



# Containerized Test Automation

*Sven Hettwer*  
*Software developer*

2018-10-10

# Agenda

- Testing of microservices
- CI/CD on container platforms
- Integration testing with Citrus
- Demo
- Summary



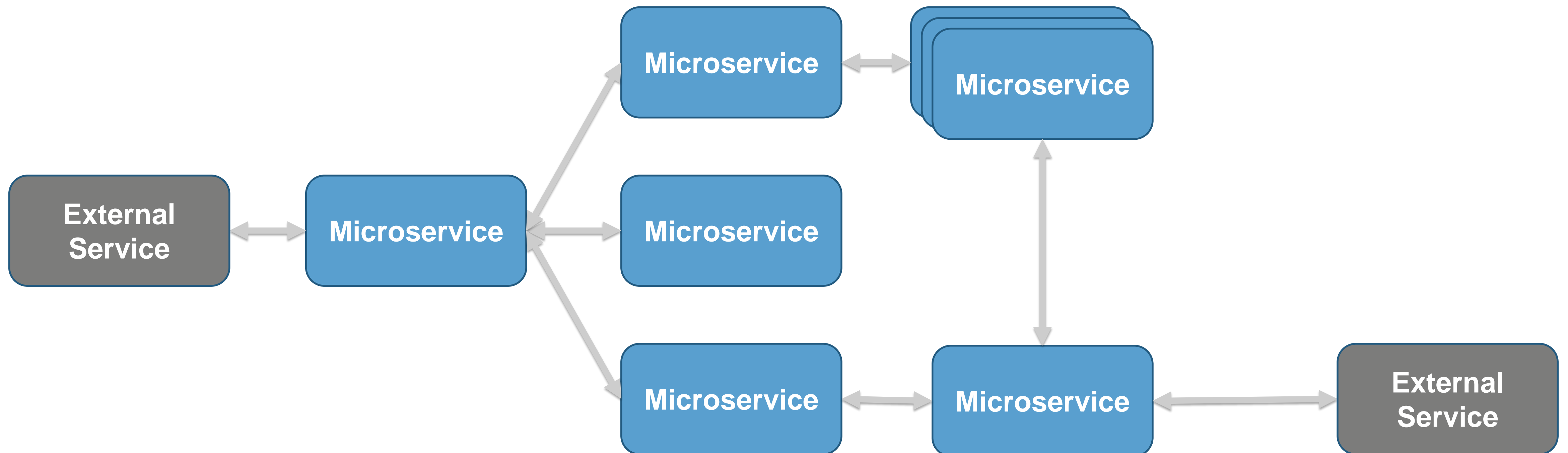
# Agenda

- Testing of microservices
- CI/CD on container platforms
- Integration testing with Citrus
- Demo
- Summary



# Testing of microservices

- Testing of microservices is... different
- Microservice architectures ease and complicate testing
- Different testing challenges compared to monoliths
- Different tech stack



# Testing of microservices – The good parts

- Isolated functionality
- Less functionality in a single service compared to monoliths
- Easy to start and deploy
- Message interfaces between services
- Decoupled services connected via network



# Testing of microservices – The bad parts

- Message interfaces between services
- Decoupled services connected via network
- A lot of third party systems
- Various message transports

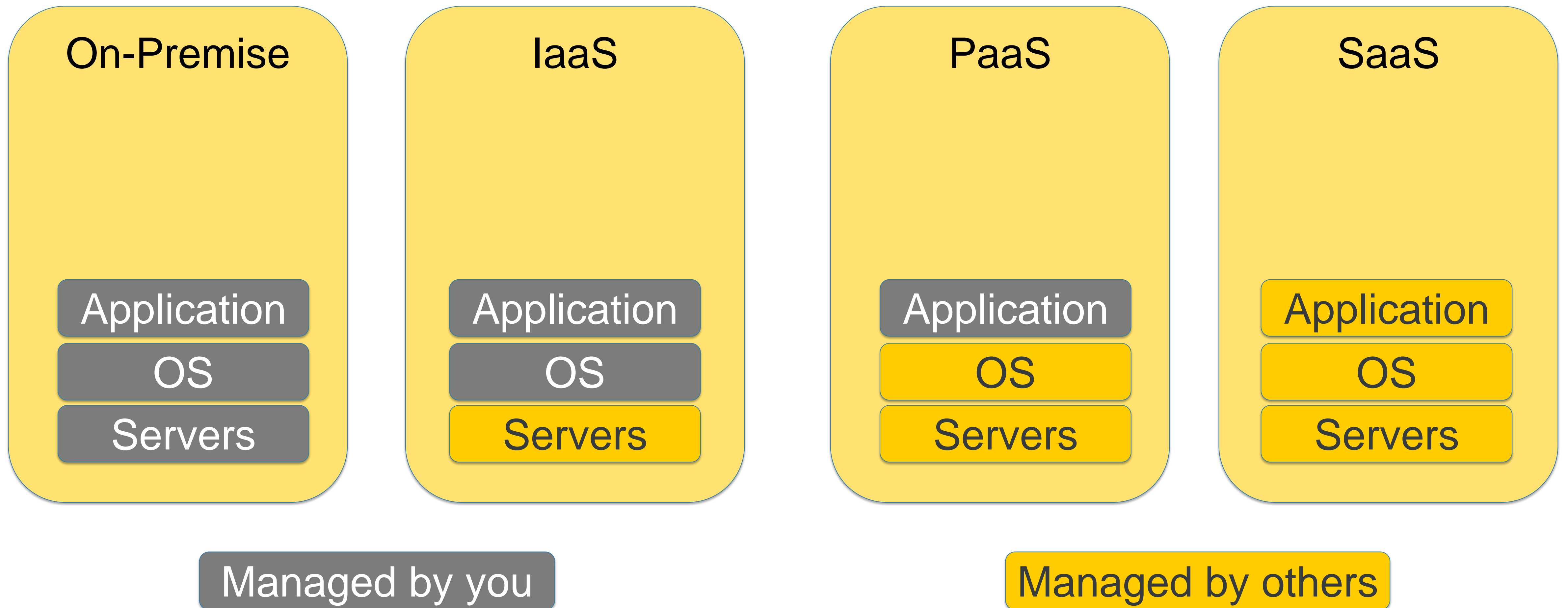


# Agenda

- Testing of microservices
- CI/CD on container platforms
- Integration testing with Citrus
- Demo
- Summary

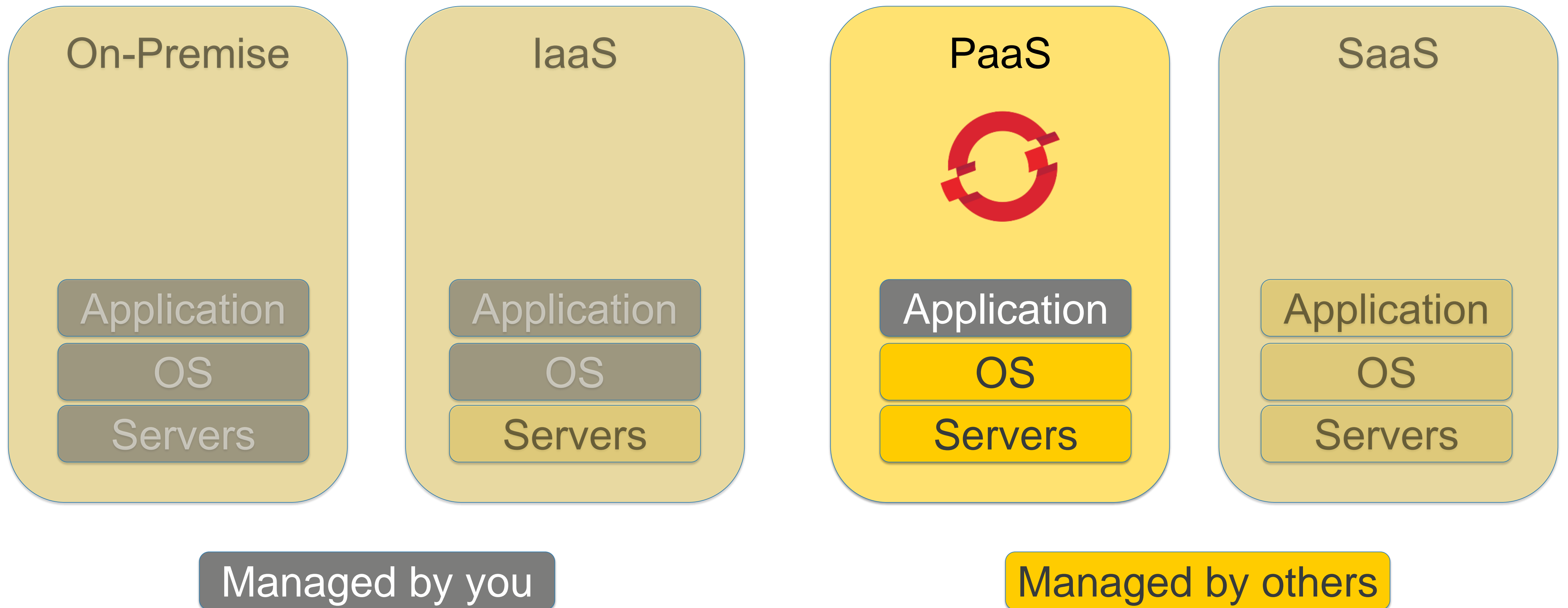


# CI/CD on container platforms – Container Platforms



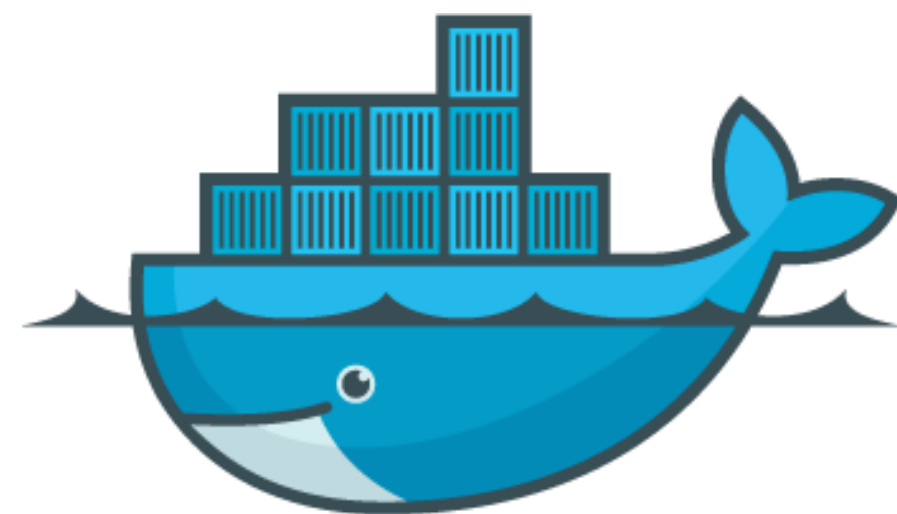
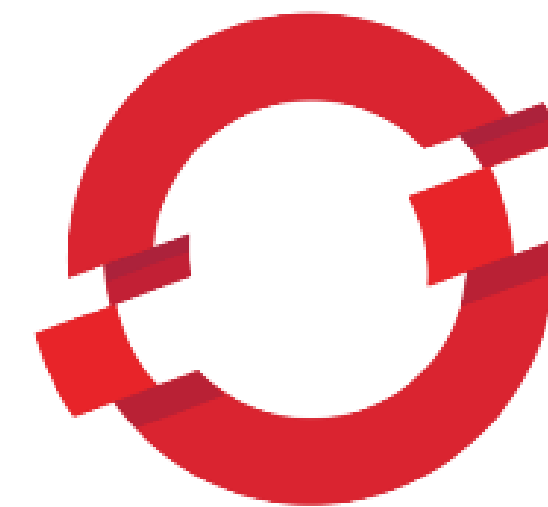


# CI/CD on container platforms – Container Platforms

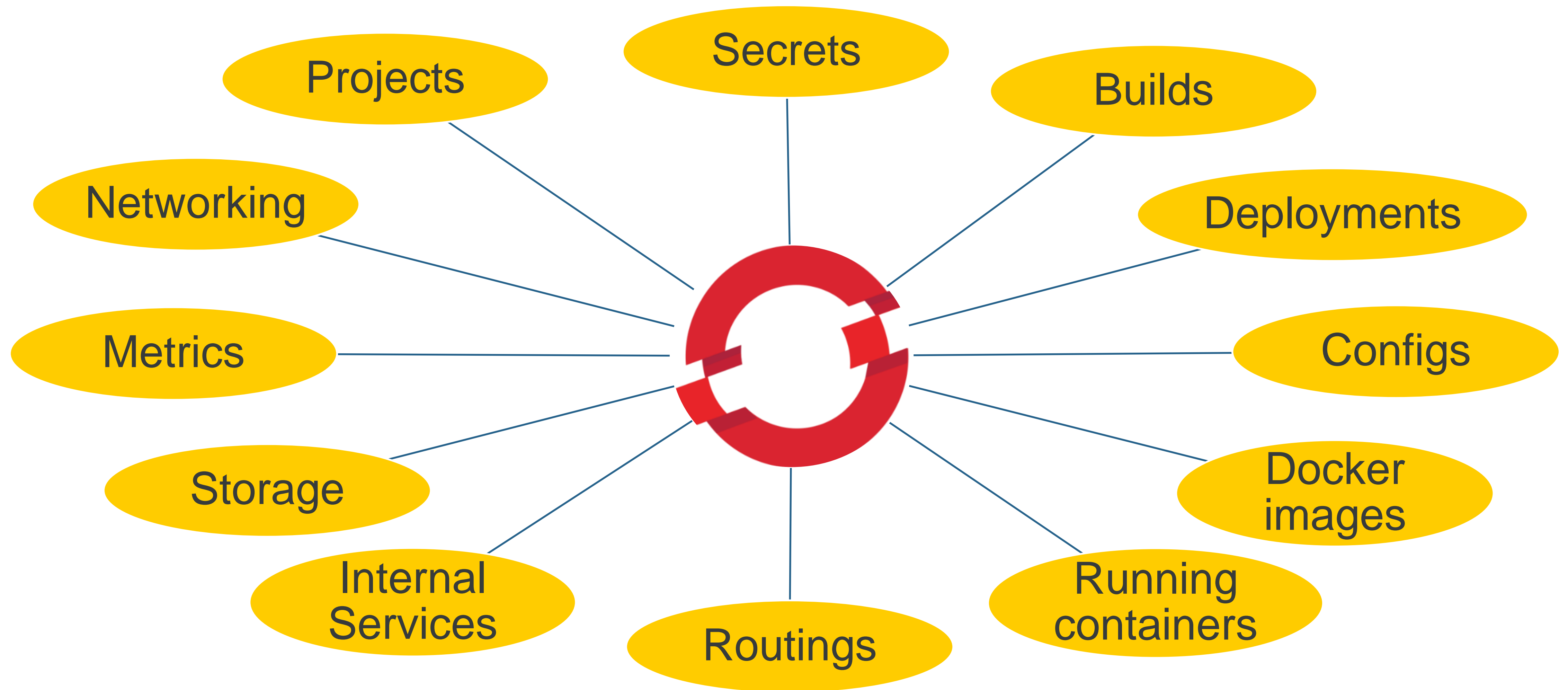


# CI/CD on container platforms – Container Platforms

- Developed and maintained by RedHat since 2011
- Open source container platform OKD (Apache Licence 2.0)
- Professional support options
- Public OpenShift cloud available
- Based on Kubernetes
- Uses Docker containers
- Additional platform specific features

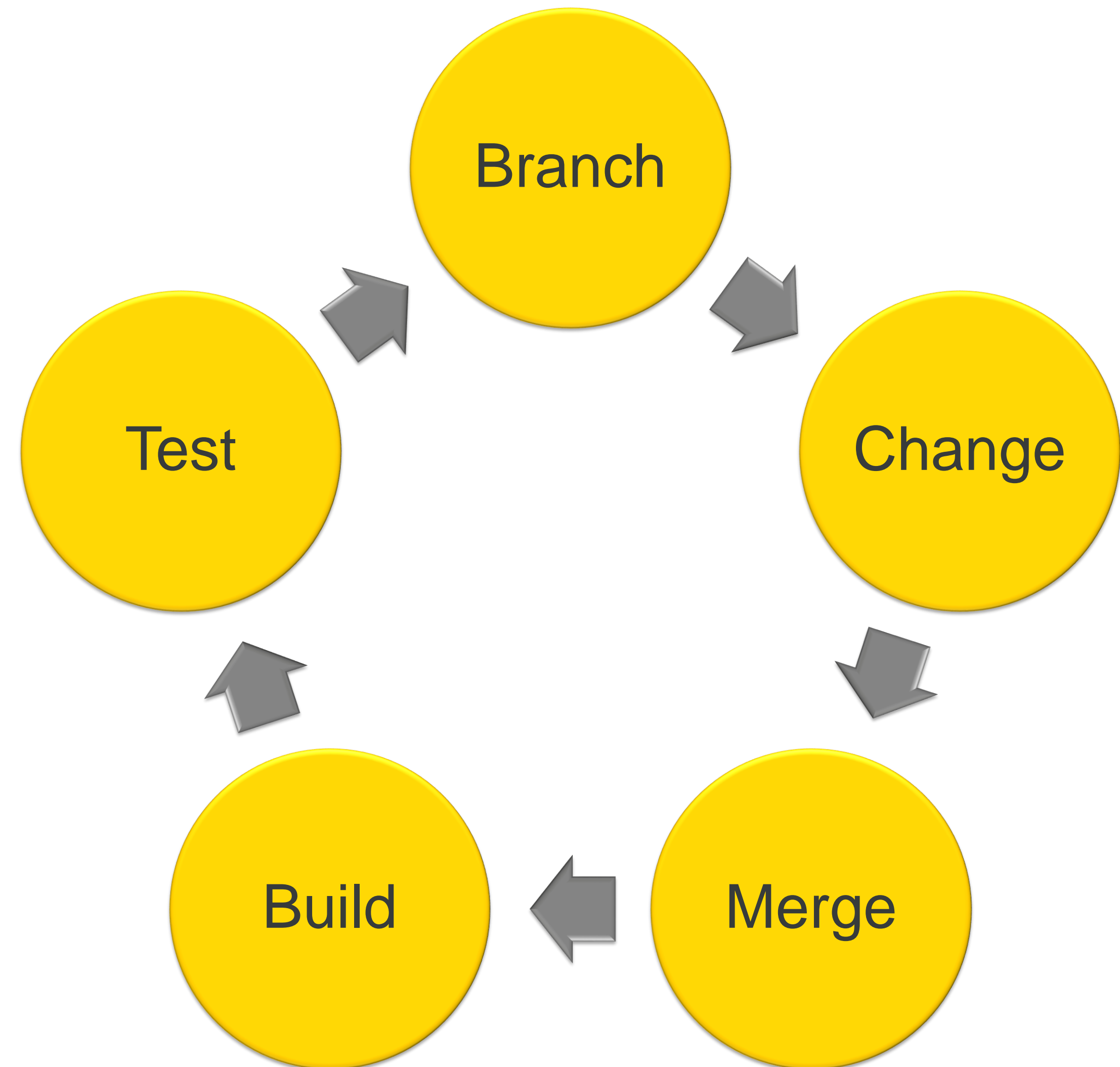


# CI/CD on container platforms – Container Platforms



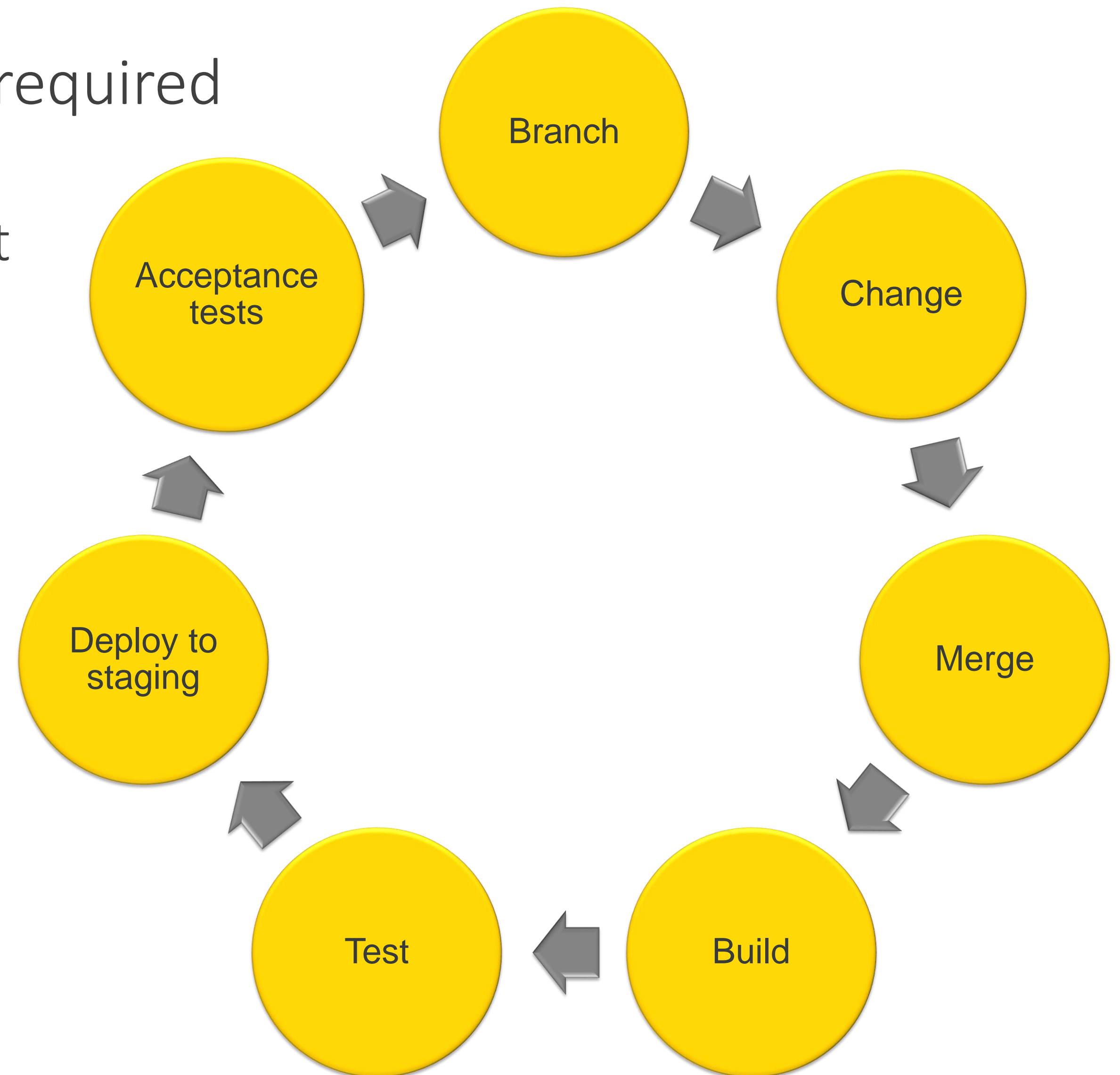
# CI/CD on container platforms – CI

- Born from extreme programming
- Merge your code early and often
- Build your code automated after change
- Test your code automated after change
- Goal
  - Early feedback on changes
  - Reduce merge conflicts



# CI/CD on container platforms – CD

- Release your software whenever it's required
- Extends continuous integration by
  - Deployment into a staging environment
  - Acceptance tests

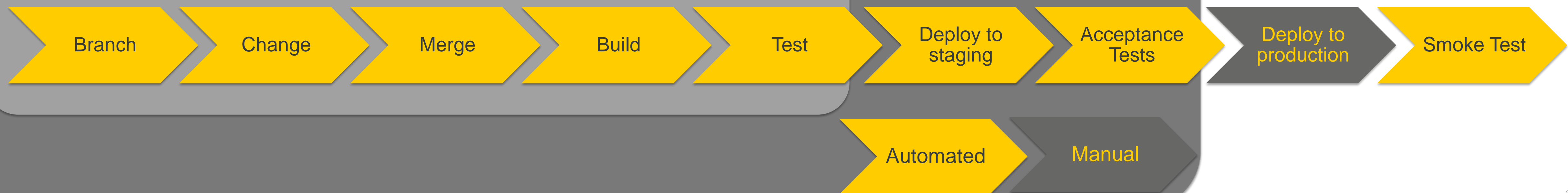


# CI/CD on container platforms – CI/CD pipelines

- Automate everything from code change to pre production
- Increase visibility of issues while building, testing, deploying your product
- Allows fast feedback loops
- Empowers you to deliver continuously
- Enables continuous deployment

## Continuous delivery

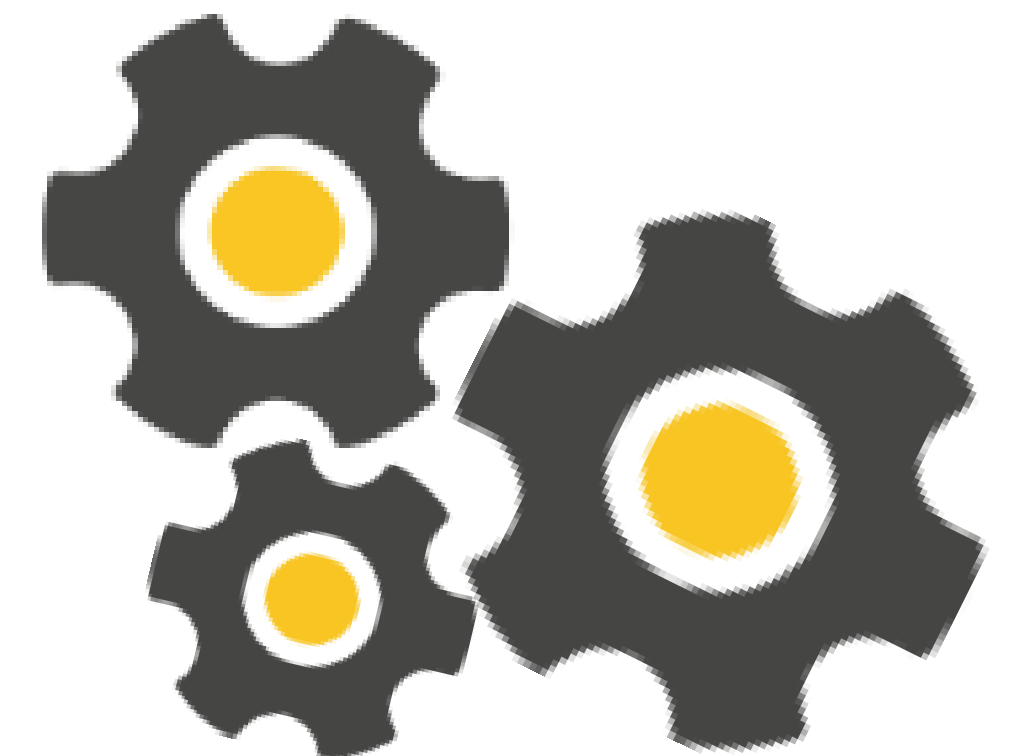
### Continuous integration



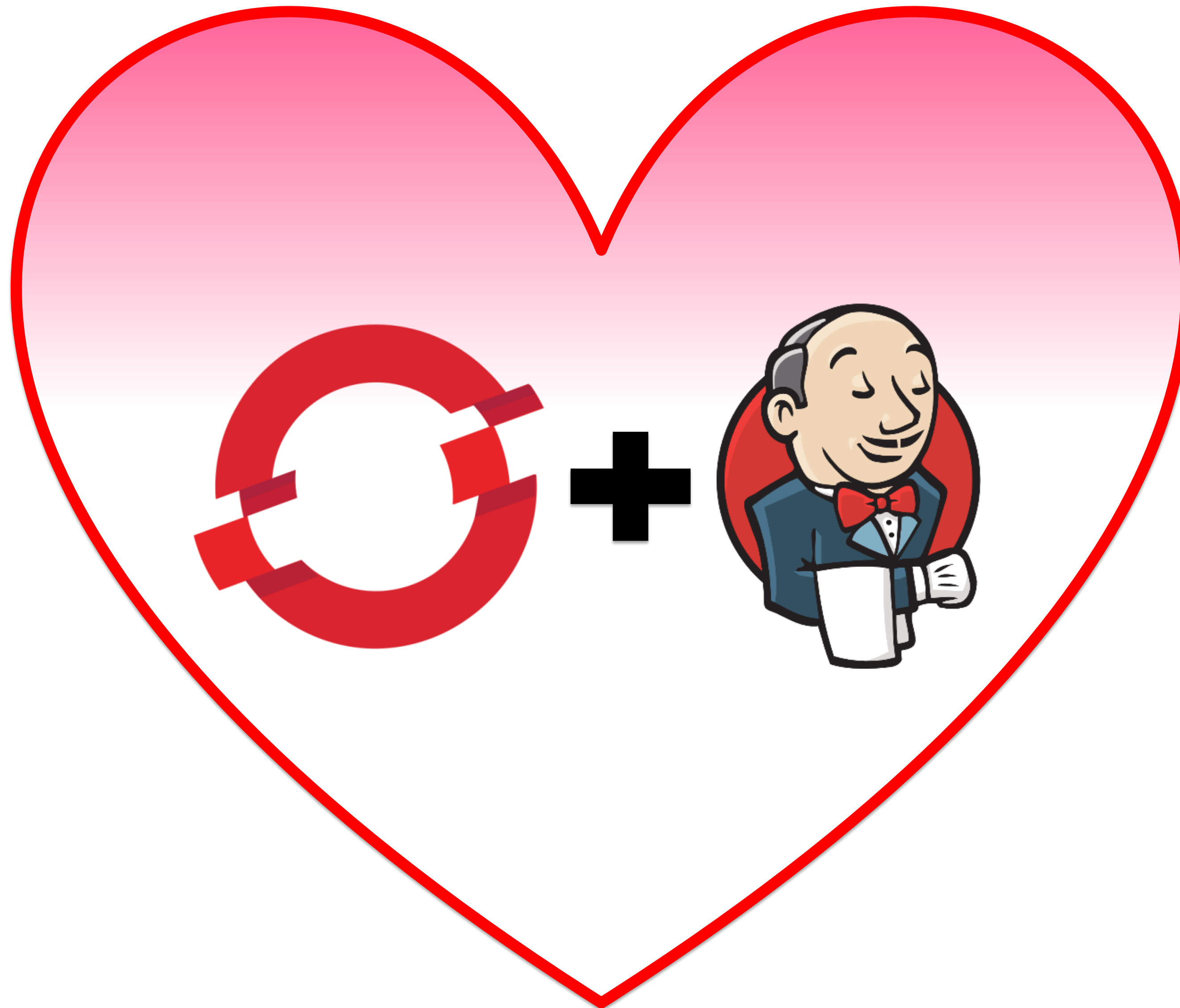


# CI/CD on container platforms – CI/CD pipelines

- Modern CI/CD life cycles have to be enabled by software
  - Configuration as code
- Various software solutions on the market
  - Travis CI
  - GitLab CI
  - OpenShift with Jenkins
  - etc.
- Provide a configurable platform to specify build pipelines



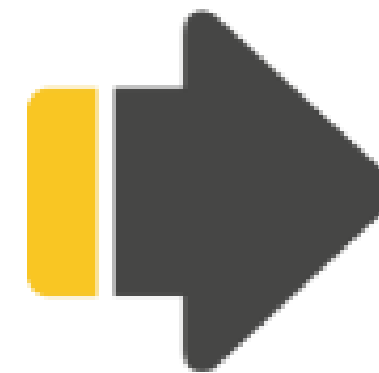
# CI/CD on container platforms – CI/CD pipelines





# CI/CD on container platforms – CI/CD pipelines

- Define your individual pipeline
- Bind it to your source control
- Deploy the pipeline to your Jenkins
- Get some coffee



# CI/CD on container platforms – Build configs

- Build configs specify how to build software
- YAML or JSON format
- Different strategies for various use cases
  - jenkinsPipelineStrategy
  - sourceStrategy
  - dockerStrategy
  - customStrategy

```
apiVersion: v1
kind: BuildConfig
metadata:
  labels:
    build: ${APPLICATION_NAME}-pipeline
    component: backend
    app: ${APPLICATION_NAME}
    name: ${APPLICATION_NAME}-pipeline
spec:
  failedBuildsHistoryLimit: 5
  runPolicy: Serial
  source:
    git:
      uri: ${INFRASTRUCTURE_REPOSITORY_URL}
    sourceSecret:
      name: gitlab-ssh
    type: Git
  strategy:
    jenkinsPipelineStrategy:
      jenkinsfilePath: infra/jenkins/Jenkinsfile
    type: Source
```

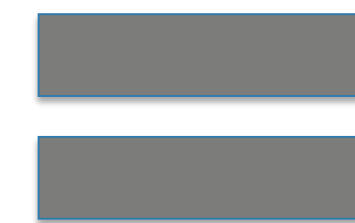
# CI/CD on container platforms – Jenkins

- We need a Jenkins file to define the pipeline
- We need a jenkinsPipelineStrategy build config
- Build a Jenkins containing the Jenkins file
- Jenkins and build configuration stored in the git repository

```
node{  
    checkout scm  
}
```



```
- apiVersion: v1  
kind: BuildConfig  
metadata:  
  labels:  
    build: ${APPLICATION_NAME}-pipeline  
    component: backend  
    app: ${APPLICATION_NAME}  
    name: ${APPLICATION_NAME}-pipeline  
spec:  
  failedBuildsHistoryLimit: 5  
  runPolicy: Serial  
  source:  
    git:  
      uri: ${INFRASTRUCTURE_REPOSITORY_URL}  
      sourceSecret:  
        name: gitlab-ssh  
      type: Git  
  strategy:  
    jenkinsPipelineStrategy:  
      jenkinsfilePath: infra/jenkins/Jenkinsfile  
    type: Source
```



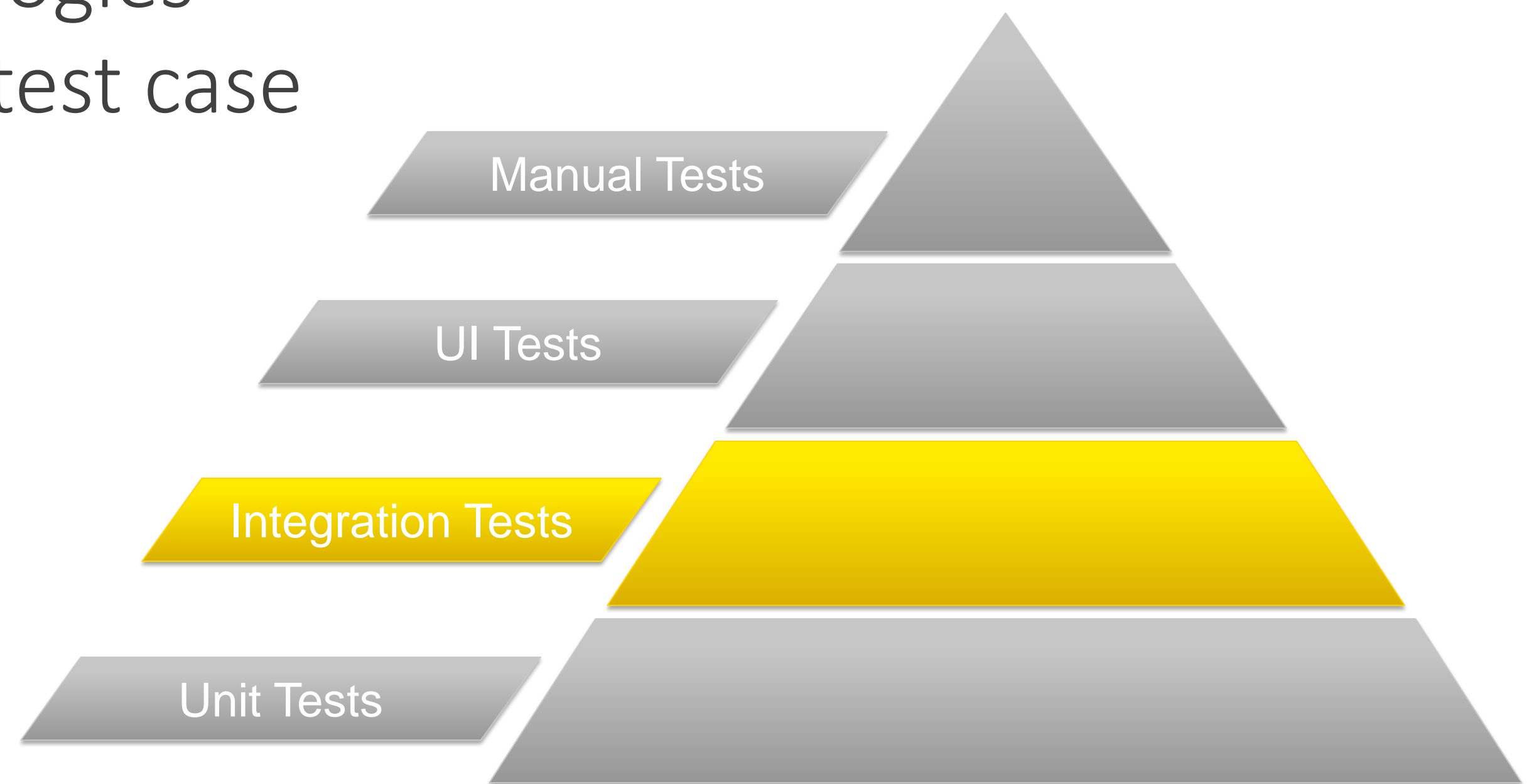
# Agenda

- Testing of microservices
- CI/CD on container platforms
- Integration testing with Citrus
- Demo
- Summary

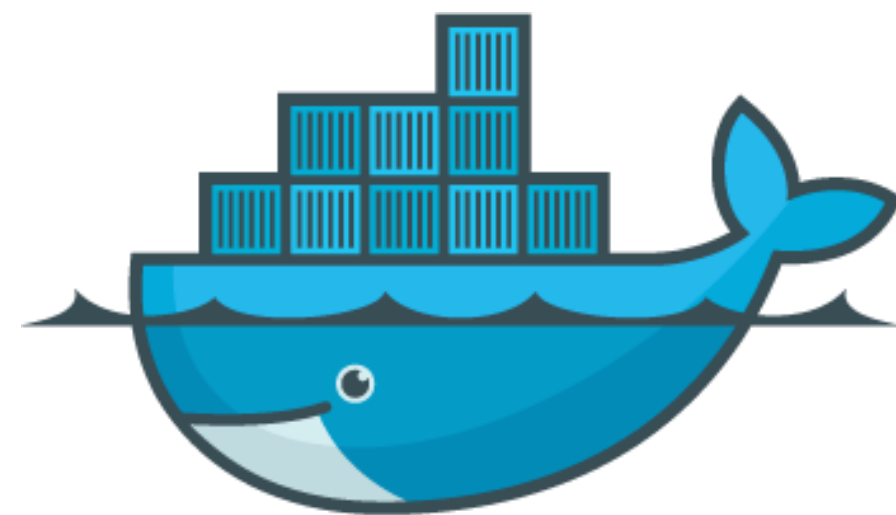


# Integration testing with CITRUS

- Open source framework
- Focus on automated integration testing
- No mocks – real messages
- Powerful message validation capabilities
- Supports a vast amount of technologies
- Transports combinable in a single test case



# Integration testing with CITRUS🍊



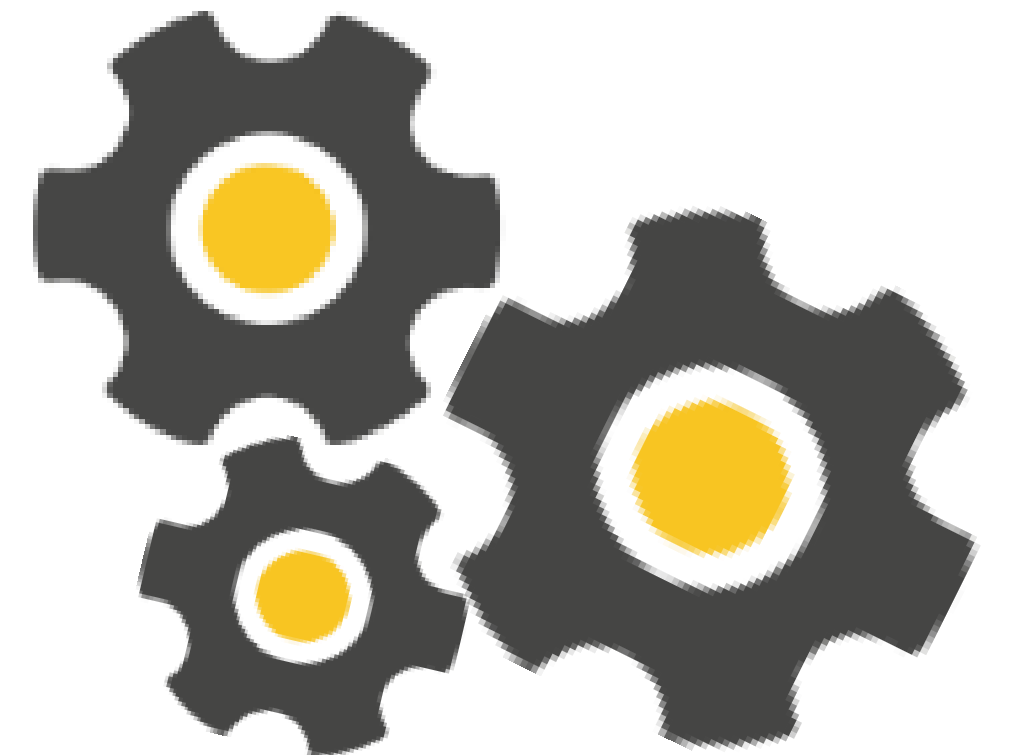


# Integration testing with CITRUS

Endpoint	Description
citrus-http	HTTP client and server
citrus-jms	JMS consumer and producer
citrus-ws	SOAP WebServices client and server
citrus-mail	Mail client and server
citrus-docker	Docker client
citrus-camel	Apache Camel components
citrus-kubernetes	Kubernetes client
citrus-selenium	Selenium browser
citrus-ftp	FTP client and server
citrus-vertx	Vert.x consumer and producer
citrus-ssh	SSH client and server
citrus-jdbc	JDBC server simulation
...	...

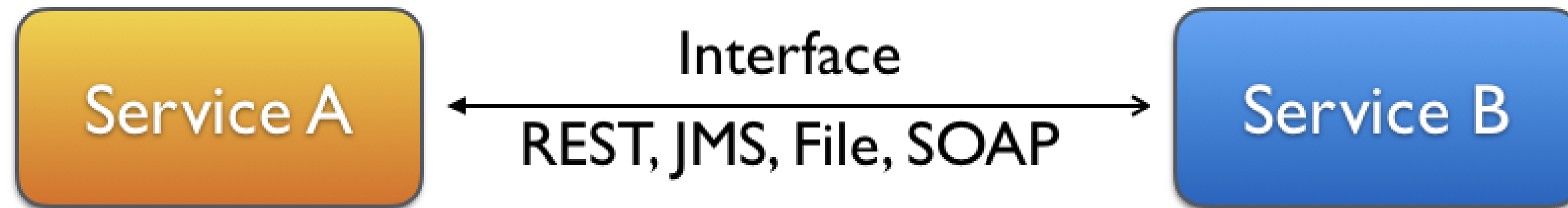
# Integration testing with CITRUS

- Rich Java- and XML-DSL to write tests
- Integrates with JUnit and TestNG
- Integrates with common build tools
  - Maven
  - Gradle
  - Ant
- Therefore tests are easily executable with Jenkins

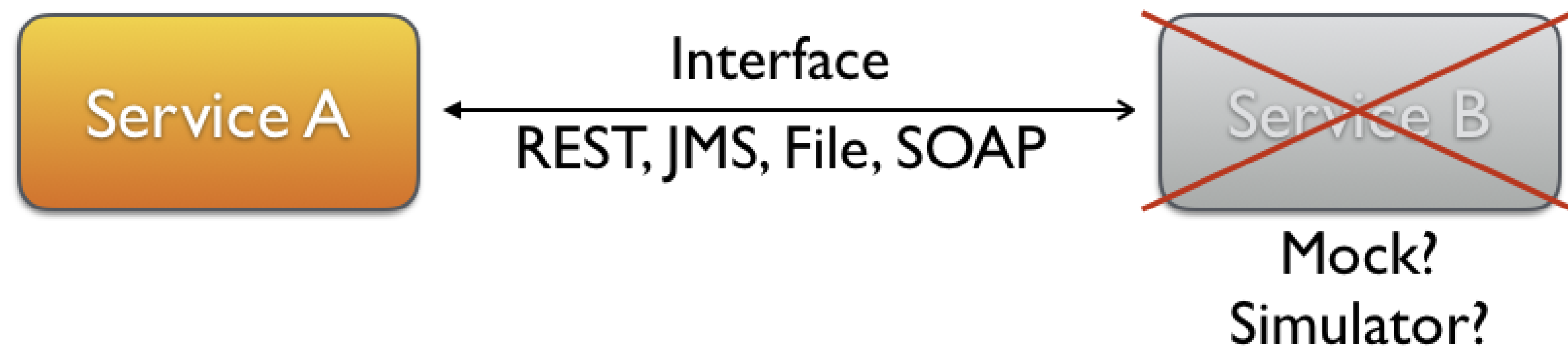




# Integration testing with CITRUS



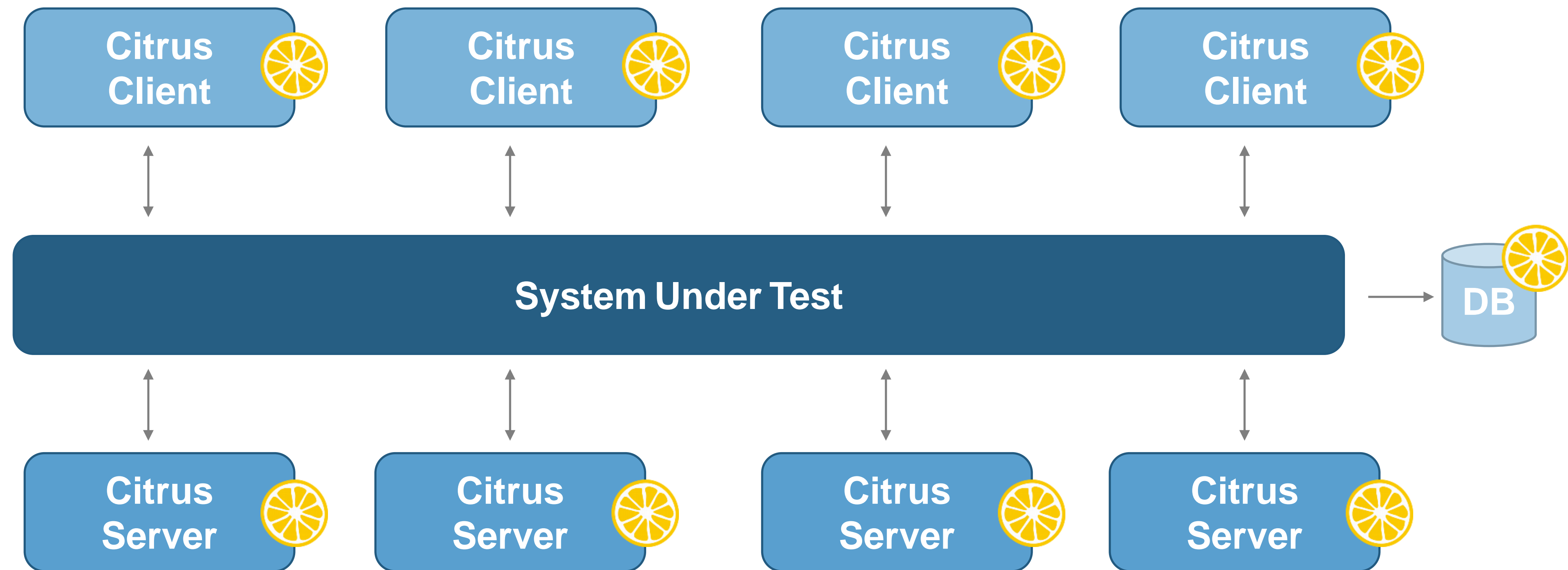
# Integration testing with CITRUS🍊



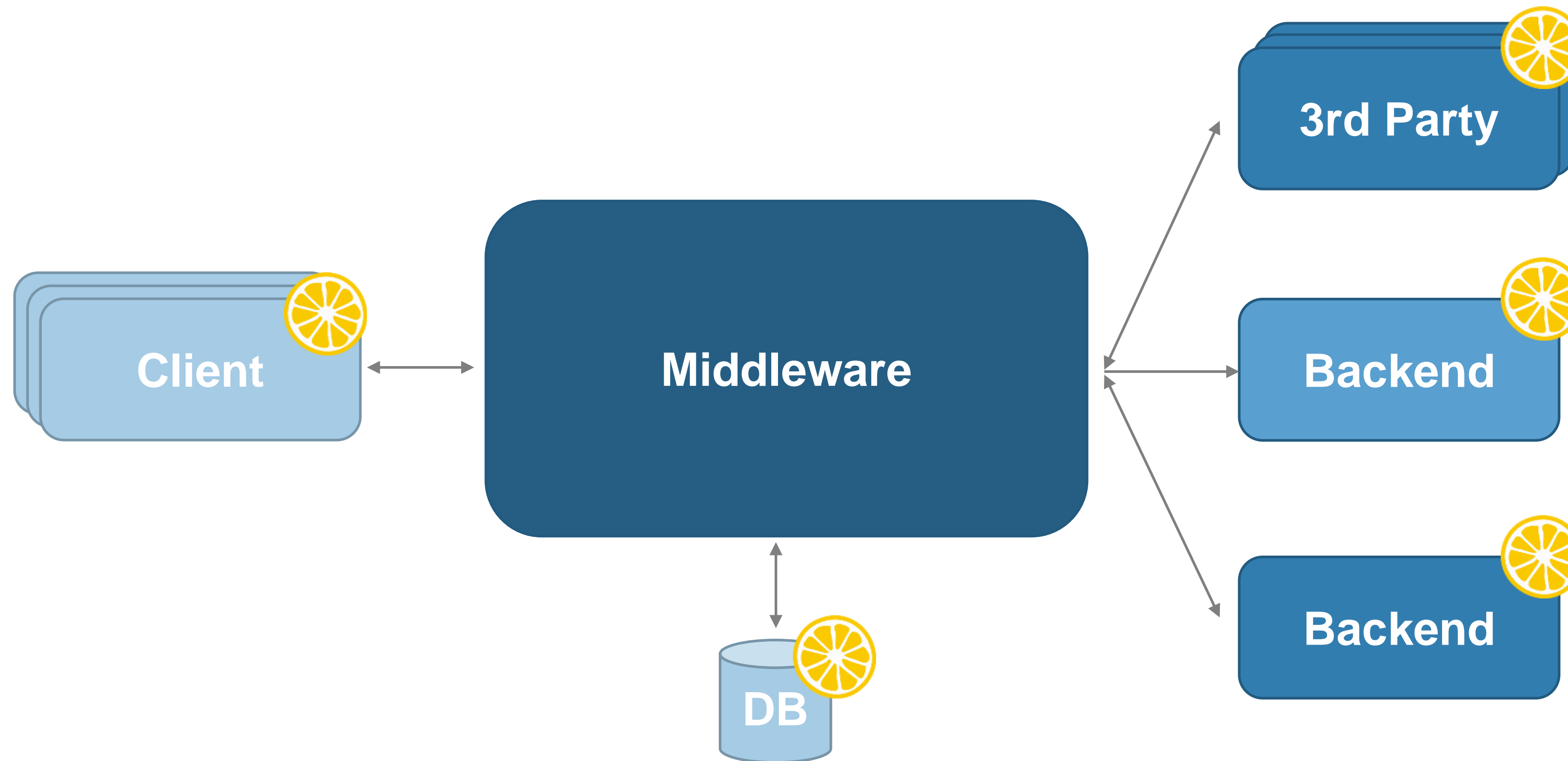
# Integration testing with CITRUS



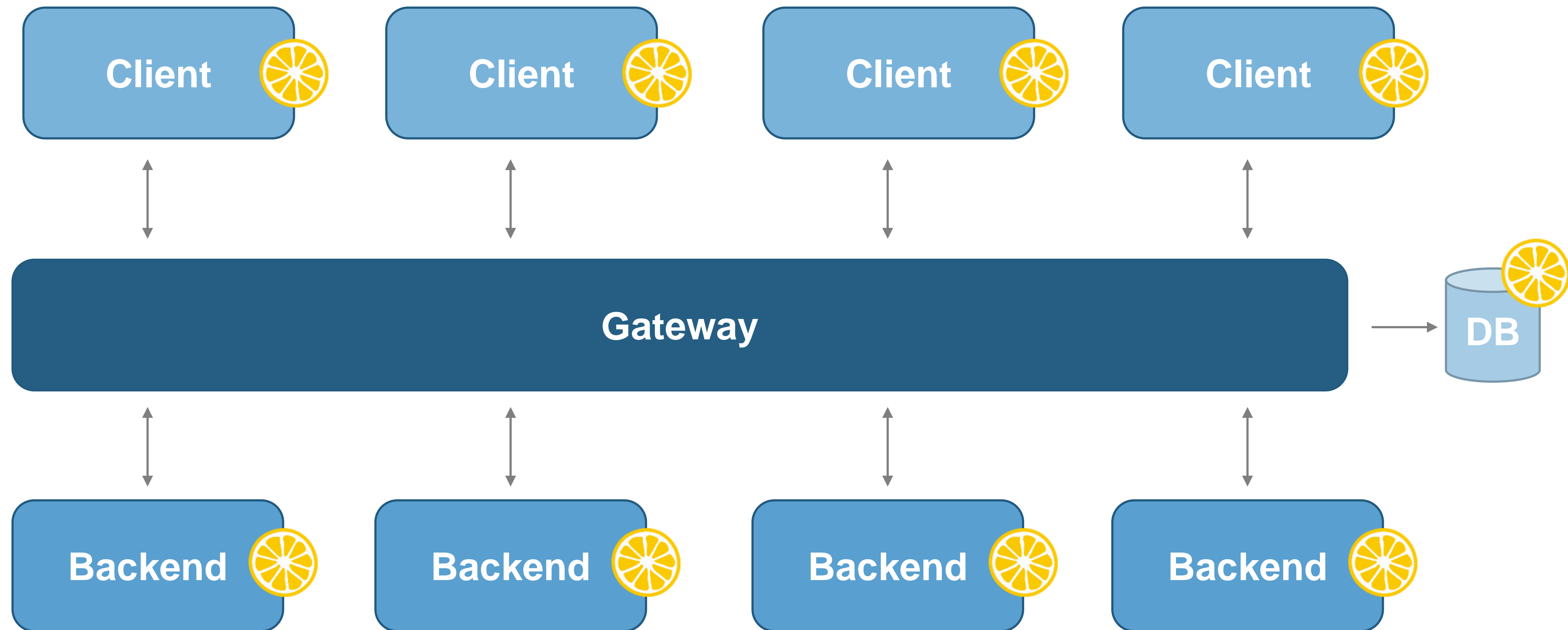
# Integration testing with CITRUS



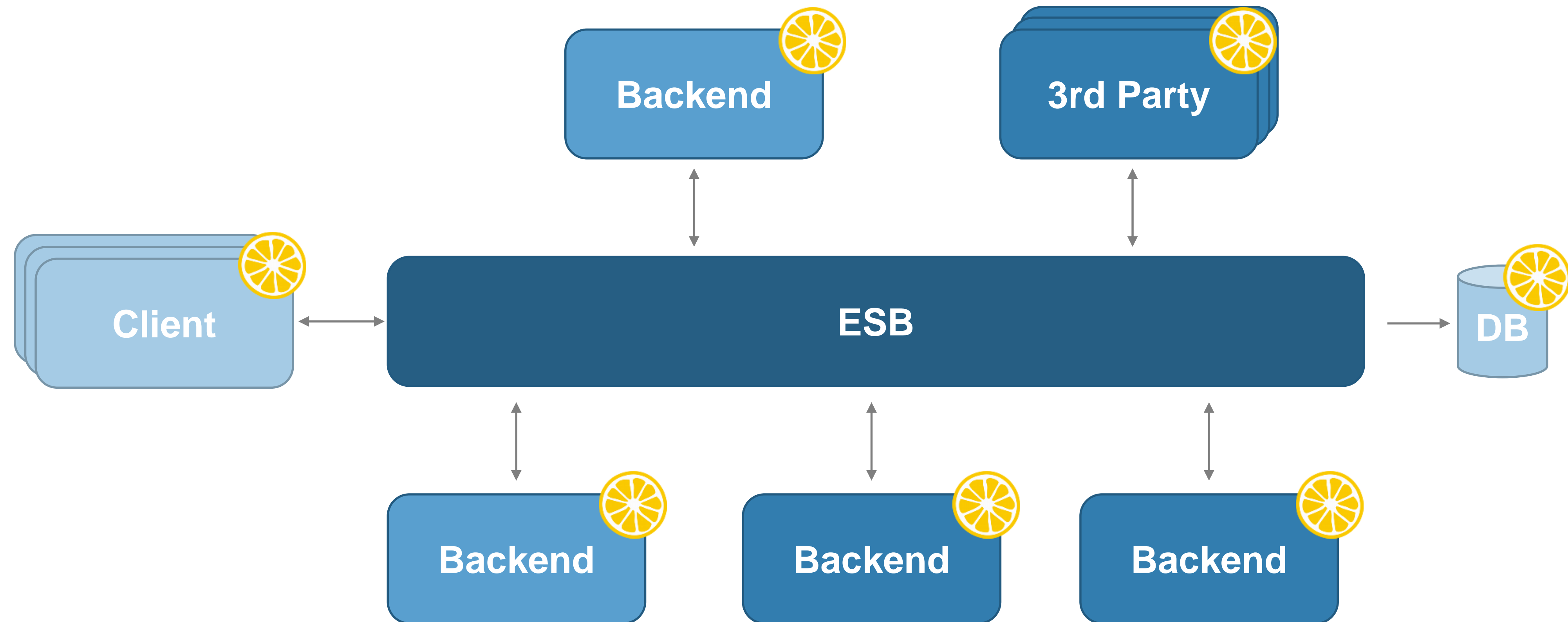
# Integration testing with CITRUS



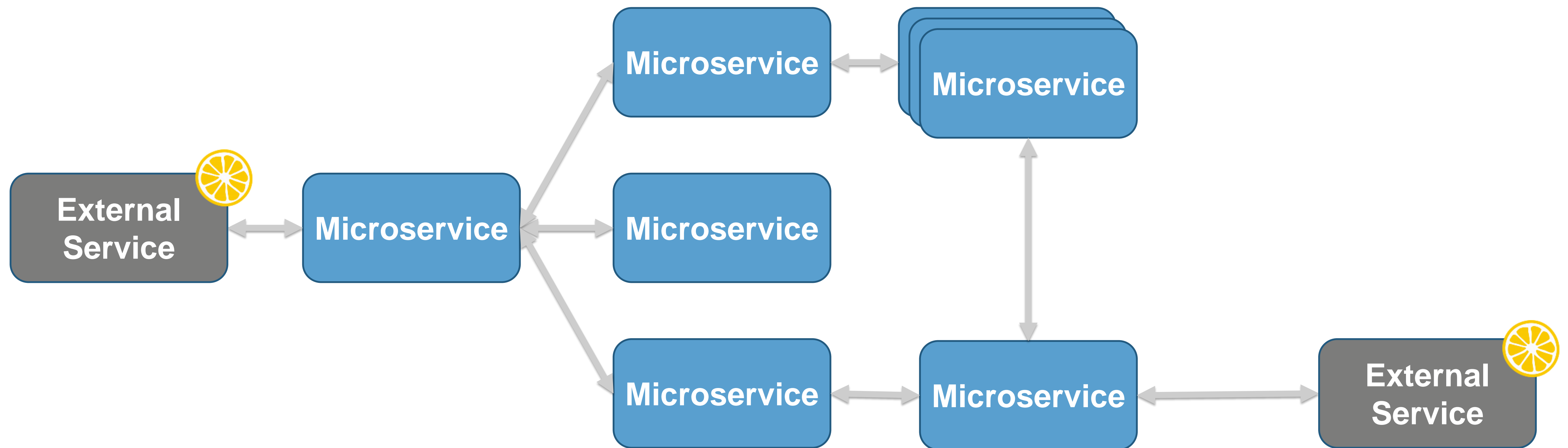
# Integration testing with CITRUS



# Integration testing with CITRUS

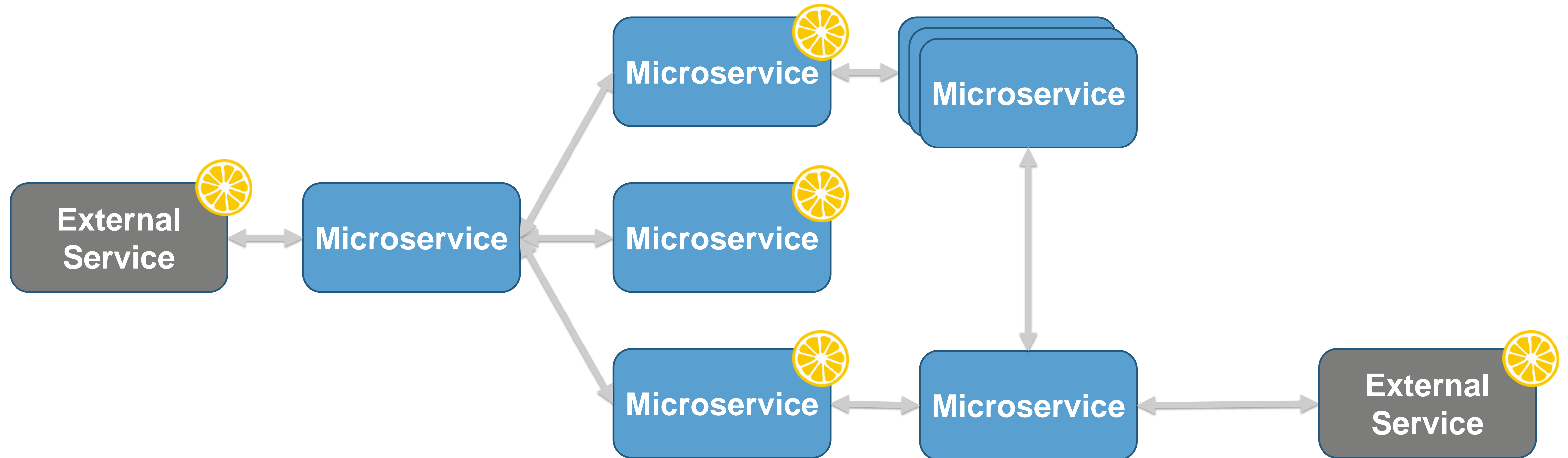


# Integration testing with CITRUS



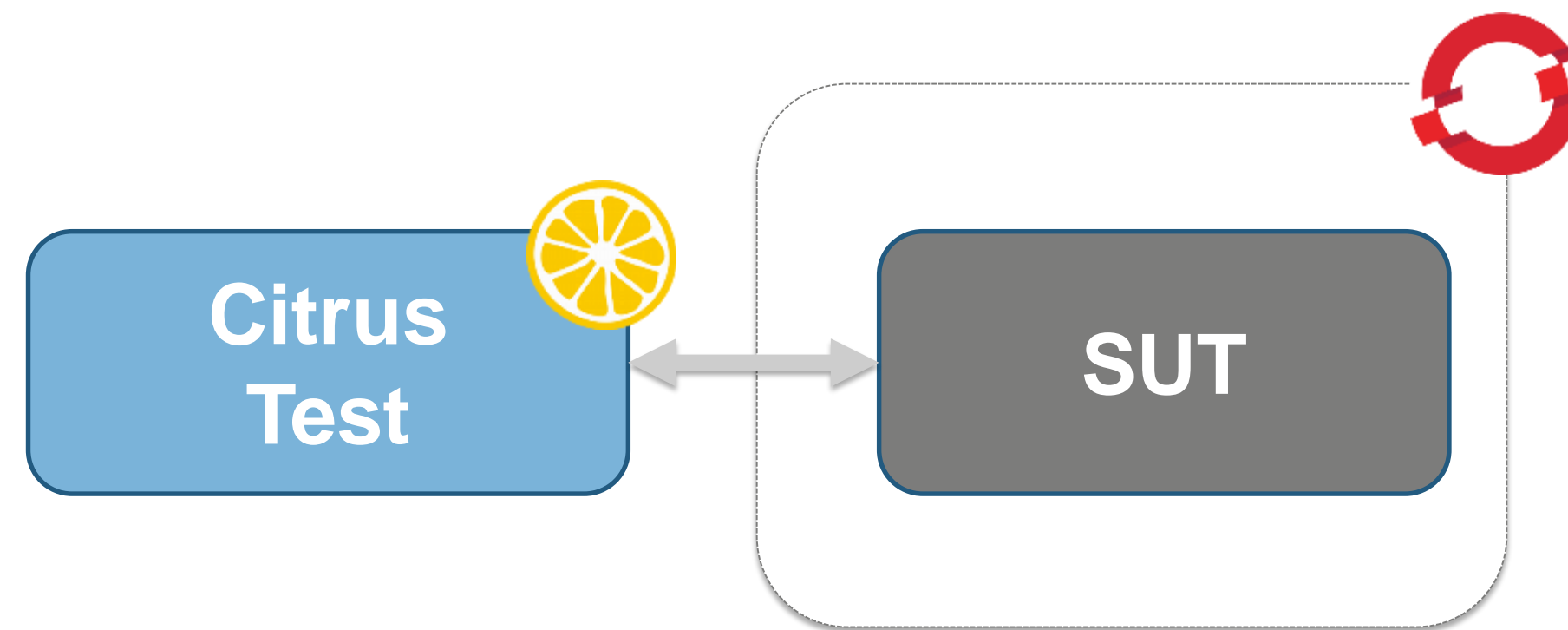


# Integration testing with CITRUS



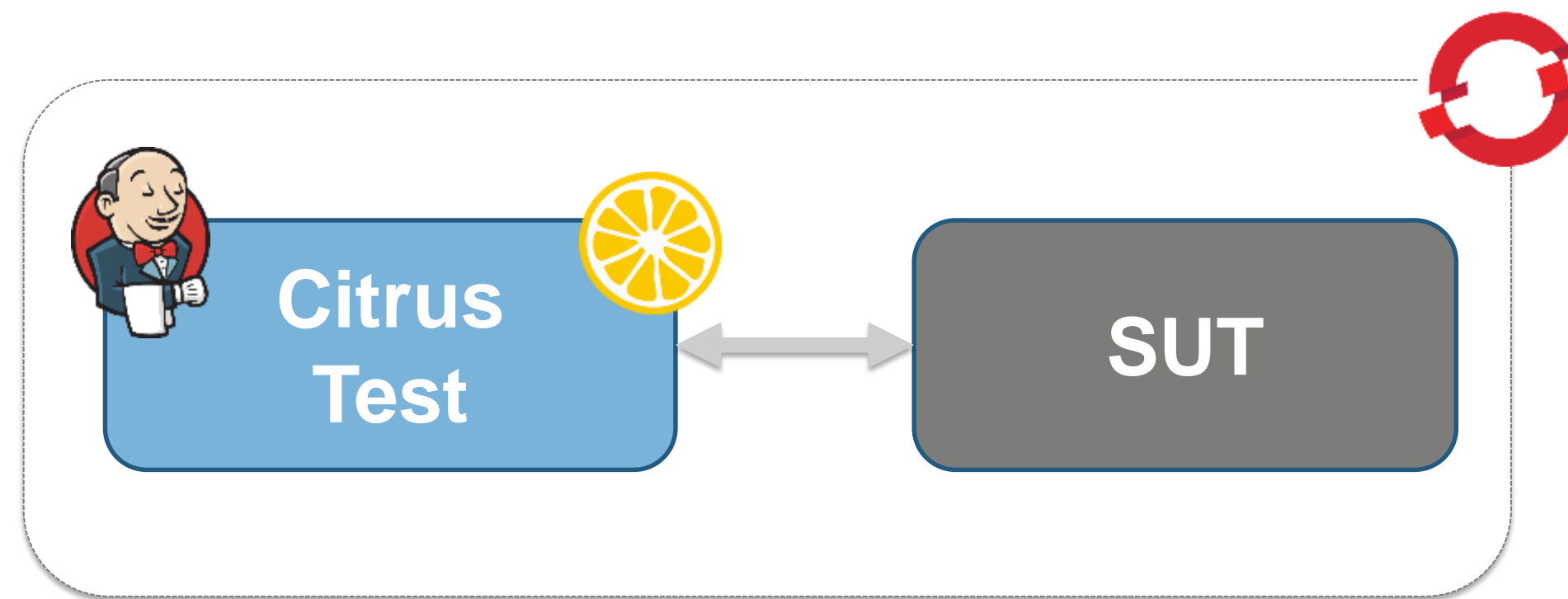
# Integration testing with CITRUS

Direct test execution



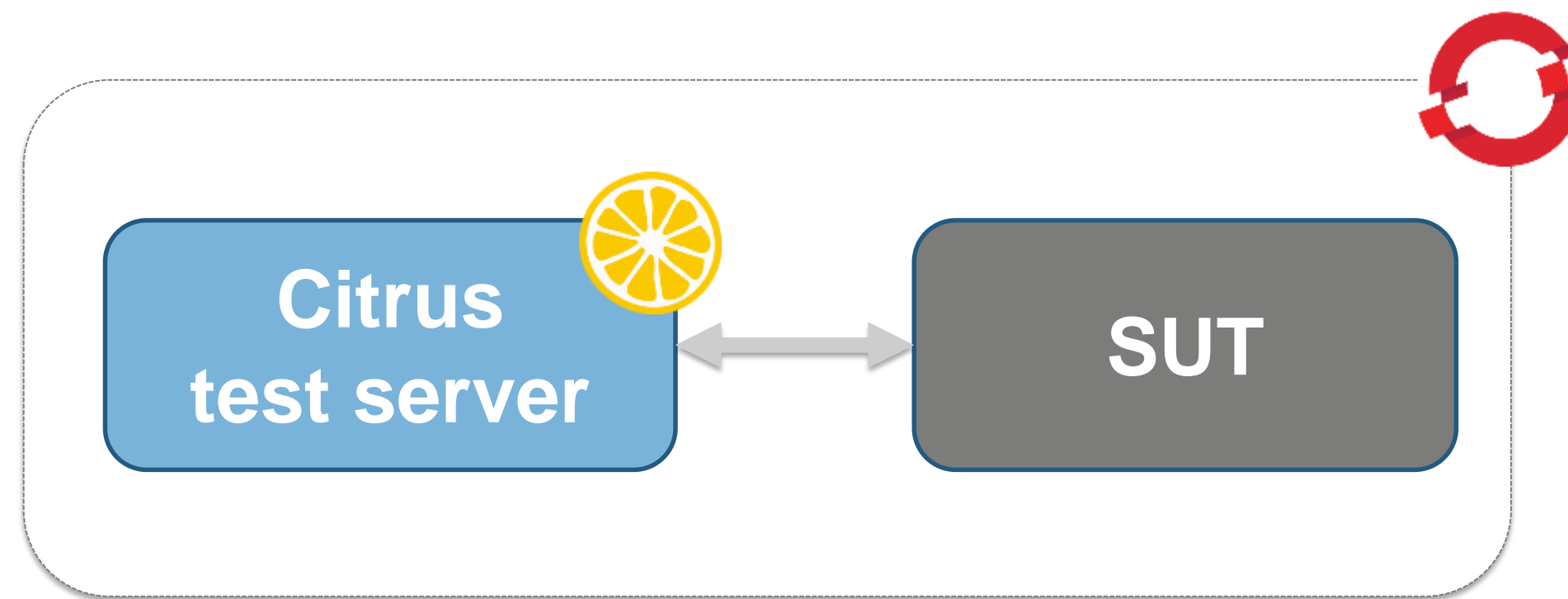
# Integration testing with CITRUS

From a Jenkins slave



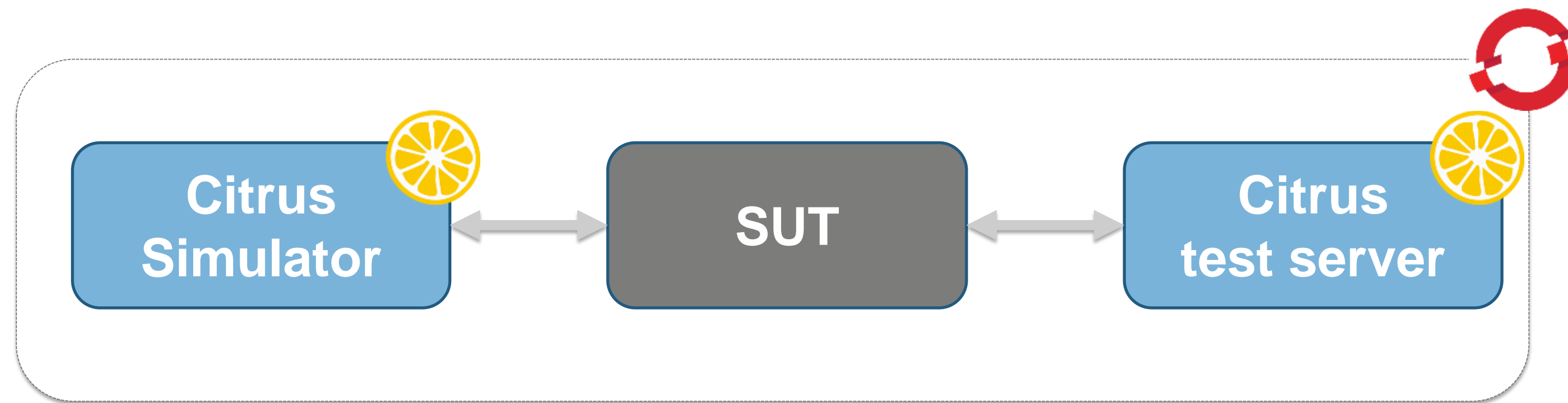
# Integration testing with CITRUS

From a dedicated test server



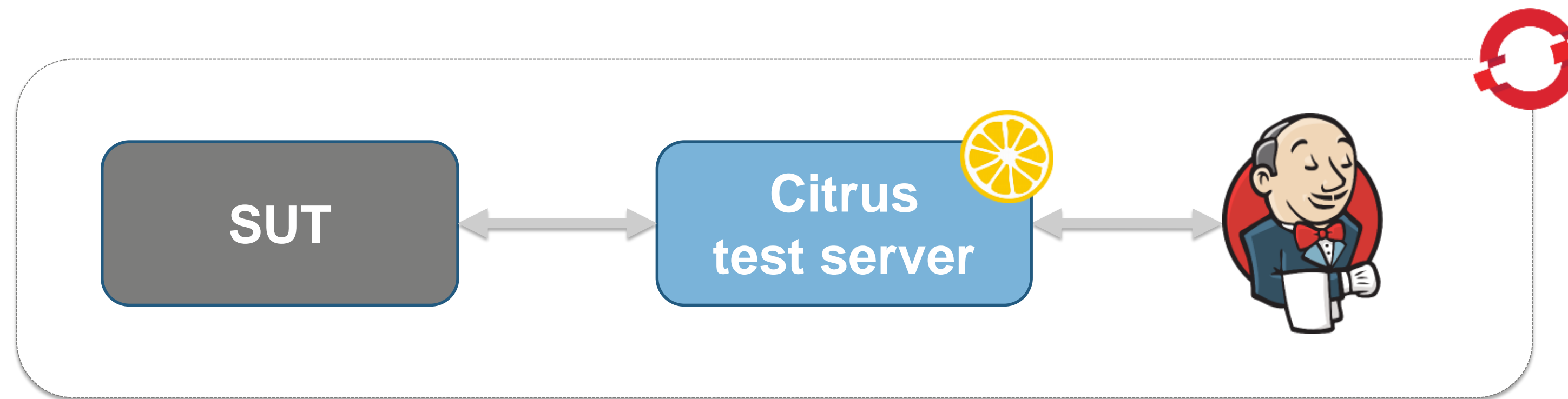
# Integration testing with CITRUS

In combination with a standalone Simulator



# Integration testing with CITRUS

With Jenkins controlling your test server



# Agenda

- Testing of microservices
- CI/CD on container platforms
- Integration testing with Citrus
- Demo
- Summary



# Agenda

- Testing of microservices
- CI/CD on container platforms
- Integration testing with Citrus
- Demo
- Summary

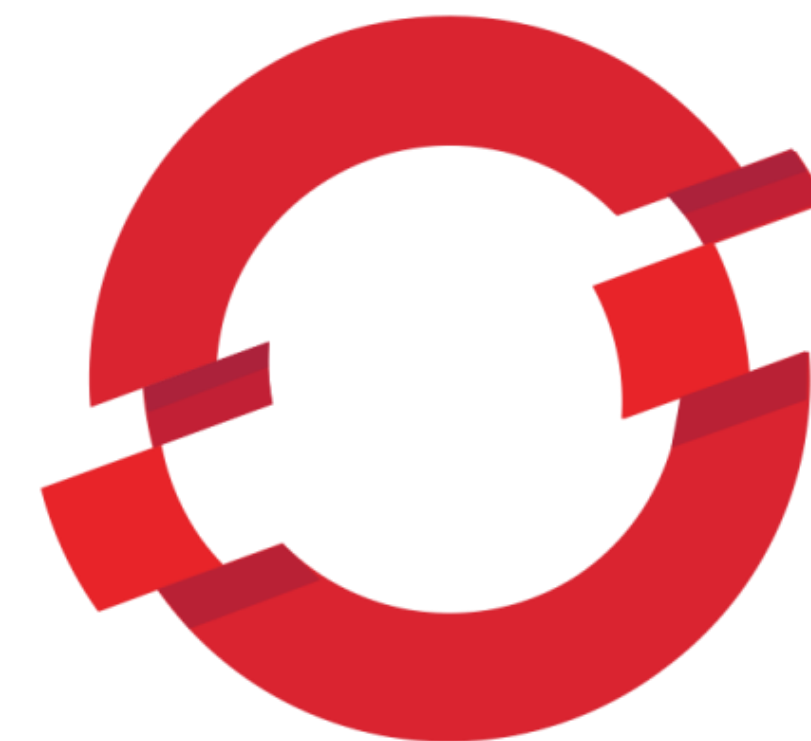




# Summary

- Testing of microservices is different
- Modern container platforms can support you
  - Simplifies CI/CD
  - Simplifies testing
- Citrus for integration testing
  - Supports a vast amount of technologies
  - Flexible test execution
  - Transports combinable in a single test case

CITRUS 









Questions?



Sven Hettwer  
Software developer

Kanzlerstraße 8  
D-40472 Düsseldorf

 @SvenHettwer  
 <https://github.com/svettwer>  
 Sven.Hettwer@consol.de  
 +49-211-339903-86



Thank you!