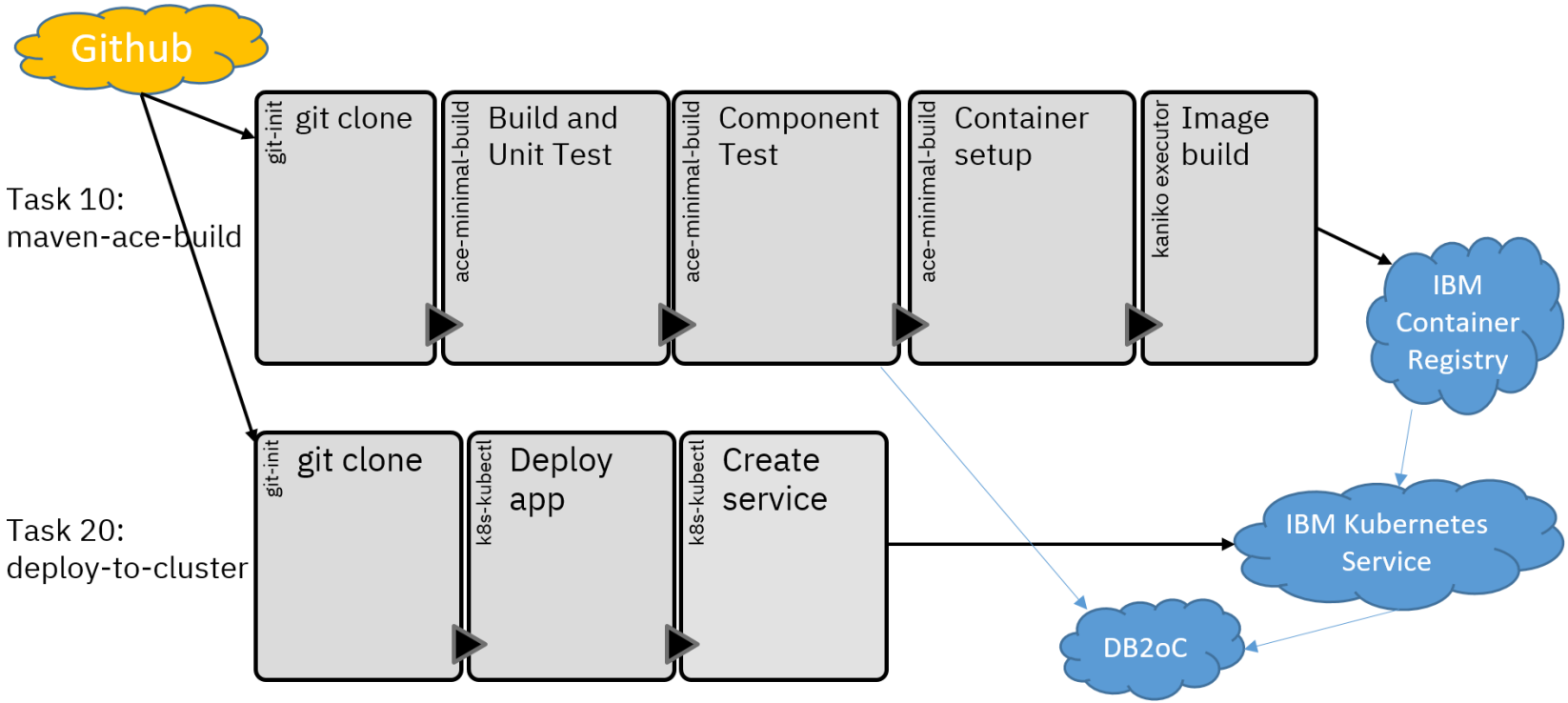
Code as well as configuration

- BAR files are not the same as MQSC files

Demo pipeline



Tekton, github, Maven, DB2onCloud, and IBM Cloud (free!)



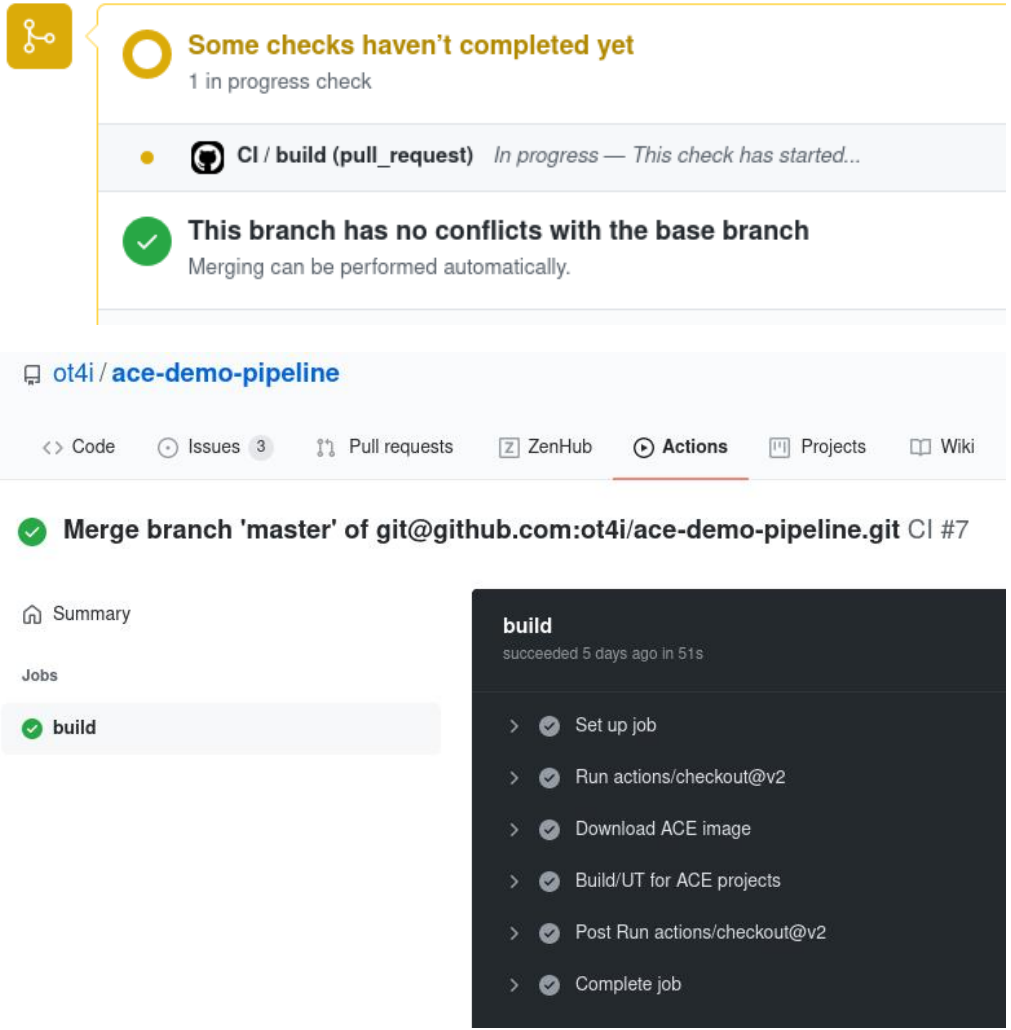
https://github.com/ot4i/ace-demo-pipeline

Continuous Integration

GitHub Action to unit test changes

Blocks changes that break tests from being merged

Would also be compatible with CircleCI, GitLab, TravisCI, etc.



Summary

Topic	Summary
ACE v12	Natural fit for current and future integration needs, combining proven integration technology with modern software development.
Testing integration solutions	Standard test approaches apply, plus additional techniques for integration complexity; tooling support and auto-generated tests enable agility.
ACE in a pipeline	Fits naturally in modern CI/CD pipelines and container infrastructure.
Demo pipeline	Public repo with CI/CD pipeline available to use for free.