Implementing Contract Testing in a DevOps Pipeline

Priya Joy Kaviyil

M.Sc. in Computing in DevOps 2023



Department of Computing, ATU Donegal, Port Road, Letterkenny, Co. Donegal, Ireland.

Implementing Contract Testing in a DevOps Pipeline

Author: Priya Joy Kaviyil

Supervised by: Ruth Lennon

A thesis submitted in partial fulfilment of the requirements for the

Master of Science in Computing in DevOps

Submitted to Atlantic Technological University*Arna chur isteach chuig Ollscoil Teicneolaiochta an Atlantaigh* September 2023

# Declaration

I hereby certify that the material, which l now submit for assessment on the programmes of study leading to the award of Master of Science in Computing in DevOps, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution. I understand that it is my responsibility to ensure that I have adhered to ATU’s rules and regulations.

I hereby certify that the material on which I have relied on for the purpose of my assessment is not deemed as personal data under the GDPR Regulations. Personal data is any data from living people that can be identified. Any personal data used for the purpose of my assessment has been psudonymised and the data set and identifiers are not held by ATU. Alternatively, personal data has been anonymised in line with the Data Protection Commissioners Guidelines on Anonymisation.

I consent that my work will be held for the purposes of education assistance to future students and will be shared on the ATU Donegal (Computing) website (www.lyitcomputing.com) and Research THEA website (https://research.thea.ie/). I understand that documents once uploaded onto the website can be viewed throughout the world and not just in the Ireland. Consent can be withdrawn for the publishing of material online by emailing Thomas Dowling; Head of Department at thomas.dowling@lyit.ie to remove items from the ATU Donegal Computing website and by email emailing Denise McCaul; Systems Librarian at denise.mccaul@lyit.ie to remove items from the Research THEA website. Material will continue to appear in printed formats once published and as websites are public medium, ATU cannot guarantee that the material has not been saved or downloaded.

Signature of Candidate Date

# Acknowledgements

I would like to thank…(write only the week before submission)

# Abstract

Written here is no more than 250 words of summary of the problem, hypothesis and main conclusions. The abstract will entice people to read the rest of the document. Remember that this abstract may also be used as a descriptor for the storage of dissertation as necessary. Write only when all other sections are complete.

(no references in the abstract; strictly 250 words so that it works with the online repository)

# Acronyms

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Page |
| SSA | Some Silly Acronym | 12 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Table of Contents

[1 Declaration 4](#_Toc137560205)

[2 Acknowledgements 5](#_Toc137560206)

[3 Abstract 6](#_Toc137560207)

[4 Acronyms 7](#_Toc137560208)

[5 Table of Contents 8](#_Toc137560209)

[6 Table of Figures 10](#_Toc137560210)

[7 Table of Tables 10](#_Toc137560211)

[8 Table of Code Listings 11](#_Toc137560212)

[1. Introduction i](#_Toc137560213)

[1.1. Purpose ii](#_Toc137560214)

[1.2. Background iii](#_Toc137560215)

[1.3. Problem Statement iv](#_Toc137560216)

[1.4. Research Question v](#_Toc137560217)

[1.5. Scope and Limitations v](#_Toc137560218)

[1.6. Report Outline v](#_Toc137560219)

[2. Literature Review viii](#_Toc137560220)

[2.1. Problem Context viii](#_Toc137560221)

[2.2. Role of Microservice in DevOps viii](#_Toc137560222)

[2.3. Role of Containers in Microservice Architecture ix](#_Toc137560223)

[2.4. Testing of Microservices ix](#_Toc137560224)

[2.5. Tables x](#_Toc137560225)

[2.6. Code and Formulae xi](#_Toc137560226)

[2.7. Chapter Conclusions xii](#_Toc137560227)

[3. Design Introduction xiii](#_Toc137560228)

[3.1. Design Considerations xiii](#_Toc137560229)

[3.2. System Context Diagram xiii](#_Toc137560230)

[3.3. Specific Technology/Concept 1 (can you use one of your keywords in this title?) xiv](#_Toc137560231)

[3.3.1. Commercial Technology/narrow scope Example1 xiv](#_Toc137560232)

[3.3.2. Commercial Technology/narrow scope Example2 xiv](#_Toc137560233)

[3.4. Specific Technology/Concept 2 xv](#_Toc137560234)

[3.4.1. Commercial Technology/narrow scope Example1 xv](#_Toc137560235)

[3.4.2. Commercial Technology/narrow scope Example2 xv](#_Toc137560236)

[3.5. Framework or other relevant title xv](#_Toc137560237)

[3.6. Chapter Conclusions xv](#_Toc137560238)

[4. Implementation xvi](#_Toc137560239)

[4.1. Some Title with keyword from RQ xvi](#_Toc137560240)

[4.2. Some Title with keyword from RQ xvi](#_Toc137560241)

[4.3. Chapter Conclusions xvi](#_Toc137560242)

[5. Test Strategy, Results and Analysis xvii](#_Toc137560243)

[5.1. Test Strategy xvii](#_Toc137560244)

[5.2. Test Type 1 (Name linked to keyword in RQ/Aims) xviii](#_Toc137560245)

[5.3. Test Type 1 (Name linked to keyword in RQ/Aims) xviii](#_Toc137560246)

[5.4. Chapter Conclusions xviii](#_Toc137560247)

[6. Conclusions xix](#_Toc137560248)

[6.1. Conclusions on Theoretical Research xix](#_Toc137560249)

[6.2. Conclusions on Practical Research xix](#_Toc137560250)

[6.2.1. X Discussion (name linked to keyword 1 in RQ/Aim) xix](#_Toc137560251)

[6.2.2. X Discussion (name linked to keyword 2 in RQ/Aim) xx](#_Toc137560252)

[6.2.3. X Discussion (name linked to keyword 3 in RQ/Aim) xx](#_Toc137560253)

[6.2.4. Some other Technology Example Heading xx](#_Toc137560254)

[6.3. Limitations xx](#_Toc137560255)

[6.4. Chapter Conclusions xx](#_Toc137560256)

[9 References xxi](#_Toc137560257)

[10 Appendices xxii](#_Toc137560258)

[11 Appendix A: Code Listing Location xxiii](#_Toc137560259)

# Table of Figures

[Figure 1 Atlantic Technological University Donegal Logo 2](#_Toc101374228)

[Figure 2 ATU Donegal, Letterkenny 2](#_Toc101374229)

# Table of Tables

[Table 1. Table Formatting Guidelines 15](#_Toc429429027)

[Table 2 Second Sample Table 16](#_Toc429429028)

# Table of Code Listings

[Code Listing 1 MDBean Message Handling 16](#_Toc429428842)

# Introduction

In the current era, technology is witnessing a dual transition, shifting from monolithic architectures to microservice architectures and from traditional software development life cycle (SDLC) practices to DevOps, where the two transitions are interconnected as both DevOps and microservice deployment stresses on continuous integration (CI) and continuous deployment (CD). With this industry shift, it is important to ensure the speed of release cycle as well as a reliable deployment for which correct testing is required. The goal of fast and reliable testing pipelines is to be able to release software often and regularly (Nagel n.d.). One testing method that has emerged as a testing method specifically designed for microservices is consumer-driven contract (CDC) testing. Contract testing is a concept that allows testing communication (both synchronous and asynchronous) between the services in isolation (‘Testing Microservices - Contract Tests’ 2023). In this thesis, we study how integration of consumer-driven contract tests into the DevOps pipeline by executing them within containers, maximize the effectiveness of continuous testing (CT) in a distributed system. For this we design a DevOps pipeline, that incorporates multiple microservices and leverages containerization technology Docker for orchestration and deployment. This design integrates contract testing within the containers using prominent frameworks like Spring Cloud Contract (SCC) and PACT, ensuring that the interactions between microservices are thoroughly validated based on defined contracts.

Technology is evolving faster than ever. People depend heavily on the internet for all kinds of regular tasks, from shopping to banking and healthcare. This forces service providers to abandon monolith software development methods and adopt Agile and DevOps approaches that help them quickly adapt to changing requirements. Another trend is microservices-based architectures, where applications are built as multiple loosely coupled services. (Wickramasinghe 2023).

DevOps practices have a simple principle. It focuses on breaking down huge problems into smaller ones and handling them in order of preference as a team. Likewise, Microservices also operate through small teams and make one functional change at a time. Besides, microservices can easily adjust the scaling without impacting the resource allocations for the remaining system. This simply indicates that the blend of DevOps and microservices in the development process is the fastest way to enhance the output of your teams. (‘DevOps for Microservices - Creating Change Together’ 2021)

The blend of DevOps and microservices enable organizations to develop faster at a lesser cost. This makes them more agile and productive. Microservices bring added productivity to DevOps as they can work on the same toolset for development and operations. This marks similar terminology and processes for requirements, reliance as well as obstacles. This way Devs and Ops can easily work together. When organizations don’t supplement DevOps with microservices, it can lead to competitive disadvantage and impact them negatively.

The combination of microservices and containers is becoming increasingly popular in the context of DevOps, which entails a shift in IT culture towards agile and lean practices to enable rapid IT service delivery. This conjunction of microservices, containers, and DevOps aims to streamline application lifecycles and improve overall efficiency.

Testing interactions among distributed components is hard. One reason is that testing stubs created on the consumer side aren't tested against the producer's code. Unit testing alone doesn't answer the question of whether components work together properly. Integration testing, especially of communication between client and server, is necessary. Contract testing is a concept that allows testing communication (both synchronous and asynchronous) between the services in isolation (‘Testing Microservices - Contract Tests’ 2023). The main idea behind it is a contract. This is a place where communication rules are agreed upon and written down by the sides. The contract is a single point where the contract is defined. The parties don’t have to synchronize when the contract changes. Both producer’s and consumer’s tests are based on this contract. It ensures that if the tests pass, the services get along in production.

## Purpose

The purpose for this thesis stems from the growing adoption of microservice architecture and the need to ensure its successful integration within DevOps pipelines. With the increasing complexity of distributed systems, the speed and reliability of software releases have become critical. Consumer-driven contract tests offer a promising solution in this regard. Containers, as lightweight and isolated environments, offer an ideal solution for managing and deploying microservices. By incorporating consumer-driven contract tests into the DevOps pipeline and executing them within containers, organizations can effectively validate service contracts and identify compatibility issues early on. This approach enables teams to effectively validate service contracts, detect compatibility issues early on, and improve the overall quality and reliability of distributed software systems. This research aims to provide insights and practical guidance on integrating consumer-driven contract tests using prominent test tools Spring Cloud Contract (SCC) and PACT to DevOps pipeline leveraging containers to enhance integration testing within DevOps practices, empowering organizations to deliver robust software in a rapidly evolving technological landscape.

## Background

DevOps is a paradigm shift in the way organizations approach software development, deployment, and maintenance. DevOps shifts the whole software development lifecycle (SDLC) to a more collaborative process (Wickramasinghe 2023). A microservice architecture is an approach to software design where an application is broken down into a distributed array of loosely coupled services. Containers are a lightweight, efficient and standard way for applications to move between environments and everything needed (except for the shared operating system on the server) to run the application is packaged inside the container object: code, run time, system tools, libraries and dependencies (‘Microservices and Containers 101 - Learn all About Microservices’ 2023) . A microservices framework including microservices and containers creates a massively scalable and distributed system which enables continuous integration / continuous delivery (CI/CD) pipelines for applications and modernizing the technology stack. Meanwhile, when dealing with microservices, the testing process becomes more complex due to the crucial role of communication within a microservice architecture. Fortunately, there is a solution in the form of contract testing, which addresses the challenges specific to microservices testing. Consumer-Driven Contract testing is a way to test integrations between services, the main idea is that when an application or a service (consumer) consumes an API provided by another service (provider) a contract is formed between them (Lehvä 2019). There are two popular frameworks for the contract test — Spring Cloud Contract (SCC) and PACT that provide a way for contract definition and automate the test process (Fong 2022). A blend of these emerging technologies, including DevOps practices, microservices architecture, containerization with Docker, and contract testing using Spring Cloud Contract (SCC) and PACT frameworks, can have a significant impact on enabling fast and reliable software product releases. By leveraging these technologies together, organizations can achieve enhanced scalability and maintainability in their software development and deployment processes. This integrated approach facilitates quicker iterations, smoother collaboration between teams, efficient resource utilization, and robust validation of microservice interactions, ultimately leading to accelerated and more dependable software releases.

## Problem Statement

With the rise of microservice architecture and the increasing emphasis on continuous delivery, there is a demand for ensuring the quality and reliability of software systems. However, traditional testing approaches often struggle to keep up with the speed and complexity of microservices.

The problem statement is:

Testing among distributed components within a DevOps pipeline is hard. Unit testing alone doesn't answer the question of whether components work together properly where Integration testing, especially of communication between client and server, is necessary (‘How Contract Tests Improve the Quality of Your Distributed Systems’ 2023). This could be solved by implementing consumer-driven contract testing into the testing strategy that allows consumers to define their expectations through contracts, ensuring that service providers comply with these expectations (Selleby n.d.).

This thesis aims to address the challenge of effectively implementing contract testing in a DevOps pipeline, enabling organizations to validate service contracts, detect compatibility issues early on, and ensure the smooth and reliable deployment of microservices within a dynamic and rapidly evolving software development environment.

## Research Question

The research question considered in this research is:

How does the integration of contract testing frameworks, such as Spring Cloud Contract (SCC) and PACT, within a Container-based DevOps pipeline enhance the validation of microservice interactions and ensure compliance with defined contracts, thereby improving the quality and reliability of the overall system?

In order to answer this research question three aims were identified. They are:

1. To design and implement a DevOps pipeline that seamlessly integrates multiple microservices and leverages Docker containerization technology for efficient orchestration and deployment.
2. To explore and evaluate the integration of contract testing frameworks such as Spring Cloud Contract (SCC) and PACT within the designed DevOps pipeline, ensuring thorough validation of microservice interactions based on defined contracts
3. An aim to implement consumer- driven contract testing for microservices. To assess the effectiveness and benefits of integrating contract testing within containers in terms of enhancing the quality, speed and reliability of microservice-based systems within the DevOps pipeline.

## Scope and Limitations

Outline what is or is not covered and why. This is just a short paragraph.

## Report Outline

A review of current literature related to the subject matter was conducted and is discussed in

chapter 2, role of Microservices in DevOps, role of containerization in microservice architecture, microservice integration testing, consumer- driven contract testing and frameworks. Chapter 3 provides an overview of the testing applications… Chapter 6 closes the dissertation with a series of conclusions and suggests further work. bbtrb

In some sections of the thesis an image may be required. Any image utilised must be referred to within the main body of text. In Figure 1 the ATU Donegal logo can be seen with an appropriate caption.



Figure . Atlantic Technological University Donegal Logo

In order to provide a caption for an image, table or equation the item should be selected. The Microsoft References ribbon should be selected. From there the Insert Caption button should be selected. The label should be set to the most appropriate one. In this example the Figure label was selected. New labels can be created as necessary. Remember not to simply copy and paste from above. Instead insert the image into the file and select it. Add the caption as described.

Where an image does not have a clearly defined border, one should be added. Care should be taken to ensure that all details of images are clearly visible both when in print and when in electronic format. Careful selection of colours should be considered for this purpose. Images as shown in Figure 2 should always be referred to from the main text.



Figure . ATU Donegal, Letterkenny

All text highlighted in yellow is text that you have written and are ready for me to read.

All text highlighted in grey is text you have written as notes to yourself and is not yet ready for me to read.

All text highlighted in green is something that has been reviewed and needs to be rephrased.

All text that no longer has highlight is deemed good enough for now but will be reviewed again during the last edit.

# Literature Review

The chapters should all begin on a new page. Page numbers appear at the bottom right of each page. Page numbers appear from Chapter 1 and onwards. Appendices should be paginated using roman numerals (I, II, etc.).

* Bullets should be aligned with the text.
* Bulleted items should have one blank line above and below.
* If there are only 1 or 2 items a bulleted list is not required.

Any paragraph after the list, image or table should resume its normal position for the given header. When using any acronym such as Some Silly Acronym (SSA) it must be expanded on its first occurrence within the text. All acronyms should appear in an acronyms list preceding the main chapters.

## Problem Context

Expand on the problem to be solved via showing the area of interest. Referenced material from IEEE database or ACM. Use google scholar, or the databases in LYIT Library as primary sources. Databases can be accessed from https://library.lyit.ie/databases/ with full journal access available simply by logging in with your LYIT log in credentials.

## Role of Microservice in DevOps

Begin with the wider area of interest and quickly move to the narrow field of study. Do not go back to the beginning of the Internet or the first attempts to create a database management system. Consider your audience. Talk about the area rather than individual tools. Provide an analysis of the broad area and focus down to the narrow area under research. When writing chapter 3 there should be a clear link from the broad topic area in chapter 2 to the specific tools selected in chapter 3.

The text of any given chapter may refer to an interesting idea presented in another book, paper, journal or whitepaper. On-line sources should not normally constitute more than 50% of your references. ALL of your references must be peer-reviewed or small percentage of whitepapers. Further details on this will be given in the Research Workshop and may be obtained from your supervisor. This is supported by research (Bloggs, 2022) carried out how best to reference. The idea is succinctly expressed by Murphy:

“A reference in a thesis should be of the previously demonstrated Harvard Style.” (Murphy, 2021a)

Notice that the quote has indentations on both sides and is surrounded by quotes. If the quote abstracts only part of a sentence double dots should be placed before or after to show where there is missing text. Further, where additions for clarification are used in the text square brackets should be used. According to Murphy:

“A reference in a thesis should be of… Harvard Style.” (Murphy, 2021b)

Quotes should be short. Long quotes are not considered acceptable. Neither should there be too many quotes. It is better to provide a critique of what the person has stated.

While the example above is provided for demonstration purposes it is obviously not a good idea to provide the same quote twice so for the purposes of this example, we will assume this quote was taken from a different book by the same author. Further the general use of a single or a small number of sources multiple times is referred to as ‘over-reliance on a source’ and is deemed plagiarism.

## Role of Containers in Microservice Architecture

## Testing of Microservices

Always begin a sentence or two before moving to subheadings.

## Tables

The text of any chapter may include tabular data. To aid legibility some simple guidelines should be adhered to. Refer to Table 2.1.

Table .. Table Formatting Guidelines

|  |  |
| --- | --- |
| Format | Description |
| Size | The table should be able to fit into one page and should not overrun. |
| Margins | The table should not extend past the normal margins of the page |
| Colour | Colour may be used but consideration should be given to both on screen display and printed display. |
| Design | Simple designs are best. At all times consider that the information in the table is more important than the ‘flashy’ design. |

The title for an image or code must appear directly underneath and on the same page. If this is not possible then move the item within the text to ensure that the caption remains with the item. The title for a table must appear directly above the table.

Refer to Table 2.2 for the second short table sample.

Table .. Short Table Example

|  |  |
| --- | --- |
| Format | Description |
| Size | The table should be able to fit into one page and should not overrun. |

Tables should never run over from one page to another. Move or split the tables as needed. Don’t forget to refer to the table such as Table 2.2, or figure from within the main text.

## Code and Formulae

Where code requires listing within the text it should be treated as an image in that it is sectioned off with a border and has a caption directly underneath. Refer to Code Listing 1.1 below. The full code listing should be included in electronic format through a repository. It should not be included in printed format in the main document. Code listings, tables and images should not run over multiple pages.

…

name: CI

# Run this workflow every time a commit gets pushed to main or a pull request gets opened against main

on:

push:

branches:

- main

pull\_request:

branches:

- main

jobs:

build:

name: Call Azure Pipeline

runs-on: ubuntu-latest

steps:

- name: Azure Pipelines Action

uses: Azure/pipelines@v1

with:

azure-devops-project-url: https://dev.azure.com/organization/project-name

azure-pipeline-name: 'My Pipeline'

Code Listing 2.1 GitHub YAML calling Azure

Notice that only minimal commenting is provided within the text. The code is shown in Cambria, 10 point. This reduces the overall text size and clearly distinguishes it from the main text.

If a single line of formula is required it can be referred to within the text as formula (Equation 1) for example with the formula example shown slightly indented and with the formula number to the far right.

Equation 1

A single line space above and below the formula (Tsiolkovsky, 2000) also aids legibility. Note also that the font size is increased by one point. A further item to note is that equations may also be referenced.

## Chapter Conclusions

Make sure that each chapter has a one paragraph chapter conclusions. Make sure this paragraph somehow relates to the research question. Make sure that the last line in the summary introduces the next chapter by relating it to the last chapter.

# Design Introduction

The design of the system should be presented succinctly. Justification for the selection of the design elements should be considered. The titles of the chapters are only samples. However, it is important that the background to the problem, the review of existing work in the area, design, results and conclusions are all discussed in some format.

The selection of tools and techniques are often best served by providing a high-level diagram early in this chapter. This helps the reader to understand where each element of the system is applied. In particular, it is often helpful to use the broad technology terminology from chapter 2 in this high-level diagram at the beginning of the chapter. At the end of the chapter a similar diagram with the exact technologies chosen can then be presented. This is not a requirement but may be considered helpful.

If the dissertation is focused on analytics rather then technology describe the process including, data ingress, etl, analytics, etc.

## Design Considerations

Describe the frameworks, methods, laws or other considerations prior to describing the system.

## System Context Diagram

Describe the solution very briefly by providing a simplified graphical image of the system. This is a high level diagram used to set the context for the technologies you will talk about. This should not be a UML diagram but something that can be easily understood.

In a cyberpsychology example a method may be shown as a flow chart.

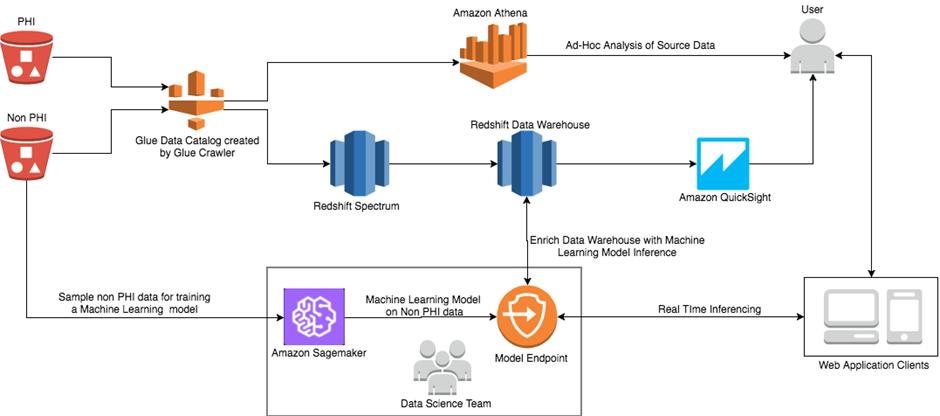


Figure . Top Level Context/System Diagram

## Specific Technology/Concept 1 (can you use one of your keywords in this title?)

Describe commercial implementations available for the Broad Technology in Chapter 2. This could also be a concept you wish to investigate. Mention the options. Mention how you evaluated them. What was your final choice and why. For example, when speaking of technology this time talk about Docker as an implementation of containers. Do this for each of the technologies, algorithms or concepts selected.

### Commercial Technology/narrow scope Example1

Why is this a good example? How do you evaluate the best? Why did you or did you not pick this technology or concept. How is the scope narrowed by using this example.

### Commercial Technology/narrow scope Example2

Why is this a good example? How do you evaluate the best? Why did you or did you not pick this technology or concept. How is the scope narrowed by using this example.

## Specific Technology/Concept 2

Describe commercial implementations available for the Broad Technology in Chapter 2. This could also be a concept you wish to investigate. Mention the options. Mention how you evaluated them. What was your final choice and why. For example, when speaking of technology this time talk about Docker as an implementation of containers. Do this for each of the technologies, algorithms or concepts selected.

### Commercial Technology/narrow scope Example1

Why is this a good example? How do you evaluate the best? Why did you or did you not pick this technology or concept. How is the scope narrowed by using this example.

### Commercial Technology/narrow scope Example2

Why is this a good example? How do you evaluate the best? Why did you or did you not pick this technology or concept. How is the scope narrowed by using this example.

## Framework or other relevant title

Make sure that the broad areas include the key words mentioned in the research question and in the individual aims.

## Chapter Conclusions

Provide specific conclusions relating back to key words mentioned in the research questions and in the individual aims.

# Implementation

The implementation chapter may describe a case study, interviews, observations, combinations of technologies or code structure depending on the type of dissertation you wish to carry out.

The implementation should be presented succinctly. Clearly define how the system/algorithm/process works in practice showing the most important parts and how they relate to each other. Describe your specific contribution making it clear what is yours and what api’s you used from elsewhere.

Show snips of code/configuration/questionnaire or similar. This may include screengrabs of your implementation if a software or hardware artifact is involved. Relate the implementation elements to the sections described in the literature survey. Be very careful to end each second level heading with a statement showing how the section relates to the research question.

## Some Title with keyword from RQ

Some text.

## Some Title with keyword from RQ

Some text.

## Chapter Conclusions

Provide specific conclusions relating back to key words mentioned in the research questions and in the individual aims.

# Test Strategy, Results and Analysis

Test strategy and how it aligns to the problem statement and hypothesis should be provided. This can be integrated with the results chapter or can be separate. This depends on the type of dissertation. Results can be quantitative or qualitative. Discuss this using appropriate, high level terminology.

Results of tests carried out should be presented. Where extensive testing is carried out the majority of the results should be moved to the appendix. Each graph should include values on the axes and should be appropriately labelled. All graphs/tables must be referred to and explained within the main text. Refer to all graphs in the main text. See Figure 5.1

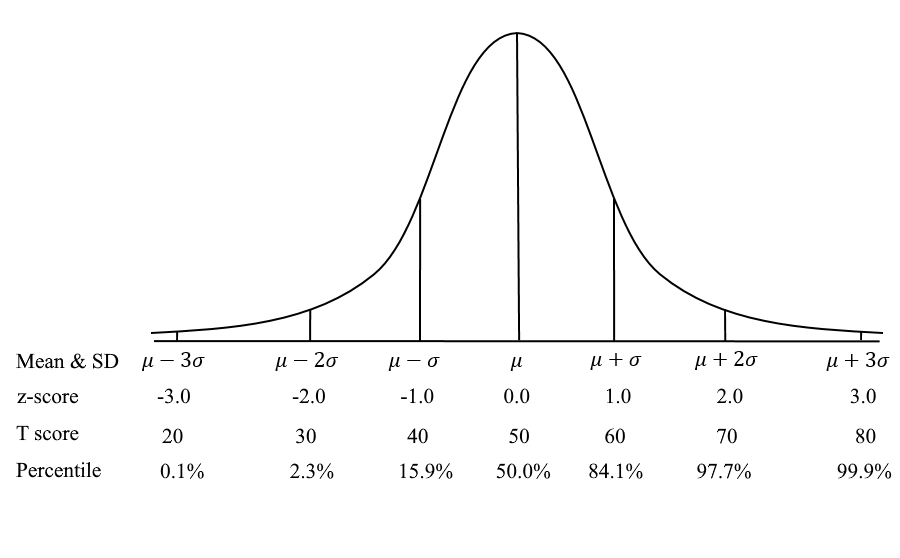


Figure . A graph showing Z score, t score and percentile information

## Test Strategy

Describe the type of tests that were selected as part of the test strategy. Why are these types of tests considered important?

Remember that these graphs cannot simply be the sum or average of figures obtained. You need to carry out detailed analysis and that takes time.

## Test Type 1 (Name linked to keyword in RQ/Aims)

Describe the type of test carried out. Say why it is an important type of test. What were the inputs and outputs. State how many times the test was run or the confidence factor for the test. State the facts drawn from the tests but do not give an opinion on the importance of the result. Repeat for each test type.

## Test Type 1 (Name linked to keyword in RQ/Aims)

Ensure that there is a dozen or so test types each with appropriate graphs or similar demonstration of practical tests. Include tables, graphs and screen shots as needed. Average is not sufficient as a mechanism to evaluate data.

## Chapter Conclusions

Describe how the test strategy included all appropriate test types as could be covered in the time allowed. Describe general observations on the results.

# Conclusions

Each week write at least one or two lines of notes to yourself as to what you might say in the conclusions chapter. You will forget when you get to the end, so it is really important to write this as you go along.

Conclusions should not introduce new material. The conclusions section should present your final thoughts in an organised manner. Try to organise your conclusions into logical paragraphs where points surround a given topic.

It could be argued that references should not appear in this chapter as that would indicate new material other than your own work. Similarly, it could be argued that images should not appear in the conclusions chapter.

Note that the number of pages in the conclusions should generally balance with the number of pages in the introduction. This is because the introduction is created to introduce the problem whilst the conclusions determine the validity of the hypothesis

## Conclusions on Theoretical Research

Try to organise your thoughts under headings as that will help you focus. Jot down your thoughts as you go along rather than waiting until the end to write the conclusions. You may find that you will forget a lot of the important conclusions if you don’t do this. The last statement on each of these topics should relate directly to the research question.

For example:

Bloggs, et al. showed that…indicating a need for research in this area…link to RQ

Smith, et al. described an experiment where…However, they did not consider…link to RQ

From the survey of existing literature has been found that a gap exists where….Link to RQ

## Conclusions on Practical Research

Focus on your practical element to test our hypothesis here. Focus strongly on why this is new.

### X Discussion (name linked to keyword 1 in RQ/Aim)

Discussion on how the theory from chapter 2 relates to the practical work in the remaining chapters. Show how your work agreed with, disagreed with or extended what was already known.

### X Discussion (name linked to keyword 2 in RQ/Aim)

Discussion on how the theory from chapter 2 relates to the practical work in the remaining chapters. Show how your work agreed with, disagreed with or extended what was already known.

### X Discussion (name linked to keyword 3 in RQ/Aim)

Discussion on how the theory from chapter 2 relates to the practical work in the remaining chapters. Show how your work agreed with, disagreed with or extended what was already known.

### Some other Technology Example Heading

Always relate the theory to the practical element when discussing conclusions.

## Limitations

Very short recap on limitations outlining any new discoveries on limitations. How would limitations be overcome if you had more time. Don’t just list the limits. Keep this very short as you don’t want the work to seem unfinished.

## Chapter Conclusions

Discussion on why this is so good there is some further work to be done. Relate your answer to why this can be applied in other domains. Relate to social/real world examples.

# References

DevOps for Microservices - Creating Change Together [online] (2021) *softwebsolutions*, available: https://www.softwebsolutions.com/resources/devops-and-microservices.html [accessed 12 Jun 2023].

Fong, G. (2022) Contract Test — Spring Cloud Contract vs PACT [online], *Medium*, available: https://blog.devgenius.io/contract-test-spring-cloud-contract-vs-pact-420450f20429 [accessed 17 May 2023].

How Contract Tests Improve the Quality of Your Distributed Systems [online] (2023) *InfoQ*, available: https://www.infoq.com/articles/contract-testing-spring-cloud-contract/ [accessed 17 May 2023].

Lehvä, J. (2019) ‘Testing Integrations with Consumer-Driven Contract Tests’, available: https://helda.helsinki.fi/handle/10138/304680 [accessed 6 Jun 2023].

Microservices and Containers 101 - Learn All About Microservices [online] (2023) *Avi Networks*, available: https://avinetworks.com/what-are-microservices-and-containers/ [accessed 13 Jun 2023].

Nagel, F. (n.d.) ‘Analysis of Consumer-driven contract tests with asynchronous communication between microservices’.

Selleby, F. (n.d.) ‘Creating a Framework for Consumer-Driven Contract Testing of Java APIs’.

Testing Microservices - Contract Tests [online] (2023) *SoftwareMill*, available: https://softwaremill.com/testing-microservices-contract-tests/ [accessed 13 Jun 2023].

Wickramasinghe, S. (2023) The Role of Microservices in DevOps [online], *BMC Blogs*, available: https://www.bmc.com/blogs/devops-microservices/ [accessed 12 Jun 2023].

# Appendices

# Appendix A: Code Listing Location

Some description or essential code snips can go here.

Code for this dissertation can be found in the repository listed below. To ensure that the code is accessible the repo must be a public repo.

<https://github.com/studentlnumber/reponame.git>