Research Project Exhibition 2024

M.Sc. in Software Solutions Architecture, and the M.Sc. in DevOps,

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Technology Ireland ICT Skillnet

FOREWORD

Today's Research Project Exhibition is a day presenting great achievement by our taught Master's students in our M.Sc. in Software Solutions Architecture, and the M.Sc. in DevOps, Programmes. The students now presenting their research projects started their Masters programme journeys in January 2022. While it seems like only yesterday, yet so much has happened since then.

Those graduating today, started as students of the Department of Computing which was then part of the School of Science and Computing. You will graduate later this year as some of the first graduates to complete a programme in the new School of Enterprise Computing and Digital Transformation. This new school is part of a new Faculty of Computing, Digital and Data and represents a major investment by TU Dublin which sees Computing, Data Science and Digital Transformation as major activity areas for now and the future

As a School of Enterprise Computing and Digital Transformation we are using this opportunity to say that we are very proud to have you as graduates. Your dissertation work, which is at the cutting edge of Technology and Computing as it is practice in industry, represents some of the best work produced by students in this School. As a new School we are, engaged on an extensive consultation process around our mission, vision and activities and we welcome any comments, feedback and input on how our activities as they are anow and as they should or could be.

These programmes are being taught here in TU Dublin but they are collaborative programmes where we have joined with industry in defining and scoping these programmes. Industry needs in these programmes were refined and proposed by Technology Ireland Skillnet and their ICT employer partners identified the market need for these programmes, TU Dublin, Computing responded, co-designing an industry list of requirements into an academic Masters programme.

For the Software Architecture programme we are proud to partner with the International Association of Software Architects (IASA) and their Irish partners the Irish Computer Society (ICS) in offering this programme. It should be noted that this programme is recognised by the IASA for membership of the association.

Today we have a new cohort starting their journey as well as a cohort moving to the final part of their journey. They have plenty to learn but they too will soon reach this point. Creating and passing on knowledge is the duty of a University and doing this in fast moving Technological Fields relevant to industry is the particular mission of a Technological University.

We hope you enjoy the project symposium.

Finbarr Feeney PhD / Head of School
School of Enterprise Computing and Digital Transhformation
OT Bhaile Átha Cliath / TU Dublin - Tallaght Campus
D24 FKT9
Ireland

M.Sc. in Software Solutions Architecture, and the M.Sc. in DevOps

RESEARCH PROJECT SYMPOSIUM AGENDA

19th January 2024

14:00 - 14:15 Opening Addresses

Gary Clynch, School of Enterprise Computing and Digital Transformation, TU Dublin

Dr. John Burns, School of Enterprise Computing and Digital Transformation, TU Dublin

Róisín Faherty, Head of Discipline, School of Enterprise Computing and Digital Transformation, TU Dublin

Sean McHugh, Head of Discipline, School of Enterprise Computing and Digital Transformation, TU Dublin

Dr. Barry Feeney, Head of School of Enterprise Computing and Digital Transformation, TU Dublin

14:15 - 15:00 Speaker - DevOps

Ann-Marie Sexton, Senior Delivery Manager, Presidio

15:00 - 15:45 Speaker - Software Solutions Architecture

Sean Wallace

15:45 - 17:30 Poster Presentations

Technology Ireland ICT Skillnet

TU DUBLIN
OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

Online MSc in DevOps

With most technology organisations moving their delivery platforms to a DevOps approach the shortage of people with cross sectional skills in DevOps is now acute. Developed by industry as a direct response to this need this first-ever Master's degree in DevOps aims to fill these important talent gaps and give credit, recognition and credibility to technologists working in this field.

The advantages of Development Teams and Operations Teams collaborating to improve the delivery of technology solutions has meant a rapid adoption of DevOps approaches to the Software Development Lifecycle. Closely associated with Lean and Agile concepts in enhancing the delivery of technology solutions, the DevOps approach has impacted very rapidly on the Technology industry.

Most existing DevOps 'specialists' grow or develop into their role with no formal standards or certification, and a modicum of training in the actual practice of cross functional DevOps practices. They may already be experienced, highly skilled, competent and high performers in their own field of Software Development, Computing, IT Management, or Quality Assurance but they can lack the knowledge and understanding of the other cross functional disciplines they now find themselves working with daily. Understanding not only the technical, but also the business and human factors at play during the high pressure demands of modern software delivery processes, is essential in the modern discipline of DevOps.

Award Level

There are two phases to the award. Candidates are registered for the full Masters of Science in DevOps Level 9 degree (90 credits) however candidates may opt to exit the programme on successful completion of the first three semesters with 60 credits and receive a Level 9 Postgraduate Diploma in DevOps (60 credits). Please note exit awards are at the discretion of the college and no refund of fees will be due.
The award structure will place greater emphasis on continuous assessment, practical and project work rather than on formal examinations. In fact there are only 2 modules that carry an actual exam.
The aim is that participants will gain a deep understanding of the topics and content covered, and be able to demonstrate this acquired knowledge as proven competence in tests and exercises drawn from practical "real life" DevOps scenarios.

Programme Delivery

The programme will start with a 3 day workshop which will involve all participants being physically present. This is seen as important to facilitate networking, experience sharing and group learning.

It is expected that lectures will be delivered one evening per week in term time and every 3-4 weeks there may be a requirement to hold lectures twice in that week. There will also be a requirement to attend one on-campus day at the end of each semester.

Lectures will be streamed live from TU Dublin (Tallaght Campus) i and will be available for download and offline viewing.

Semester 1: Introduction to DevOps

Human and Organisational Issues	Software Development Methodologies
• Lean and Agile movements and methods • Assess and evaluate organisational design and culture to facilitate DevOps style development, deployment and support • Develop and manage global multi-disciplinary teams including an understanding of the cultural and practical issues which arise • Be able to form, lead and develop teams • Assess competence, accountability, responsibility, norms and operational management • Collaboration, negotiation and partnering • Managing the Future - Creating a readiness for organisational change, organisational development and change management	• Technical implications of DevOps – the philosophy, the history, the SDLC, Lean, Agile Manifesto, continuous feedback and learning • Change, Source, Defect Control Systems, Examination of major industry implementations (e.g. Atlassian, VSTS) • Code Promotion • Code Synchronization • System Debugging • Software QA • Automated Testing • Software Security Vulnerability Management • Software Telemetry and Monitoring • Feedback and Learning

Semester 2: DevOps Fundamentals

Business Technology Strategy	IT Infrastructure Fundamentals for DevOps
• The Business Case for Agility and DevOps • Lean/Agile management/methods/frameworks (SAFE) • Product road maps, pipelines, backlogs, valuing new features and technical debt • Business case development and risk assessment • Creation/management of multi-annual business plans • Financial Management of Product and Technology life-cycles • Project Management and Methodologies • The end of the monolithic project • Designing for agility and value • Challenges for DevOps • Regulated Software • Impact for Customers of DevOps approach	• Automation of Infrastructure • Task and Process automation languages • Advanced System Administration • Software Security • System Hardening • Policies and implementation • Virtualisation • Containerisation • IT Network and Infrastructure Protocols • IT Network Monitoring • Continuous Deployment • Cloud Computing Concepts • Infrastructure as Code

> *Understanding not only the technical, but also the business and human factors at play during the high pressure demands of modern software delivery processes, is essential in the modern discipline of DevOps.*

Semester 3: Advanced DevOps

Advanced IT Infrastructure for DevOps	DevOps in Practice
• Architectural Design to support DevOps • The DevOps supply-chain and PLM relationship • DevOps in the Public Cloud • Comparative Analysis of Cloud Offerings • Cloud Scalability and Elasticity3 • Load Balancing • Virtualisation Automation • Provisioning and Orchestration • Software Configuration Management • Software Provisioning Management • Security in the Public Cloud • Degrading systems gracefully • Chaos Monkey • Server-less Compute in the Cloud	• The DevOps paradigm/pipeline in practice requirements • Develop Continuous Integration/Test/Deployment Release management • Monitor and Learn • Feedback and Iteration • Detailed DevOps Case Study of the technical and human experiences of typical practitioners, e.g. - Google SRE (Site Reliability Engineering) - Intercom (Customer Messaging Platform)

Semester 4: DevOps Research

Research Methods	Research Project
• Academic Writing • Qualitative and Quantitative research • Surveys • Statistics	• Applied piece of Research in DevOps area • Encompasses a Proof of Concept/Prototype • Supplements DevOps Theory knowledge *This is an opportunity for students to carry out a piece of work which is at the cutting edge of the field and explores in depth a feature or element of that field. It is perfectly feasible, and there are many examples of this, for students to carry out their research project on a piece of work of direct relevance to their company or organisation. The academic team in TU Dublin (Tallaght Campus) have deep industry experience and have supervised and developed MSc. projects which explore business values, infrastructure automation and DevOps projects with real industry relevance.*

Technology Ireland ICT Skillnet

M. Sc. Applied IT Architecture (online)

In conjunction with Irish Computer Society and accredited by International Association of Software Architects. A Technology Ireland Skillnet funded programme. This programme is 80% online with two days per semester attendance required.

M. Sc. Computing with DevOps (online)

This programme was designed in conjunction with leading ICT companies such as Microsoft, Fidelity, IBM, Ericsson who form the Technology Ireland Skillnet. This programme is 80% online with two days per semester attendance required.

Non-Standard Applicants

Note for interested applicants: Next intakes for these Skillnet programmes set for January 2019. Standard admissions requirements include a relevant bachelor's degree at honours level. It is recognised that there are experienced and skilled potential participants for the programme who may not fit the standard entry profile. A non-standard admission process is available here which can be based on prior experiential learning and/or qualifier modules. These qualifier modules can be taken from September 2019 for admission in January 2019. Contact bfeeney@it-tallaght.ie or mhendrick@it-tallaght.ie for more information.

Accreditation of Master of Science in Applied IT Architecture by IASA

Technology Ireland ICT Skillnet

The TUDublin (Tallaght Campus) M.Sc. in Applied IT Architecture is the first of its kind in the world to be developed based on the IASA Five Pillars.

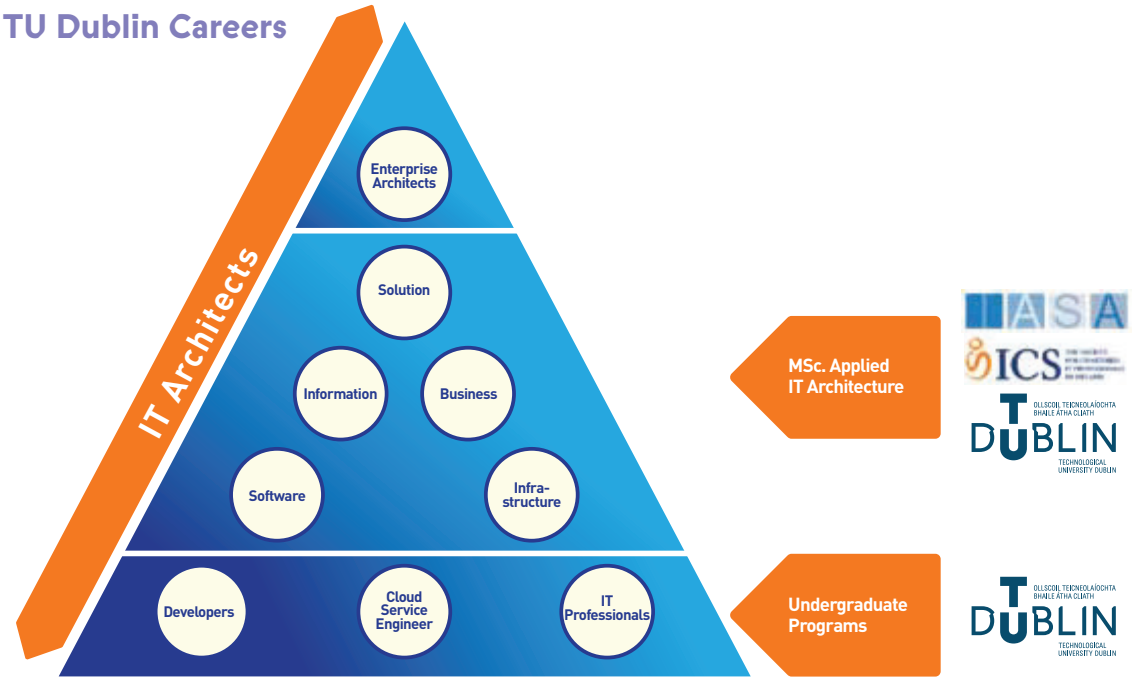
For the first time, candidates can gain a full Masters of Science degree in this specialist area through a mixed learning process with an emphasis on practical application in the workplace.

What is IT Architecture? (From IASA Global)

Architecture at IASA is the practice of business, organization or client gain through the application of technology strategy. It is the art and science of designing and delivering valuable technology strategy. At its core, the ITABoK describes how to create a professional person or group of professionals who can consistently find new applications of technology to generate positive outcomes for their client or employer. IT Architects:

- Retain depth in technical skill as well as business skill
- Able to successfully work with both business and technical staff
- Develop their own or others business cases based on technology driven innovation
- Retain the ability to deliver projects on those business cases
- Deliver business projects more successfully based on outcomes than others

TU Dublin Careers



PROJECTS

PROJECTS contd.

Azure Container Orchestration

Rachael McLoughlin
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193202@myTUDublin.ie

Introduction

Cloud providers are increasing their serverless offerings surrounding container orchestration, all aimed at focusing on accelerating the development of applications and shifting focus away from infrastructure management and complexities.
These CaaS offerings are still relatively new to the market, Azure Container Apps was only released in May 2022 and Google Cloud Run in Nov 2019.
This research project aims to give developers or software architects the necessary information to be able to make informed decisions in choosing which of the Azure container management offerings, ACA or AKS, is most suitable for a particular application and how much complexity has actually been taken away. Additionally it aims to assist in determining what types of resources would be required in a team and the level of upskilling that may be required within an existing team.

Research Questions

1. Does the introduction of additional abstraction have an impact when compared to running AKS in terms of performance ?
2. Does the additional abstraction provided by the CaaS solution, Azure Container Apps, result in a measurable reduction in complexity from the developer perspective ?

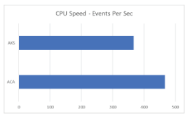
Infrastructure Specification

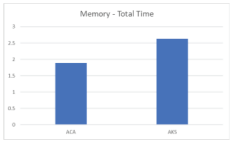
Orch. Tool	Instance	CPU	*Memory*
ACA	N/A	1	2GB
AKS	D2Ids v5	1	2GB

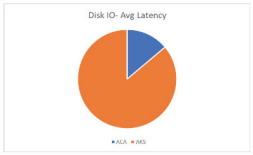
Workload Definition & Results

1. Infrastructure Performance Evaluation:

Utilize performance testing tool sysbench to ascertain variances in performance of the underlying infrastructure.**Findings:** Results were better overall on the azure container app infrastructure across each of the areas tested CPU, Memory and Disk IO

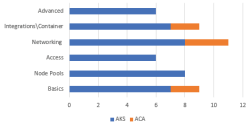






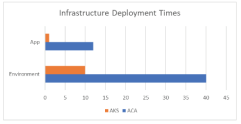
2. Ease of configuration:

Capture the perceived level of difficulty configuring the service. Taking into account the volume of options presented to the user. One of the key areas examined was the ease of scaling configuration, JMeter was utilised to send load and trigger scaling up and down of nodes. **Findings:** The learning curve is significantly reduced and ACA allows a much easier entry point to utilizing container solutions for an organization.



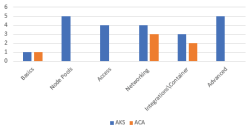
3. Deployment time:

Capture the amount of time taken to deploy the ACA and AKS Environments and applications following configuration.**Findings:** Considerable time was recorded for deployment of the Azure container Environment, which is a secure boundary around groups of container apps that share the same virtual network and logs, taking approx 40 minutes for deployment of the resource.



4. Upskill time required:

Record the amount of time spent upskilling in order to have sufficient knowledge to carry out the deployment of the service and record this against the experience level of the person carrying out the deployment.**Findings:** A knowledge of containers and container registries was sufficient to complete deployment of the application within an azure container app environment.



5. Configuration Limitations:

Determine what are the use cases where a particular offering will not meet the requirements and force the user to use a service that allows more in detail configuration. **Findings:** ACA Limitations that would render the service unsuitable are container support, limited network configuration, scale triggers. operating system and the inability to run privileged containers.

Conclusions and Future Work

Based on the researchers experience the learning curve is significantly reduced by using Azure Container Apps over Azure Kubernetes Service. This finding was based on a scoring system that was used across a range of areas to compare the differences in complexity across both services.
Utilizing benchmarking tool sysbench findings showed results were better overall on the ACA infrastructure across each of the areas tested via sysbench tool testing on CPU, Memory and Disk IO, leaving scope for future research into why this was the case.

QR Code for Recording

Knative vs Azure Functions

Ainul Habib
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00159358@myTUDublin.ie

Introduction

The concept of **Serverless computing** has generated immense attention due to the promise of facilitating a ***zero-administration*** approach to application deployment, management, scalability and improve productivity by reducing delivery time. It intends to reduce infrastructure and operational cost of Cloud Solution, by scaling application instances to Zero when traffic is ideal or zero.

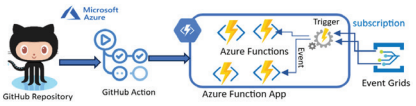
Microsoft's Azure *Functions* is a Serverless offering by Microsoft. It allows developers to deploy their Serverless application on Azure Cloud using a Pay-as-you-go model.

Knative, is an Enterprise level, Open-Source and Cloud-Native Solution framework, to build and run Serverless and Event Driven applications on a **Kubernetes Cluster**.

Proprietary Cloud Providers, like Microsoft Azure, Amazon AWS or Google Cloud, keeps the organization **Vendor locked by Design, Use and Licensing**, making them difficult to shift between cloud provider, reducing their opportunity to take advantages of new technological offering and discounts.

The study compares both **Knative** (a Cloud Native Serverless Platform) and **Azure Serverless** platforms. The study is based on several criteria, such as performance, Cold-Start, cost, adaptability, portability, development effort and maintenance.

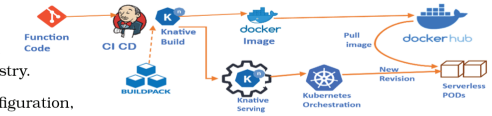
Azure Function: Offering



Microsoft Azure provides a platform to build, deploy, manage and run Serverless Application in Azure Cloud.
Developers can use Azure DevOps Pipeline, GitHub Actions or Azure CLI to build and deploy their Serverless code to an **Azure Function App**, which groups Azure Functions as a logical manageable unit.
These Serverless functions are triggered by events from various Sources like Event-Grid, Event-Hub or by HTTP incoming requests.

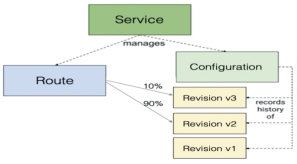
Knative: Cloud-Native Offering

Knative provides a ***Cloud-Native Platform*** over Kubernetes, offering Build, Deployment, Orchestration and Management tools to create enterprise grade, auto-scalable, event-driven **Serverless** Application. It is offered as ready-to-use components built with Kubernetes ***CRDs*** (Custom Resource Definitions) and ***Services***.

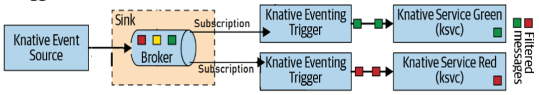


Knative Build is responsible in building "*Functional Code*" to a Docker Image, and push to a Docker Registry.

Knative Serving manages deployment, routing, configuration, revision and deployment of Serverless Function Containers as PODs



Knative Eventing is responsible for consuming events triggering the Serverless function. It consists of **Event-Source, Channel and Subscription**. Further abstraction is created by introducing **Brokers, Trigger and Filters**.



Topic Overview

- **Cloud-Event Support**: Both Azure Function and Knative follows Serverless event-driven architecture, supporting "*Cloud-Event*" API. Developers only produce a functional code, containing business logic. It is invoked by Serverless framework, passing *Cloud-Event* object as parameter. "*Cloud-Event*" contain event payload and metadata. *No infrastructure code injection is needed.*
- **Cold-Start:** All Serverless platform suffer from Cold-Start latency. It is delay in getting Serverless application scaled from **Zero** to **One**. **Application runtime** plays a major role in cold-start latency, e.g. Azure Function perform better using .Net runtime, while NodeJS produce lower start-up latency compared to Java. Application request latency may increase due to Cold-Start delay, effecting application performance. Azure Functions, under *Premium Plan*, keeps a *"Warm instance"* of Serverless, to reducing the cold-start latency. Knative also provide mitigation to cold-start latency by keeping at-least one instance alive.
- **High Scalability:** Knative and Azure Functions scale horizontally, based on high volume of events metrics. These metrics includes "**Concurrency**", "**TPS**", "**CPU**" and "**Memory**", which can be configured for Serverless applications.
- **Logging and Metrics:** Azure Functions and Knative emit logs to console-out, which must be streamed to external logging system. Knative emits many Kubernetes and custom metrics related to health and performance. Monitoring system Prometheus is deployed to capture and archive the required metrics. Application Insight is easily configured in Azure Platform to capture and present Azure Function's metrics.
- **Build and Deployment:** Azure and Knative offer CLI to package and deploy the Serverless application to the Cloud. They also allow easy integration to DevOps pipeline and GitHub action. Knative CLI packages the function code and runtime to a docker image. Azure function offers many form of packaging and deployment like Zip, Docker Image, Cloud and Git Sync.

Conclusions and Future Work

Knative is a **Cloud-Native** alternative to traditional Serverless platform offered by public cloud providers like Azure and AWS. Customers can easily migrate their Knative Solutions from current public cloud Kubernetes platform to another.
Azure Function on other hand keeps the customers "*vendor locked*", making transition to different cloud provider difficult and costly. But Azure Functions lowers the maintenance cost by fully managing the platform, servers and infrastructures.

Future Works: A study into feasibility making the running cost of Knative applications to **Zero**.

QR Code for Recording

2

Effectiveness of Generative AI on the Development of Graphic Software Artifacts

Ellen Mezera
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X12345678@myTUDublin.ie

Introduction

The surge in Artificial Intelligence research, particularly in Generative AI following the release of models like GPT-3 in 2022, has prompted extensive exploration of its potential for enhanced productivity in various industries, especially in coding tasks. As AI solutions mature, software engineering practices are evolving, with a notable adoption of Generative AI for coding, while the application of these technologies to tasks involving image generation is still in its early stages. This research sets out a methodology leveraging Large Language Models to generate graphic software artifacts. More specifically it applied GPT-4's text-to-text capabilities in generating UML Diagrams supported by Mermaid as a rendering tool. The resulting diagrams were validated by Subject Matter Experts to assess results produced by GPT-4.

Research Question 1

RQ-1 *How can OpenAI's GPT-4 model be leveraged to generate software architecture diagrams?*

This research sets out a methodology on how ChatGPT-4 can be used for software diagrams generation using its text-to-text capabilities, while text-to-image are not yet mature. Moreover, it applies this methodology on the generation of UML Class and Sequence diagrams.

The methodology is applied in an experiment where Chat-GPT is asked to generate diagrams for three use case scenarios around a Online Shopping System.

Research Question 2

RQ-2 *What is the efficacy of OpenAI's GPT-4 model generating software architecture diagrams based on the evaluation from Subject Matter Experts?*

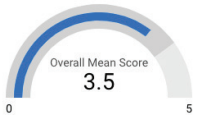
In order to validate the results obtained from the experiment created to address *RQ-1*, Subject Matter Experts assessed the results via a validation questionnaire, scoring 5 categories: Accuracy, Clarity, Completeness, Technical correctness and Usefulness.

Results

Class and Sequence diagrams were generated by GPT-4 for three use case scenarios around a Online Shopping platform in ascending order of complexity. The six diagrams were scored by Subject Matter Experts via a in 5-point Lickert scale questionnaire.

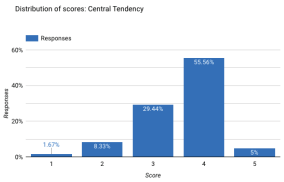
1. Effectiveness of ChatGPT in UML Diagram Generation:

The overall Mean Score of 3.54 out of 5 indicates a moderate level of effectiveness of ChatGPT in generating UML diagrams through the methodology applied in this experiment.



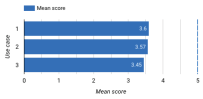
2. Distribution of scores:

The majority of scores given by evaluators through the questionnaire across all diagrams are concentrated between 3 and 4, with 4 having its majority share. The central tendency of scores indicates that evaluators found the diagrams to be above average in terms of the categories accessed.

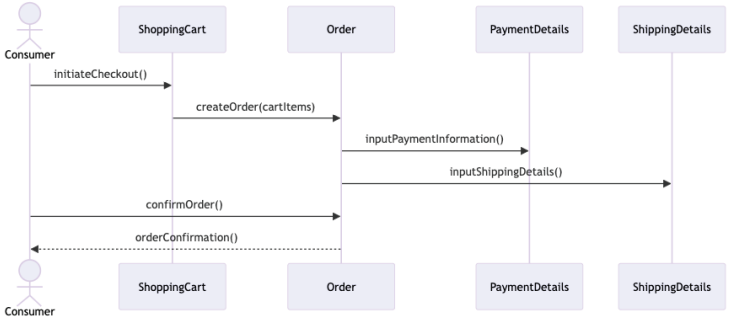


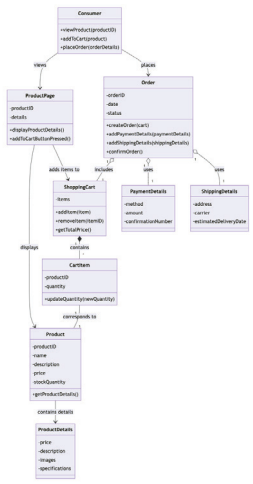
3. Performance across Different Complexity Levels:

The three use cases scenarios selected for the experiment were in an escalating complexity order. The mean scores outline a drop as use cases grow in complexity, however the drop is quite minimal. The data suggests a possibility of limitations of the LLM's capability to effectively handle higher UML diagrams of higher complexity.



UML Class and Sequence Diagrams generated by ChatGPT-4





Conclusions and Future Work

This research underscores the growing importance of exploring Generative AI's role in Software Engineering tasks, particularly in generating software architecture diagrams. While the study reveals that ChatGPT-4 demonstrates moderate efficacy in producing UML Class and Sequence Diagrams, further investigations should extend to other diagram types, involve a more diverse group of evaluators, and integrate a feedback loop for continuous improvement. Additionally, ethical considerations must be carefully addressed as Generative AI becomes increasingly integrated into real-world scenarios.

QR Code for Recording

Case Studies of RestAPI and GraphQL Architecture

Janni Daniel Balraj
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193260@myTUDublin.ie

Introduction

The adoption of microservices architecture has experienced significant growth, driven by its appeal in terms of modularity, scalability, and deployment ease. However, this proliferation of microservices has intensified the demand for efficient data exchange among them. While REST API has long been the standard protocol for microservices communication, its limitations in flexibility and performance have spurred the ascent of GraphQL as a more efficient alternative, as highlighted by studies comparing their performance. As the number of microservices continues to rise, traditional methods like REST APIs prove challenging, leading to issues such as over-fetching or under-fetching of data. GraphQL addresses these challenges by allowing each microservice to define its schema, thereby simplifying data integration and reducing coordination efforts between teams. In addition to its flexibility, GraphQL contributes to enhanced performance by empowering clients to specify their precise data requirements, resulting in a reduction of unnecessary requests and improved response times. Despite these advantages, the integration of GraphQL with microservices presents its own set of challenges, including ensuring data consistency across services and requiring upfront planning for schema and resolver functions. This research paper aims to delve into the integration of GraphQL with microservices, conducting a comparative performance analysis with REST API in an on-premises environment.

Research Questions

The conventional approach, using RESTful APIs for communication, often encounters challenges related to over-fetching and under-fetching of data, hindering system performance and efficiency. The emergence of GraphQL offers a compelling solution to these challenges, allowing precise data retrieval and reducing the volume of redundant requests in a microservices environment.

The core problem addressed in this research is:

Q1 - To what extent does the integration of GraphQL improve the efficiency and performance of data exchange between microservices when compared to RESTful APIs, and

Q2 - what are the implications and challenges of implementing GraphQL in a microservices architecture?

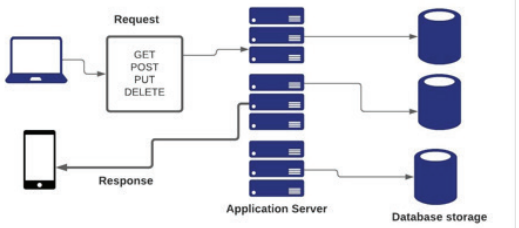
Test Strategy and Methodology

- Functional Testing, Performance Testing and Usability Testing
- Data Preparation: 10 million stock records were generated and inserted into the PostgreSQL database.
- Load tests were conducted using Apache JMeter to simulate concurrent user requests for stock data retrieval.

Architecture

1. REST API Architecture:

Stateless Communication: REST APIs are stateless, meaning that each request from a client to a server contains all the information needed to understand and fulfill the request. The server does not store any information about the client's state between requests.

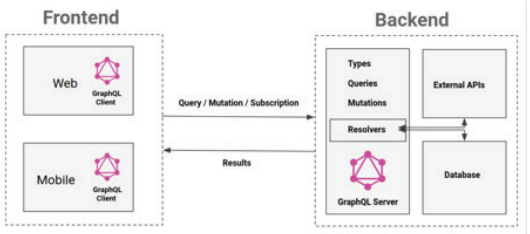


Resource-Based: Resources, such as data objects or services, are identified by URLs (Uniform Resource Locators). Clients interact with these resources using standard HTTP methods like GET, POST, PUT, and DELETE.

Representation: Resources are represented in a format such as JSON or XML, and clients can manipulate these representations.

2. GraphQL Architecture:

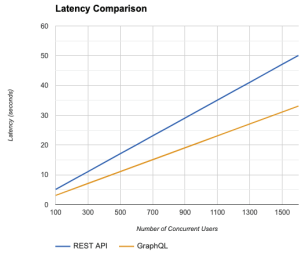
Query Language: GraphQL is a query language for APIs. Clients can request only the data they need, and the server responds with the requested data in a single JSON object.

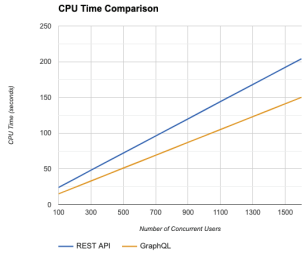


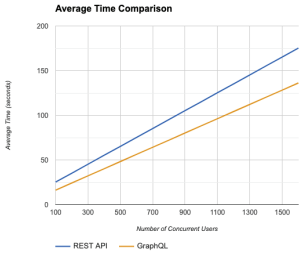
Hierarchical Structure: The structure of a GraphQL query mirrors the structure of the data it retrieves. This hierarchical nature allows clients to request nested data in a single query.

Single Endpoint: Unlike REST, which often has multiple endpoints for different resources, GraphQL typically exposes a single endpoint for all interactions

Performance Graph: RESTAPI vs GraphQL







Conclusions and Future Work

In real-time data scenarios, GraphQL surpasses REST API by efficiently delivering specific data fields, thereby reducing network traffic and enhancing responsiveness. The adaptability of GraphQL enables seamless integration of new data sources without disrupting existing endpoints in complex applications. Looking ahead, the current implementation in a local environment is slated for deployment in cloud platforms like AWS, GCP, or Azure. The future plan includes enhancing scalability and establishing a resilient architecture to optimize performance and reliability.

QR Code for Recording

Infrastructure as code testing, Terraform Vs Terratest

Ruchira Anil More
Department of Computing, TU Dublin, Tallaght, Ireland
X00193212@myTUDublin.ie

Introduction

Cloud adoption has become vital for organizations to deliver new services. Early time to market with stable code delivery has introduced use of automation in DevOps to provision infrastructure. Such automated process of infrastructure provisioning is termed as Infrastructure-as-code. Infrastructure-as-code is one of the practices that many IT industries have embraced to automate their infrastructure provisioning and maintaining process. Though, this has been favourable in administering and provisioning infrastructure using machine readable scripts, there have been issues with the functionality since they might also have defects. The manifestation of defects in the scripts is unavoidable since they are like any other software code, lines of code in a file. Thus, it is essential to have a qualitative analysis on these files to achieve defect free Infrastructure as code. This poster represents a brief overview of the research, where a study was performed to compare the two testing tools/frameworks for PaaS IaC, Terraform test framework Vs. Terratest. Below research question were identified in order to perform further experiments:

RQ1: How does Terratest (Gruntwork) and Terraform-test (Hashicorp) compares in terms of use cases, developer experience, flexibility, exception handling and reporting.

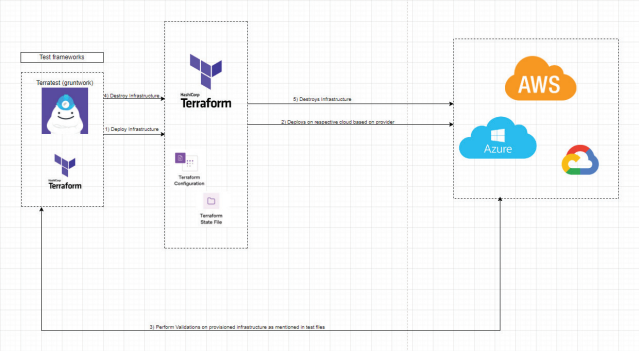
RQ2: Limitations and challenges to adopt Terratest and Terraform test.

Comparison

Metrics	Terratest (Gruntwork)	Terraform's test command (Hashicorp)
Release Date	April 2018	October 2023
Language	Go	Hashicorp's HCL
Syntax	Imperative	Declarative
Exception Handling	Error handling and timeouts implemented to allow scripts to terminate in case of exception.Sometimes infrastructure is not destroyed.	Exceptions are handled and the test terminates and destroys infrastructure as clean-up process.
Built in Helper methods	Good support	Minimal support
Flexibility	Allows to control flow of execution	No control from test framework, only possible from within terraform
Learning Curve	New language, may take some time to kick start	HCL, easy to learn
Execution time	Same as terraform code execution	Same as terraform code execution
Assertion	Importing 3rd party go library for assertions	Built in assertions on conditions
Testing stages	Unit, Integration, end2end	Unit, and Integration, end2end is not supported
Tool support	Terraform, packer, docker , K8s	Terraform, k8s provider within terraform
Cloud support	Tested for AWS, GCP and Azure	Tested for AWS, GCP and Azure
Execution logs	Difficult to read the logs, but with Terratest hepler utility it makes it readable and can be integrated with CI tools.	Easy to read,neat logging on console. Do not create xml report that can be integrated with CI
Reporting	With the helper utility, UI reports can be generated.	No reporting framework
Test file extension	_xx_test.go	.tftest.hcl

Test Architecture, Introduction to Terratest and Terraform test framework

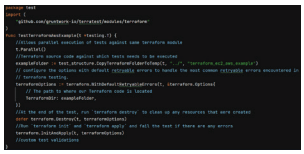
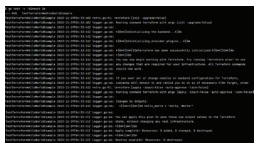
1. Test Architecture



2. Introduction to Terratest and Terraform test framework

Terratest is a testing library developed by Gruntwork. It has libraries and functions that help with testing terraform, docker, packer, Kubernetes code on AWS, GCP and Azure. Hashicorp announced its test framework to test terraform code on 4th of October 2023.The idea behind doing so was to enable terraform practitioners to know that their configurations will function as expected.

Sample test files and output

Conclusions and Future Work

The Terraform test framework is favored by developers due to its ease of adoption and familiar HCL syntax. Terratest offers more flexibility in handling complex tasks, particularly in complex integration tests and end2end tests. Despite Terratest's ability to handle error and exceptions, it sometimes fails the test and cleans up the spun infrastructure, causing manual intervention and potential cost concerns. For users seeking more control, Terratest is a better option. Reporting with Terratest can be improved by using an additional Terratest utility, which allows users to identify errors and generate UI reports. However, Terraform's reporting is limited to console output and cannot be integrated into a UI reporting tool. Overall, Terraform is better for unit test validations, but Terratest is recommended for complex integration tests and end2end tests. As for future work, Using the same set of experiments, the CI/CD support can be tested. Also, as part of Terratest a deeper analysis can be made as it offers other services like testing Kubernetes, it was tested briefly in the above experiments, yet an extensive study can be performed. Along with this, Terratest also offers testing Docker and Packer that was not evaluated in these experiments. One of the other interesting topics can be studying if these test frameworks detect configuration drift and does it provide any feature of analysing the same.

QR Code for Recording

An Analysis of Computational Efficiency in Azure App Service

Robert Beattie
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193224@myTUDublin.ie

Introduction

This paper examines Microsoft Azure App Service, focusing on service tiers, thread counts, and computing efficiency. It explores the performance of single- and multi-threaded workloads, particularly in Fibonacci calculations, across different tiers. Results show that lower cost tiers often outperform premium ones in cost efficiency and computing, albeit with some inconsistencies. The study also addresses the inherent unknowns in Platform as a Service (PaaS), offering insights for Developers, DevOps, and Software Architects. Utilising heat maps and line charts, the research analyses the calculation time for Fibonacci numbers on servers across East US, West Europe, and Southeast Asia. It investigates the correlation between server-tier performance, costs, and regional architectural differences. The findings, presented through graphs and critical observations, underscore the importance of tier selection based on workload and location, concluding with recommendations for cloud service optimisation.

Azure Analysis

This study investigates Azure App Service's handling of single-threaded applications. Two key questions guide the research: **1)** How does Azure handle concurrent HTTP requests in single-threaded architecture regarding core utilization? and **2)** Are there performance differences in Azure deployments across regions? The research uses a Fibonacci calculation API to explore core utilisation and performance across service tiers and regions and cost-effectiveness in cloud applications.

Knowledge Gaps

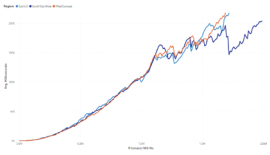
Despite extensive Azure App Service learning resources, gaps remain in understanding the PaaS model's resource utilization. This thesis focuses on four key areas: **Tier Performance**, **Real-World Application Scenarios**, **Impact of Regional Variations**, and a **Detailed Cost-Benefit Analysis**. It examines explicitly the performance and cost of single-threaded versus multi-threaded workloads in Azure, addressing crucial knowledge gaps in cloud computing literature.

Thesis Structure

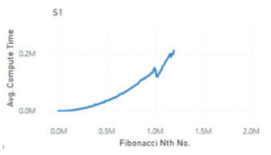
Methodology:

This research examines Azure App Service's performance using a FibonacciCalculationAPI. Tests were conducted across three regions (West Europe, East US, Southeast Asia) on nine Azure tiers, focusing on CPU and memory usage. The API, developed in ASP.NET Core, was monitored using Azure Application Insights. Custom C tests simulated different thread loads, with results analyzed in Power BI to understand efficiency and scalability in cloud environments and regional variation impacts.

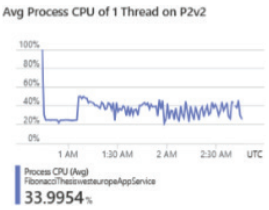
Regional Compute Times:



Tier Compute Times:



Core Utilisation Analysis:

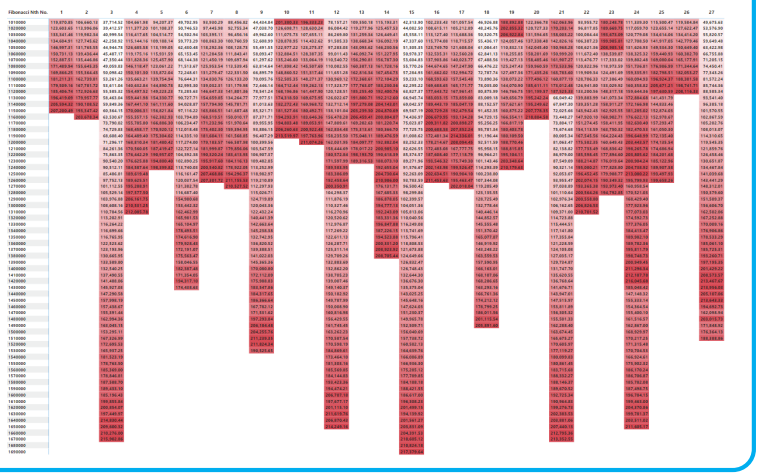


Summary:

Power BI analysis of Azure App Service tests revealed: **1)** Higher performance in premium tiers, suggesting a price-performance correlation. **2)** Single-threaded tasks are uniformly handled, while multi-threaded tasks show concurrency complexities. **3)** The premium tier demonstrates efficient CPU utilization across varying loads. **4)** Cost efficiency is crucial in computational power allocation. These insights guide IT professionals in understanding Azure's hardware management, aiding in informed service tier selection for applications.

Topic Overview

This matrix visualisation analyses thread count impact on Azure App Service performance, using Fibonacci computations as a metric. It shows average compute times across single, two, and four-threaded tests, visualised through a heat map indicating time from white (minimum) to red (maximum). Key insights include uniformity in single-threaded performance, variable multi-threaded behaviour, and the effect of thread failure on overall efficiency. The study highlights the nuances in resource allocation and thread management within Azure.



Conclusions and Future Work

This thesis concludes by synthesizing Azure App Service test findings, addressing how Azure handles concurrent requests and regional performance variations. It confirms Azure's finesse in managing core utilization, especially in premium tiers, and notes regional disparities in resource usage. The study guides developers in service tier and region selection, enriching academic understanding of single-threaded applications in serverless environments. Future work suggests broader, real-time analyses and comparative cloud provider studies for more profound insight.

QR Code for Recording

Software Repository Assessment in DevOps: A Machine Learning Approach to Quality

Edmund Fitzgerald
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193258@myTUDublin.ie

Introduction

This research embarks on an innovative journey to evaluate software repository quality within the DevOps realm. Utilizing a machine-learning model, it analyzes data from the top 100 GitHub repositories in JavaScript, Python, and Java, focusing on commit history and GitHub metrics such as stars and forks. This approach transcends traditional, subjective assessment methods, offering a unique blend of qualitative and quantitative analysis. It aims to establish new benchmarks for software development quality by integrating technical and community-driven data. This study not only contributes to software engineering best practices but also paves the way for advanced AI-driven quality assessment tools in software development.

Machine Learning Model

The model employs Python's 'sklearn' package and integrates four distinct types: Linear Regression, Ridge Regression, Random Forest, and Gradient Boosting. Each model's unique methodology and algorithmic framework are comprehensively detailed, highlighting their specific capabilities in identifying and analyzing patterns and trends within the dataset. This detailing aids in understanding the nuanced performance of each model type and their roles in predictive analysis.

Data Collection and Preprocessing

The data was collected by scraping all the commit metadata for 4 months from GitLab, via GH Archive and storing this information in a PostgreSQL database. This was then queried and the results saved as CSV files for model training. To ensure accuracy once complete, one repository's data was moved out into a separate CSV to verify the model for QA.

Results

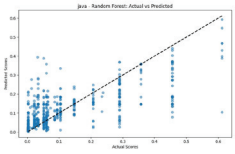


Figure 1: Java

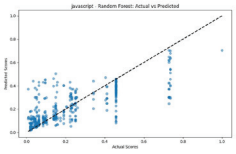


Figure 2: JavaScript

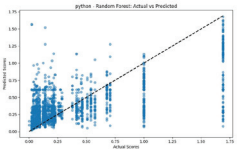


Figure 3: Python

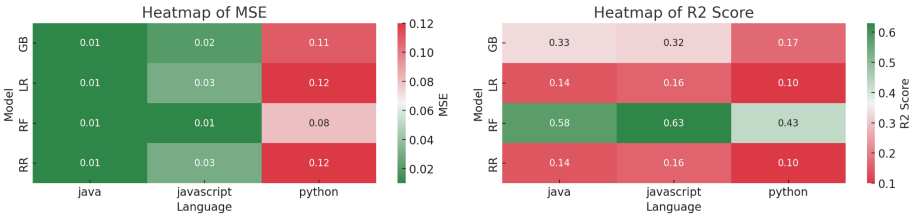


Figure 4: MSE and R squared

The findings of this thesis demonstrate statistical significance, indicating promising directions for future research. However, they currently fall short of being immediately applicable for production use. With further refinement and development, these methods have the potential to not only outperform existing approaches but also provide quantifiable enhancements in practical applications.

Topic Overview

Focusing on the top 100 GitHub repositories in JavaScript, Python, and Java, this research aims to develop an extensive evaluation method by analyzing commit data and GitHub metrics like stars and forks. This innovative blend of qualitative and quantitative analysis seeks to enhance traditional, subjective methods of repository assessment.

The study begins with exhaustive objective data collection from GitHub, considering the top three languages to ensure relevance and broad applicability. It involves gathering comprehensive commit history, offering insights into development practices, and using GitHub comparative metrics such as Stars, forks, and followers as a proxy for subjective surveys or interview data.

Central to this research is a machine-learning model trained on a rich dataset. It employs algorithms adept at identifying complex patterns, which is crucial for understanding the nuances of code quality. The study addresses the challenge of defining and quantifying 'quality' in software repositories, employing community ratings like GitHub stars and followers as proxies, and acknowledging these metrics' weaknesses while offering them as a more useful metric than subjective surveys.

This approach aims to assess the feasibility of using machine learning to predict ongoing repository quality based on objective metrics. The model is expected to be a valuable tool for developers, project managers, and organizations, aiding in informed decision-making regarding open-source projects.

Conclusions and Future Work

This thesis shows machine learning's effectiveness in improving DevOps software repository assessment. Analyzing commit data and GitHub metrics offers an objective approach surpassing conventional methods. It also highlights the challenge of defining 'quality' in software development due to metric subjectivity, suggesting varied quality indicators. Future research areas include:

1. **Algorithm Enhancement:** Apply advanced techniques for improved model accuracy and adaptability.
2. **Tool Integration and Development:** Utilize findings in DevOps tools for real-time quality assessment.
3. **Broadening the Model's Application:** Expand the model to predict various software development aspects.

QR Code for Recording

Technical Debt Tool Comparisons

Basil Roy
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193213@myTUDublin.ie

Introduction

Technical Debt can be sometimes known as tech debt or code debt. Technical debt basically describes what results when development teams take certain actions towards technical implementations to speed up the delivery of a certain functionality of the project to meet business deadlines set which will need refactoring in the future. Briefly Technical Debt can be put forward as the outcome of prioritizing speedy delivery of code over perfect quality code. If you are an individual working in the software industry this is were technical debt creeps up mostly. To tackle this debt there are various technical debt tools available in the market. With Technical debt there are a few risks associated with it. With risks there comes impact. These factors are to be considered as they are very crucial in the long run and on the sustainability of the applications in question.
With the usage of technical debt tools, we can certainly maintain or erase technical debt altogether from our environment, but a thing to lookout for is that there are a lot of tools available in the market. How can we determine which is the correct one for our team / organization. That is the challenge we are facing.

Research Question

RQ1 : What factors mainly influences a user when deciding to Onboard a new tool ?

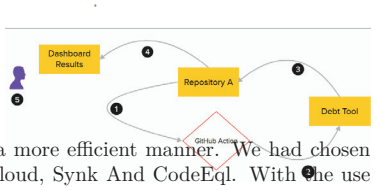
RQ2 : Do users fall into the trap of selecting the most popular tool in the market and not consider the requirements at stake ?

Weighted Factor Results

Factors	Weighted Score of 10
Cost	7
Features	8
Adaptability	9
Usability	8
Support	10

Testing Methods

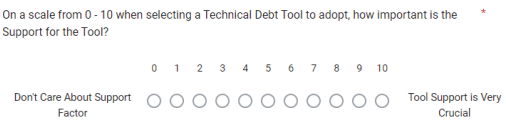
1. Testing Design Flow:



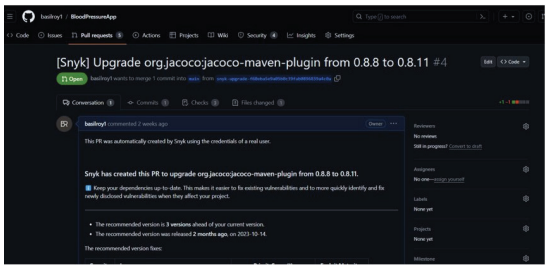
This is the testing design followed for the experiments. It is a simple flow which is repeatable for the 4 repositories that was chosen for this test. The base of the flow starts from our repositories as that is our primary location of code base to scan. We employed Github Action feature which GitHub provides to aid us to setup up scanning in a more efficient manner. We had chosen 3 Tech debt tools for this comparison which were Sonar Cloud, Synk And CodeEql. With the use of Github Actions we can use a template yaml file which has predefined steps for specific tech debt scan. But we had to modify those files to meet our requirements. Since this was a yaml file there are options to trigger the running of the file on commit or manual basis. with this option it allows us to achieve Continuous Integration for running the scan on each commits user do to maintain a healthy tech debt backlog. Once this setup is complete we focus on any configuration needed to make with the repository itself to allow to scan the repository. As there will be permission required and addition of secrets/tokens which is a way the Tool communicates with the Repository. Once the Scan is done we head to the Tool Dashboard for each individual tool and here the users can analyse the scanning results. There maybe stats available which can aid users to see the overall health of the application.

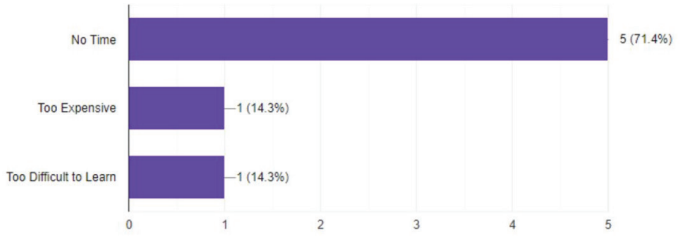
2. Questionnaire:

Survey was an important factor to the data collection for our experiments.For conducting surveys a Sample of users for participating in the survey.Simple random technique was used to collect our sample of 23 users for our survey.



Key Research Findings





Conclusions and Future Work

This research had few limitations such as repository code languages used and only the free trial versions of the tools that's only available to be used for the testing which may have limited features and other reports facility available. In some cases, for other tech debt tools even to get access to the free versions we would need to request access by filling a form in which we have no definite time defined for when the expected response would be, which also limits the usage of other tools in the market.
One area I felt from the research and existing tools is with current boost in the AI sector. So as an area of future works can be what impact does AI (Artificial Intelligence) bring to the current technical debt tools. AI is picking up pace extremely fast in the past year or two. AI is certainly influencing all aspects of the software development cycle. There are AI plugins for the IDE used for development. ChatGPT being the one of the contenders which basically writes code from scratch which is a huge win over the productivity aspects with reduced lead time. Similarly, there are certainly interesting aspects which AI will change the current way of doing things for scanning repositories.

QR Code for Recording

Evaluating the Next Generation Sidecar-less Kubernetes Service Mesh: Ambient Mesh

Anupam Saha
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193210@myTUDublin.ie

Introduction

A service mesh is a dedicated infrastructure layer in the Kubernetes cluster to make service-to-service communication safer, faster, and more reliable. Typically, service meshes are built with a two-tier architecture, housing a control plane for mesh configuration and a data plane to provide mesh functionalities. The data plane consists of a sidecar container that attaches itself to a microservice pod, the smallest deployable unit in Kubernetes. Though this sidecar container architecture reduces a lot of burden on microservice source code, it also increases the overall compute resource usage of pods. To counter this, a new data plane architecture called ambient mesh was born. Instead of running a sidecar container in each microservice pod, ambient mesh implements the data plane by deploying a single proxy per node. This utilizes the Linux eBPF technology and brings a lot of excitements in service mesh space. Ambient mesh was originally developed by Solo and Google, but now it is part of the open-source project Istio. This research explores the ambient mode that comes with the newer version of Istio to evaluate its compute resource usage and operational complexities.

Objectives

RQ1. Does Istio ambient mode consumes less compute resources than sidecar mode?

RQ2. Does the sidecar-less architecture reduces operational complexities?

Motivation

With the release of ambient mesh alpha version as part of Istio, a massive improvement in compute resource consumption is reported over sidecar architecture in multiple gray literature, however, no academic research paper is available. While most of the service mesh products available today either use Envoy or in-house-developed sidecar proxies to determine the success factor, ambient mesh applies a completely different approach by leveraging Linux eBPF technology and separating layer 4 and layer 7 capabilities. While eBPF gives the ambient mesh to improve its resource efficiency, two separated layers allow Kubernetes administrators a different level of flexibility while deploying Istio in their environment. All of these bring an exciting opportunity to evaluate and publish an academic research report on ambient mesh.

Methodology

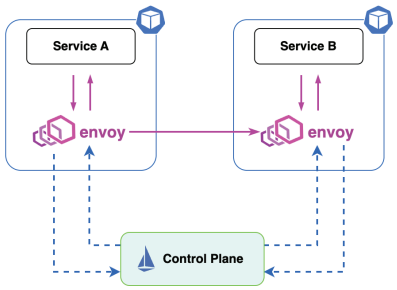


Figure 1: Istio Sidecar Mode Design

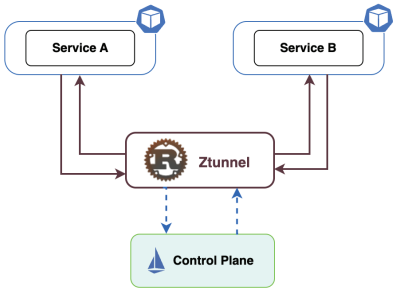
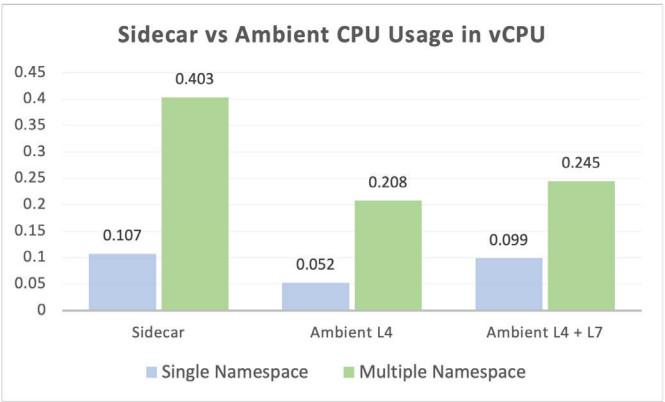
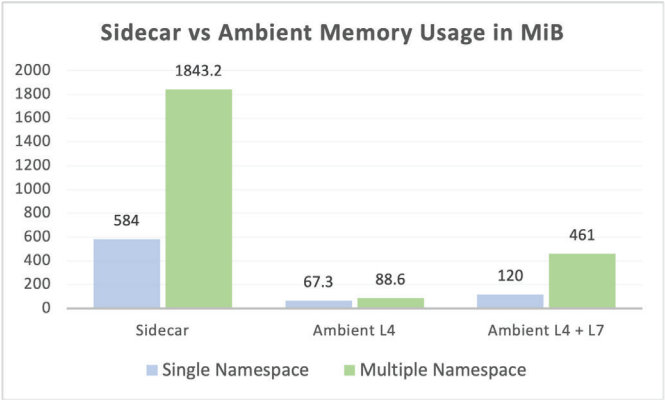


Figure 2: Istio Ambient Mode Design

A research platform is built to investigate the compute resource consumption by Istio sidecar and ambient modes after exploring the ambient mesh architecture. In sidecar mode, a tight-coupled architecture is seen where envoy proxies are attached to each running service pod, as shown in Figure 1, whereas ambient mode deploys a single Ztunnel proxy per node, as shown in Figure 2. The research is based around the principles of having multiple test executions with Ztunnel and Waypoint proxies engaged in ambient mode and Envoy proxies attached to each pod where a demo application, Bank-of-Anthos (BOA), is deployed to a Kubernetes cluster. To follow an enterprise-grade deployment model, BOA is deployed in single and multiple Kubernetes namespaces to explore the intensive nature of Istio in both modes. Google Cloud Platform is used as a cloud provider for the research, where Kubernetes clusters are provisioned using Terraform and Istio is installed using Istioctl and Istio operator. To engage Istio in resource-intensive filtering, 10-minute load testing is performed on BOA for each test session, along with layer 4 and layer 7 traffic filtering. Test results are captured from the Grafana dashboard by applying Prometheus metrics for CPU and memory. To measure operational complexity, Istio is upgraded using a blue-green deployment strategy to check whether a pod restart is required to apply the new version of Istio to the running microservices.

Research Findings





Conclusions and Future Work

While the sidecar-less model of ambient mesh brings some significant improvements in compute resource consumption and operational excellence, the sidecar model of Istio is well established and widely used today. In near future when a stable version of ambient mesh is released, this scenario shall be changed and ambient mode may become the default installation mode for Istioctl. Leveraging Linux eBPF technology, ambient mesh brings a lot of opportunity in service mesh space for future research. Some of which can be categorized as Istio's network latency due to Ztunnel and Waypoint proxies distributed nature and supporting Kubernetes Jobs or server-send-first protocols features which are currently not supported by Istio sidecar mode.

QR Code for Recording

Combining Web Application Security Testing Tools

Darragh Madden
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180709@myTUDublin.ie

Introduction

2023 has seen numerous high-profile instances of data leaks and breaches across a variety of industries. It is imperative that organisations have a comprehensive security testing strategy in place to mitigate the risk posed by malicious actors. To aid in securing web applications, organisations typically employ one of three approaches: Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Interactive Application Security Testing (IAST). Research has found that SAST & DAST tools are prone to identifying high numbers of false positives (report vulnerabilities where no issue actually exists). In an effort to reduce the number of false positives reported by tools research has investigated combining tools with increased vulnerability detection being observed.

Research Question: Does combining SAST, DAST & IAST tools result in enhanced vulnerability detection compared to using each tool individually?

Methodology

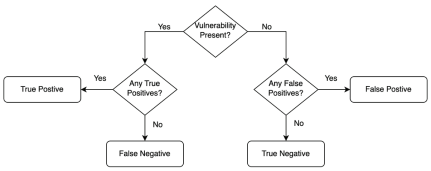
1. Tool Selection
This research investigated the performance of a combination of Open Source tools: FindSecBugs (SAST), OWASP ZAP (DAST) and Contrast Community Edition (IAST).

2. Benchmark
The OWASP Benchmark is an open source web application built in Java and is deployed in Apache Tomcat. The latest version of the Benchmark (v1.2) contains a set of approx. 2,700 fully exploitable test cases which can be analysed by security testing tools to detect true and false positives. A perfect score would see a tool identify 100% of true positives and 100% of true negatives.

3. Experiment
Each tool and the OWASP benchmark were downloaded and run locally. The benchmark provides shell scripts for running both FindSecBugs and Contrast CE. OWASP ZAP was run in two steps: first using the Spider to identify all relevant HTTP links in the web app and then the Active Scan was run. The benchmark provides a separate shell script to convert the raw results files (XML & LOG) into CSV files containing the test results.

4. Method to Combine Results
Simple 1-out-of-N approach was used to combine the results for every test, for each of the 3 tools, to get a Combined Tools result.

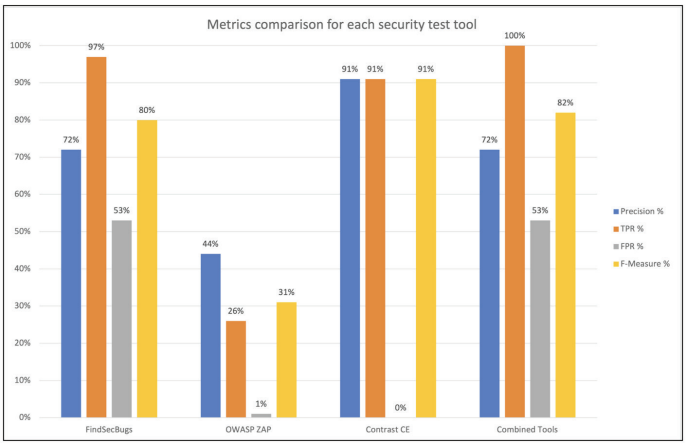
1-Out-of-N Approach

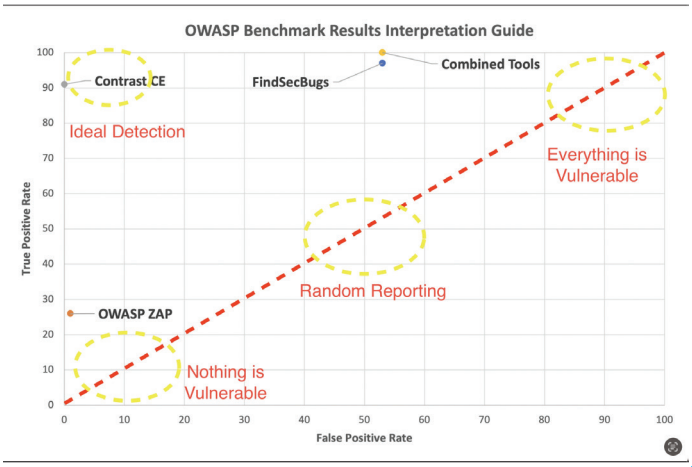


Combined Tool Test Results

	True Positive	False Negative	False Positive	True Negative
Command Injection	126	0	111	14
LDAP Injection	27	0	27	5
Path Traversal	133	0	129	6
Secure Cookie Flag	36	0	0	31
SQL Injection	272	0	210	22
Trust Boundary Violation	83	0	35	8
Weak Cryptography	130	0	0	116
Weak Hashing	129	0	0	107
Weak Randomness	218	0	0	275
XPATH Injection	15	0	19	1
XSS (Cross-site Scripting)	246	0	109	100
Total	1415	0	640	685

Results





Conclusions and Future Work

This study found that a combination approach can lead to enhanced vulnerability detection. A benefit of the approach is that each tool found vulnerabilities that the others did not, which resulted in 100% vulnerability detection.

A vital consideration however are the tools to be included: FindSecBugs reported a large number of false positives which the Combined Tools approach inherited. Surprisingly, OWASP ZAP didn't report many instances of vulnerabilities at all. These findings may bring into question the usefulness of a combined approach. A recommendation to mitigate this, and to maximize efficiency, is to give preference to tools with known low false positive detection.

Future work may research a combination approach utilizing commercial tools or implement a benchmark with modern vulnerability types present such as NISTs JULIET web application. Given the IAST tool was accurate, further research could also investigate combining multiple IAST tools to assess their performance.

QR Code for Recording

A Performance and Cost Analysis of Java-based Function-as-a-Service on AWS

Alan Kavanagh
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00080986@myTUDublin.ie

Introduction

This study consists of a research project that compares and contrasts the performance and cost of Java-based Function-as-a-Service (FaaS) solutions on Amazon Web Services (AWS). The objective of the research is to understand the improvement that each solution offers with regard to cold start mitigation for Java-based serverless functions on AWS. A series of experiments are conducted to assess the start-up performance and associated cost of lambda functions deployed to AWS Lambda, AWS Lambda with SnapStart, and AWS Lambda with Provisioned Concurrency. The results of each experiment are collected, depicted, and analysed, and then the findings and conclusions are presented in the form of a performance and cost comparison.

Hypothesis

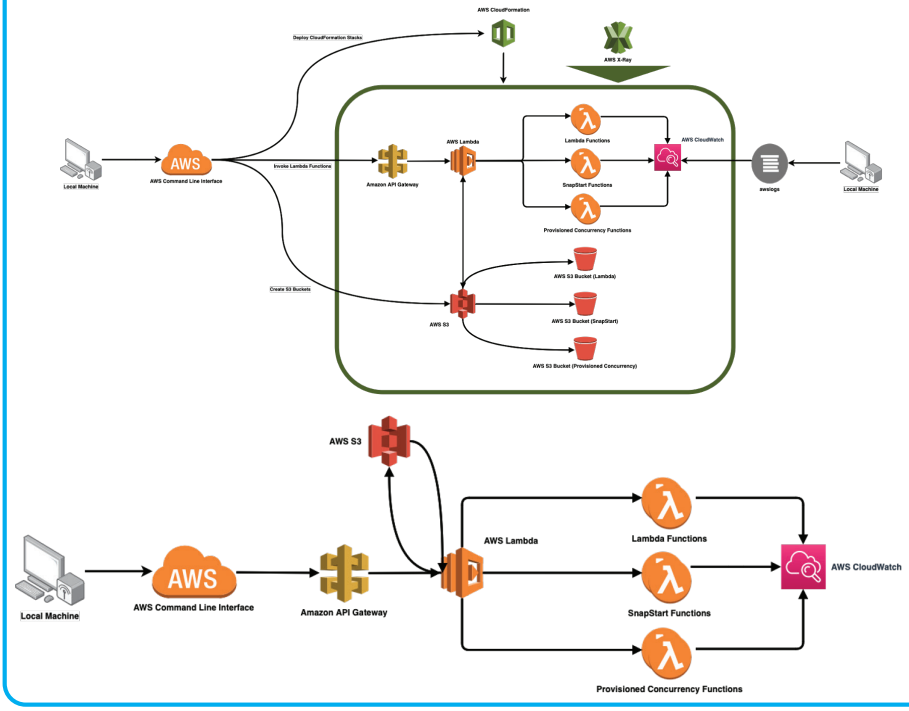
"Lambda SnapStart for Java can improve startup performance for latency-sensitive applications by up to 10x at no extra cost" - AWS

It is expected that the results produced will help identify an increased start-up performance for SnapStart in comparison to Lambda and Provisioned Concurrency. If the hypothesis is correct, the results should indicate a reduced cold-start latency, and reduced number of cold-start occurrences, without incurring any additional costs, when deploying Java-based serverless functions on AWS Lambda with SnapStart enabled

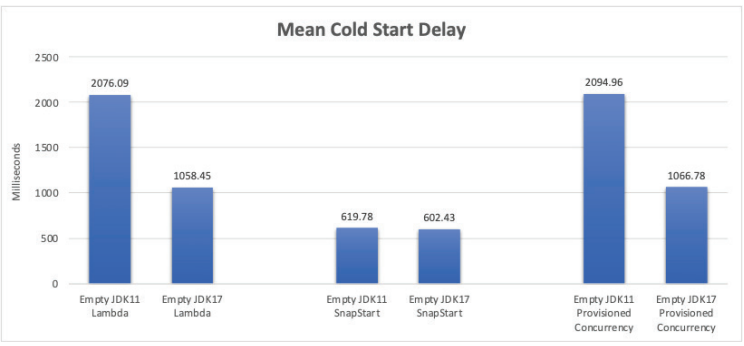
Research Questions

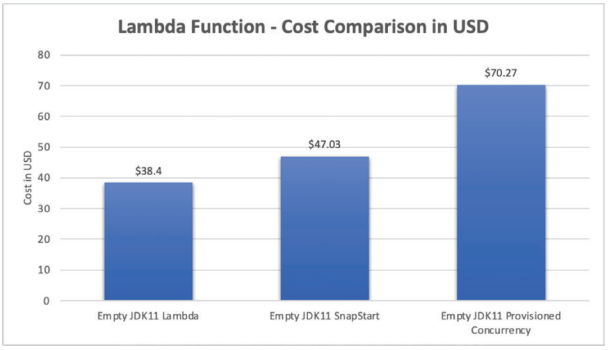
- RQ1. What impact does AWS Lambda with SnapStart have on the cold start latency, and cold start occurrences, of Java-based serverless functions?
- RQ2. How does AWS Lambda with SnapStart perform in comparison to AWS Lambda, and AWS Lambda with Provisioned Concurrency?
- RQ3. Does AWS Lambda SnapStart increase start-up performance by 10x at no extra cost?

Deployment Architecture and Invocation Flow



Results and Findings





Conclusions

SnapStart does not decrease the number of cold start occurrences in comparison to AWS Lambda. SnapStart consistently has a 600ms restoration time, and outperforms AWS Lambda for functions with a higher initialisation time. SnapStart incurs the same costs as AWS Lambda for higher execution time functions, or slightly more for lower execution time functions. Provisioned Concurrency outperformed SnapStart in all scenarios, however, it incurred additional costs up to x5 or x6 in comparison. Upgrading from JDK11 to JDK17 can decrease the average execution time by 75%.

QR Code for Recording

Comparing Tsetlin Machines to DNNs in model performance and efficiency

Stefan Przemyslaw Gliniecki
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X12345678@myTUDublin.ie

Introduction

The research compared a novel and uprising Tsetlin Machines paradigm to Deep Neural Networks in terms of model performance as understood by the Accuracy and F1 metrics, and model efficiency as understood by energy and time consumption of training and running the models.
Experiments for 9 datasets have been performed, so that the functional relationship between dataset characteristics and the comparative model performance and efficiency could be investigated.
For obtaining results of statistical significance bootstrap procedure has been used. The efficiency differences proved TMs to be better than the DNNs, with comparable results for the model performance measures. Main findings of the prior work have been replicated and built upon this work.
The differences in both performance and efficiency correlated with class imbalance while number of features and number of instances had influence on the prediction time and energy, which grants support for future work.

Sensitivity to Hyperparameters

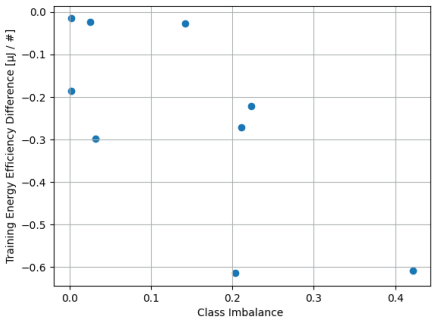
Tsetlin Machines scored much better in the energy efficiency, where the results did not depend as heavily as the Deep Neural Networks. This is perhaps because multiple parameters of the DNNs are expected to influence this metric, which are the number of hidden neurons, number of examples in the learning batch or the number of epochs, where there are only two parameters expected to have significant impact on the efficiency, which are the number of clauses and the number of epochs.

Performance and Efficiency

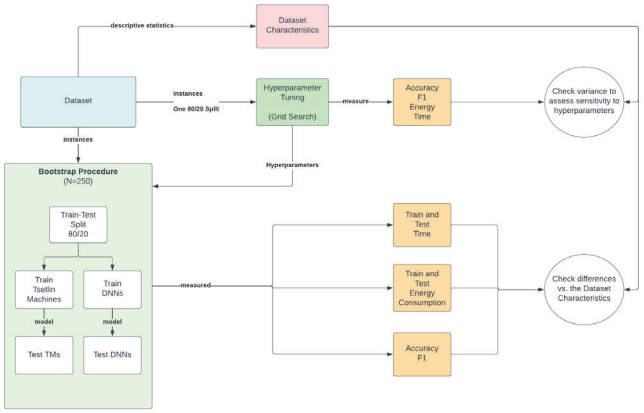
The differences in every measured metric is almost entirely statistically significant.
Findings from [**?**] are replicated entirely for the energy efficiency benefit of Tsetlin Machines over the Deep Neural Networks.
The model performance metrics are comparable: in some of the datasets such as ANNEALING, BC and FLAGS Tsetling Machines are outperforming the DNNs.
It is worth mentioning that the datasets where DNNs thrive are the purely numeric datasets such as HVR, MNIST or SONAR, the latter also replicating findings of the literature.
What is also worth mentioning is that both time and energy efficiency differences are very high.

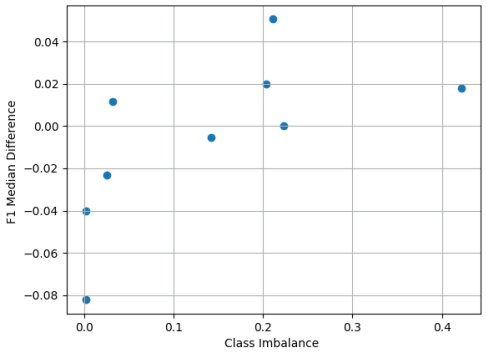
Differences as a function of Dataset Characteristics

Due to a low number of data points (N=9), only a Spearman-ρ statistics was computed, and the findings should be treated as a source of directional hypothesis for future research rather than an evidence of a strong relationship.
The differences in energy and time prediction efficiency seems to diminish with the rise of the number of instances in the training set and the number of features. Both effects could be partially explained by other characteristics of the dataset such as existence of solely numeric data for MNIST, however the data warrants future research on whether the prediction time benefit diminish with the rise of the models complexities.
It is also an interesting, however a subtle effect, that the Class Imbalance influences the $F1$ of Tsetlin Machines negatively as compared to the Deep Neural Networks. This may mean that TMs indeed do not perform well for under-represented classes.
This hypothesis gains support when looking at the training energy efficiency as seen in the Figure below as less complex thus less performing model should require less energy to train.



Topic Overview





Conclusions and Future Work

The research focused on the comparison of two machine learning models: Tsetlin Machines and Deep Neural Networks in terms of energy efficiency and model performance as a relation of the dataset characteristics. The relationships between the Dataset Characteristics should serve as an input to the continuing research, where especially experiments for tasks with imbalanced classes should be performed, perhaps by removing subsets of the classes from complete datasets.

QR Code for Recording

Istio Service Mesh vs. Capsule Operator for Kubernetes Multi-tenancy

Sooraj Shajahan
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193249@myTUDublin.ie

Introduction

The growing adoption of Kubernetes and complex deployment requirements gives rise to the need for companies to identify possible opportunities to reduce infrastructure costs, improve resiliency, and optimize operational efforts. One of the possible solutions to realizing these operational benefits is the usage of multi-tenant architectural patterns on Kubernetes. Multi-tenancy pattern in software infrastructure enables multiple users and organizations to use the common underlying infrastructure while allowing for data and resource level isolation. This research aimed to compare the efficacy between a current widely used multi-tenancy implementation pattern that uses the Istio service mesh and a similar multi-tenant implementation that uses the Capsule operator on top of Kubernetes.

The results showed that multi-tenancy implementation using only the Capsule operator did much better in terms of the latency, and requests per second (RPS) and used comparatively less Kubernetes cluster resources than the Istio-based implementation.

Motivation

The motivation for this research is to objectively compare an existing widely used multi-tenancy implementation pattern that uses the Istio service mesh and a similar implementation that uses the Capsule operator to identify, highlight, and provide recommendations to the personas in software development to decide or be informed of the considerations while making the technology choices.

Research Questions

RQ1: *What are the existing hard multi-tenancy implementation models in Kubernetes?*

RQ2: *Do multi-tenancy implementation with Capsule perform better than the implementation using the Istio service mesh?*

RQ3: *What are the pros and cons of an implementation using the Capsule operator?*

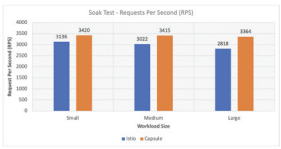
Results

1. Capsule-based implementation was less resource intensive

It was observed that the difference in CPU usage ranges from a minimum of 3% during the load test of small workloads to a maximum of 11% during the soak test of large workloads.

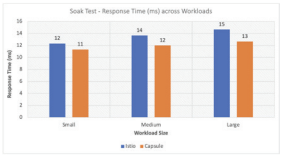
2. Capsule-based implementation performed better

The multi-tenancy setup that was based on the Capsule operator performed better than the setup that was based on Istio. The differences in RPS ranged from 284 RPS for small workloads to 545 requests for large workloads.

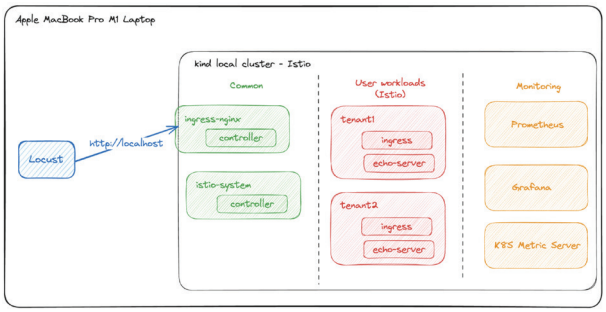


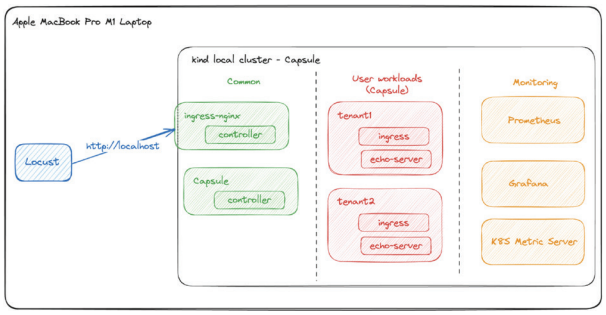
3. Response Time was better on Capsule-based implementation

The differences in response times ranged from 1 ms for small workloads to 2 ms for medium and large workloads between the two implementations.



Performance Test Setup





The above figures show the performance test setups used to compare the multi-tenancy implementations based on the Istio service mesh and Capsule operator. Python-based Locust was used as the performance testing tool. Prometheus, Grafana, and K8s Metrics Server based setup were used to capture the resource usage and performance metrics.

Conclusions and Future Work

RQ1 was answered through the literature review and different implementations one using Istio service mesh and the other using Kubernetes KCP operator were identified.

RQ2, the Capsule-based multi-tenant implementation was proven to be performing better with higher throughput and comparatively lower resource consumption than the Istio-based implementation.

RQ3, it is recommended to use ***Capsule*** as the choice of Operator for implementing multi-tenancy instead of the Istio service mesh, when some advanced features like mutual TLS (mTLS), HTTP Request Retries, and Virtual Services are not required among the micro-services.

Future work in this area could look at (i) the Ambient Mesh pattern from Istio (ii) other service-meshes like HashiCorp Consul, Linkerd, Open Shift, Nginx, etc.

QR Code for Recording

An analysis of Terraform as an enabler of a multi-cloud strategy

Ruaidhri Moran
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193219@myTUDublin.ie

Introduction

As cloud computing adoption continues to grow in Enterprises, increasingly "Multi- Cloud" is being promoted as a solution to issues with vendor lock-in, redundancy and access to the latest cloud service offerings. In this context, Multi-Cloud refers to the use of multiple Cloud Providers together based on the requirements of an organistation, and specifically this research project deals with the three largest (by market share) Cloud Providers: Amazon Web Services, Microsoft Azure and Google Cloud Platform. The thesis looks at Infrastructure as Code (IaC) and its impact on Multi-Cloud, investigating whether there are benefits to using an IaC tool for implementing this strategy, specifically Terraform tool created by Hashicorp.

Hypothesis

The initial hypothesis was that by using an IaC tool, one could switch cloud provider with relative ease in cases where there was a motivation due to any reason. It was hypothesised that the basic tenants of the cloud providers would be the same and that very little would have to change in the code between the different Cloud Providers; essentially one would be able to design an architecture and deploy this with low levels of effort (time and code-changes) on any other cloud provider and achieve the same results.

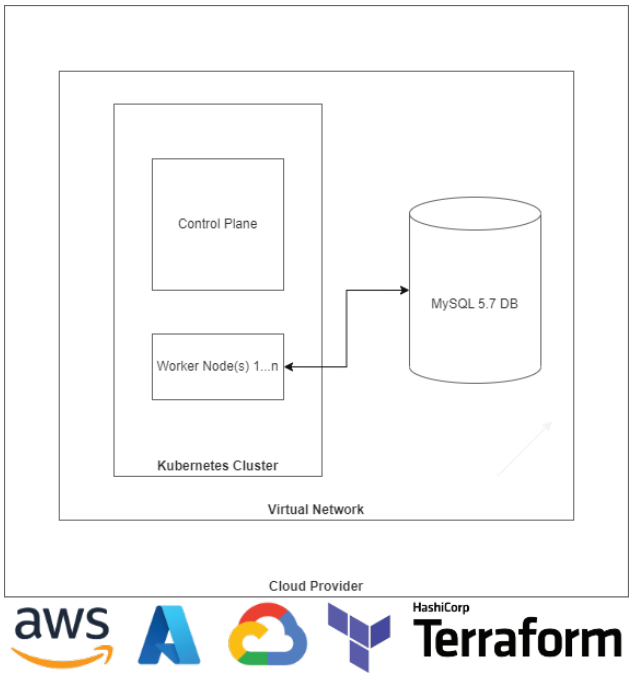
Research Questions

RQ1: "How can IaC tools support the deployment of a multi-cloud strategy?"

RQ2: "Practically, what is involved in a multi-cloud strategy?"

RQ3: "How do different Cloud Providers interact with Terraform?"

Experiment Design



Experiment & Results

The experiment involved the deployment of an architecture (shown in the diagram above) of a MySQL 5.7 database with public internet access, a virtual network and associated subnets, a Kubernetes cluster with an autoscaling node group of 1-n nodes, a WordPress workload running in the node group and connected to the database, and finally the handling of all required rules and accesses including usernames/passwords. The experiment was sufficiently complex to analyse the full use of Terraform with each of the Cloud Providers, and to support the answering each of the research questions given above.

The main result out of the research was that while Terraform is useful and offers a number of key benefits for developers and administrators, it still requires a knowledge of the relevant Cloud Provider as a key part of working with these applications is to understand the services and the Terraform providers used to interact with them. Another result was around the use of Kubernetes, and while each Managed Kubernetes Service had differences, deploying Wordpress to the Kubernetes service, allowed the use of the same code.

Conclusions

Although the initial hypothesis was proven wrong through the research project, it was found that using Terraform offered many benefits and did help to support a multi-cloud approach moreso than using each individual provider's console. In analysing what was required for a multi-cloud strategy, it was identified that deep knowledge of a Cloud Provider's service offerings, knowledge and understanding of the security aspects, and a well thought-out Architecture were all critical success factors. Finally, analysing how each of the Cloud Providers interact with Terraform were detailed through the experiments.

Overall, multi-cloud is a growing and important area of practice and research for academia and organisations alike. The use of an IaC tool, such as Terraform, allow you to essentially speak the same language as you build your infrastructure, although the individual metamodels of each of the Cloud Providers mean that you are essentially speaking a different dialect to each of them. This difference in the metamodel when describing services, infrastructure, SLAs, and SLOs mean that there is a significant workload involved in deploying a multi-cloud strategy and while IaC solves some of the problems, and reduces some of the human effort, it is not a panacea and does not replace knowledge of the Cloud Providers at this time.

QR Code for Recording

ML pipeline performance comparison

Ben Stuart
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193209@myTUDublin.ie

Introduction

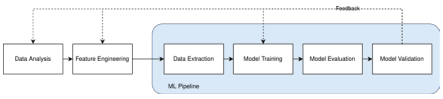
This project aimed to analyse and compare machine learning pipeline architecture performance to identify performance differences between architectures. Interest in operating machine learning has become a growing topic in the data science community, which has brought an increased focus on MLOps. Little research has been done on MLOps to date, with most works focusing on foundational information collection through literature reviews, interview studies and proof of concept architectures. Progress has been achieved in establishing a high-level state-of-the-art, but much more research is required to identify future work and deepen collective knowledge. Machine learning pipelines are often used for continuous training and machine-learning tasks in an MLOps context to automate and orchestrate model training, delivery, and other tasks. This work compares Metaflow, Apache Airflow and SageMaker pipeline frameworks deployed to AWS-based infrastructure regarding resource requirements for different training and inference workloads and runtime environments (Kubernetes cluster, SageMaker jobs and AWS Batch).

Research Question

This work aimed to discover how distributed machine learning pipeline architectures compare resource utilisation and time taken to orchestrate tasks with equivalent workloads and resource allocation.

What is an ML Pipeline?

Continuous Training pipelines are often called Machine Learning pipelines (MLP), sometimes referred to as *ML workflow pipelines*. These pipelines are written as discrete interdependent steps, forming a directed acyclical graph (DAG). Writing workflows in this way allows these pipelines to be orchestrated on distributed systems, re-run steps independently, and utilise available compute resources efficiently. Notable works and interview studies have shown writing ML workloads as MLPs avoids known pitfalls in notebooks such as scalability and low code quality.



Experiment design

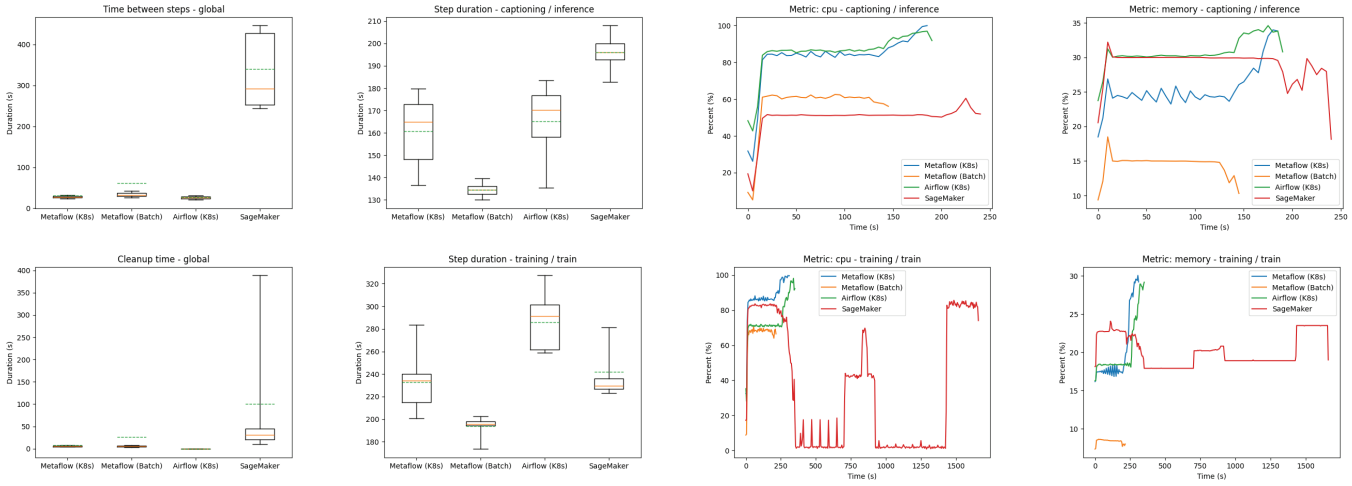
1. Workloads:

Two separate workloads have been created to test the four tools. The first is a batch inference workload to caption a set of images, and the second is a model training workload which trains a neural network. These workloads have been chosen as they represent everyday use cases for MLP tools and two critical stages of the model life-cycle, training and inference.

2. Architecture configuration:

This study focused on four architecture configurations: the Metaflow pipeline framework, backed by Kubernetes cluster for compute resources; Metaflow us- ing the AWS managed service Batch for compute resources; Apache Airflow, also supported by a Kubernetes cluster and AWS's managed service SageMaker. These configurations have been selected to represent a mixture of open-source and self-hosted solutions and proprietary services and hybrids of the two.
Metaflow is an open-source MLP framework initially developed at Netflix; container-based service that can be backed by various orchestration and computing resources such as Kubernetes, AWS Batch, and Airflow. In this study, both Kubernetes and AWS Batch based architectures were used.
Apache Airflow is an open-source workflow orchestration platform. Like Metaflow, Airflow has a Python library for describing workloads, known as DAGs (directed acyclic graph), and a container-based service that can schedule tasks on multiple backend compute resources, including Kubernetes used in this study.
SageMaker is a fully managed service from AWS that offers many sub-services, including scheduling various job types using SageMaker pipelines. SageMaker pipelines comes with an SDK for describing and executing pipelines. In this study, SageMaker Processing or Training jobs were used throughout to execute tasks.
All data was collected using OpenTelemetry exporters and psutil functions.

Results



Conclusions and Future Work

The results of this work have shown that behaviours across architectures can differ vastly, and some may be beneficial to different use cases. It has also been shown that while framework choice can contribute to performance; it is most impacted by good, well-architected compute and orchestration resources. Future work is required to understand deeply how data scientists use machine learning pipeline frameworks, what features are most desirable, and the pain points with these tools.

QR Code for Recording

Comprehensive Study of Container Orchestration Frameworks

Ashwini Ravikumar
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180728@myTUDublin.ie

Introduction

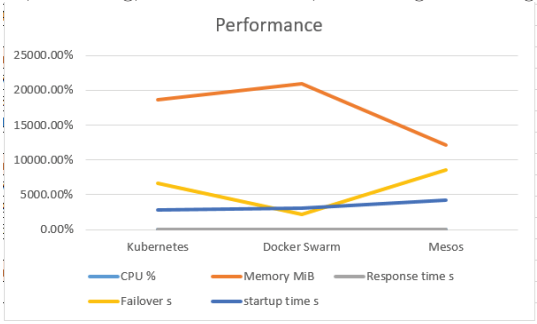
Container orchestration has emerged as a crucial technology in modern software development and deployment, providing a systematic and automated approach to application definition, deploying, managing resource allocation, scheduling, container creation, networking and scaling containerized applications. This paper offers a comprehensive overview of container orchestration, fundamental functionalities, performance and future prospects. Conducting a qualitative analysis of container orchestration technologies involves evaluating their characteristics, skills ease-of-use, advantages, disadvantages, and appropriateness for various scenarios/use cases based on the organisational needs. While they all offer container orchestration capabilities, they differ in their approach, features, and strengths, catering to different use cases and preferences. Tools leveraged for comparative study: Kubernetes,Docker Swarm,Apache Mesos. These tools are presently a shortfall of thorough and objective assessment of their functionality and performance. This absence hinders IT managers in selecting the most appropriate orchestration solution. The empirical data demonstrates that Kubernetes surpasses its competitors in terms of performance when it comes to very complex application deployments.

Functional and Performance metrics evaluation

RQ1: Comparing the key features of chosen orchestration frameworks
RQ2: Comparing performance metrics for bench-marking
1.Qualitative Analysis: Comparing Common features vs Unique features: Gave us a clear understanding of features from which we can choose a CO framework which suits our needs and case scenario.
2.Quantitative Analysis: We used Grafana, Prometheus, and built-in cAdvisor to monitor the performance of CO frameworks for the below scenarios and conducted test experiments to measure performance of the chosen CO frameworks.
Scenario 1: Deploying 2 Container (Nginx-Go)
Scenario 2: Deploying 3 Container (InfluxDB)
Scenario 3: Deploying 4 Container (WordPress)

Topic Overview

Will a functional and performance comparative evaluation be good enough to identify the best container orchestration tool? Evaluate the experiments such as cluster provisioning time the orchestration tools take to deploy, failover times, startup time and so on. This thesis contributions are mainly of two research parts. The 1st is the study aiming to offer the functionality analysis of Apache Mesos, Docker Swarm and Kubernetes. The 2nd is the study aiming to offer the performance analysis conducting experiments to measure or analyse the application definition, deploying, managing resource allocation, scheduling, container creation, networking and scaling containerized applications.



Conclusion and Future Work

Selecting an orchestration tool is a strategic choice that affects how well and efficiently applications operate in a clustered environment, not just a question of personal taste. This study presents a comprehensive analysis of the three container orchestration tools, comparing the various features and services provided by different container orchestrators.
Answering RQ1 comparing common and unique features of chosen container orchestration tools. We examined based on application provisioning, varied complexity test, container failover, CPU and Memory usage, response time to answer the RQ2 comparing performance metrics. Functional and performance comparative evaluation facilitates in identify the best container orchestration tool based on the use case and complexity of the project.

Kubernetes is currently one of the most comprehensive orchestrators when it comes to functional comparison available in the market. This is why practitioners are choosing to favour it above others. Simultaneously, the intricate structure of the system can, in certain instances, impose a substantial burden that could impede its performance. We also faced challenges like cost constraints of cloud resources, Security challenges, Scalability challenges during our research work.

Future work: Organizations can efficiently adopt and extend the utilization of Istio in Kubernetes. Security implementation and default feature provided the container itself can be studied in depth on CO frameworks.

QR Code for Recording

A Comprehensive Evaluation of Helm and Helmfile for Efficient Management of Microservices

Trinath Chakka
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193221@myTUDublin.ie

Introduction

Microservices are supported by Kubernetes, a popular container orchestration platform run by the Cloud Native Computing Foundation that offers capabilities including automated deployment and scalability. This research evaluates critical performance aspects, focusing on chart execution time and service uptime on Helm vs Helmfile. The goal of the study is to get a better knowledge of Helm and Helmfile's respective advantages and disadvantages when it comes to coordinating deployments within Kubernetes frameworks by analysing various deployment methodologies.

Deployment Comparision Table

Model (Services)	Branches-Runs	Average Time for all branches
Helm Small (5)	Main-10	3m 59s
	Downgrade-10	3m 30s
	Upgrade-10	3m 31s
	Rerun-10	2m 17s
	Total = 40	Total:13m
Helm Medium (25)	Main-5	18m 6s
	Downgrade-5	20m 51s
	Upgrade-5	18m 40s
	Rerun-5	7m 7s
	Total = 20	Total:64m
Helm Large (40)	Main-3	28m 33s
	Downgrade-3	31m 59s
	Upgrade-3	30m 19s
	Rerun-3	16m 54s
	Total = 12	Total: 107m
Helmfile Small(5)	Main-10	4m 21s
	Downgrade-10	3m 25s
	Upgrade-10	3m 6s
	Rerun-10	1m 51s
	Total = 40	Total:12m
Helmfile Medium (25)	Main-5	16m45s
	Downgrade-5	15m50s
	Upgrade-5	13m18s
	Rerun-5	3m15s
	Total = 20	Total:49m
Helmfile Large (40)	Main-3	32m 28s
	Downgrade-3	20m 02s
	Upgrade-3	15m 22s
	Rerun-3	6m 40s
	Total = 12	Total:74m

Research Questions

RQ1 - How do chart execution times differ between Helm and Helmfile, particularly when small, medium, and large components application scenarios are deployed?

The outcomes clearly show that Helmfile consistently outperforms Helm, with deployment efficiency across small, medium, and large models increasing by a factor of three. Helmfile's continuous advantage over Helm is indicated by its faster deployment times. Helmfile's efficiency benefits are highlighted by the significant time savings seen in all deployment sizes, making it an appealing choice for enterprises looking to streamline Kubernetes deployment procedures. Helmfile's continued exceptional performance confirms its continued relevance as a tool of choice for orchestration in a variety of deployment situations.

RQ2 - What effect do Helm and Helmfile have on Kubernetes application service uptime, and how do their performances differ in terms of maintaining dependable and continuous service availability?

In downgrade and upgrade circumstances, Helmfile regularly performs better than Helm, but Helm typically delivers quicker service uptime, especially in big branches. Local state management and the declarative nature of Helmfile help to speed up service uptime during upgrades. This sophisticated knowledge highlights the need of picking orchestration tools customised to particular requirements for continuous service availability in a variety of Kubernetes environments, enabling organisations to make decisions between Helm and Helmfile based on their unique deployment needs.

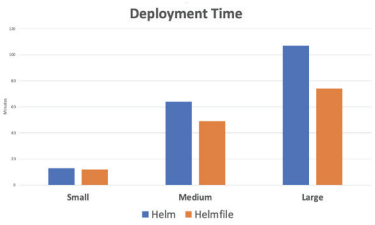
RQ3 - How does Helmfile's declarative approach and local state maintenance effect the speed and reliability of complex deployment scenarios, and how does it handle Helm's management challenges?

In complex deployment scenarios, Helmfile's declarative style and local state management greatly improve speed, reliability, and efficiency over Helm. Helm's drawbacks are addressed by Helmfile, which enables faster and more effective deployments by maintaining a local state and expressing the planned deployment state reliably. This is especially useful in large-scale installations. The ability of Helmfile to track changes at a finer level is useful for managing version transitions and gives enterprises an easier-to-use orchestration tool for complex Kubernetes installations.

Test Strategy

The experiment was carried out on Azure, using Azure DevOps for rigorous testing. The infrastructure was divided into two separate projects for Helm and Helmfile, with repositories aimed to small, medium, and large applications. Each repository, constructed with 5, 25, and 40 services. To enhance precision and robustness, we conducted 10 cycles for the small branch, 5 cycles for the medium branch, and 3 cycles for the large branch. Each repository included four important branches: "main", "downgrade", "upgrade" and "rerun". The results of all tests are gathered in the deployment comparison table above and the corresponding picture for a visual representation is available here.

Main Branch: Latest Chart version.
Downgrade Branch: Last Stable version for compatibility validation.
Upgrade Branch: Same versions as main branch for compatibility check.
Rerun Branch: Evaluates deployment speed with no version changes



Conclusions and Future Work

Helmfile shows to be more effective and adaptable in small, medium, and large-scale Kubernetes installations, improving dependability and deployment times. Helmfile's attraction for easier Kubernetes administration, especially in large-scale deployments, is highlighted by rich insights provided by statistical analysis and graphical representations. Helmfile beats Helm in deployment performance by using locally cached resources, cutting down on the times associated with Helm's sequential approach.

Future endeavors should prioritize enhancing compatibility, scalability testing, CI/CD integration, security fortification, and user experience. These cooperative projects are essential to Helm and Helmfile's ongoing development and relevance in the changing Kubernetes orchestration scene. More specifically, it is critical to prioritise compatibility with changing Kubernetes versions in order to provide smooth upgrades and feature integration.

QR Code for Recording

Evaluation of Google GCP Object Storage and Microsoft Azure Blob Storage

Melbin Paul
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193268@mytudublin.ie

Introduction

Cloud storage services have emerged as a popular choice for organizations seeking to store and manage their unstructured data effectively. Blob storage and Object storage are two common ways to store and access data in the cloud. **Microsoft Azure Blob Storage** is Microsoft's object storage solution for the cloud. **Google Cloud Platform (GCP) Cloud storage** is Google's object storage solution for the cloud. As organizations and individuals are increasingly rely on cloud storage solutions for storing unstructured data, there is a need to understand and compare the Performance, Cost, Security, and Vendor lock aspects of these two providers.
The scope of this research project is focused on comparing the performance, cost, security, and vendor lock aspects of GCP Object storage and Microsoft Azure Block Blob storage. The primary data source for this research will be the experiments conducted to evaluate the performance and vendor lock aspects. The secondary data sources which include books and online references will also be utilized to support and enhance the research for cost and security aspects.

Vendor Lock

Migration and portability experiment was done using **Azure DataFactory Copy Data tool** and **GCP Transfer Job**. Azure to GCP Migration across two regions took double the time in Europe compared to Americas. File transfers between the two providers took almost same time in Americas. .

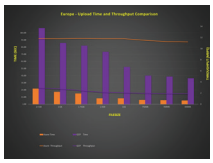
Cost

Cloud providers charge you based on **Storage** Cost, **Data Transfer** Cost and **Data Request** Cost. Azure Cool fares are better in Asia , whereas Archive and Hot is cheapest in Europe. America is cheaper for all tiers in GCP. Standard, Coldline and Archive costs same in Europe and Asia. Nearline is most expensive in Asia. In terms of data operation and retrieval, price vary across different regions in both providers.

Performance

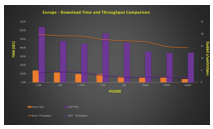
1. Upload and Download:

Azure upload throughputs are higher than GCP for all file sizes with an average of 6 times better throughput in Azure. Throughput to file size and Time to file size coorelations follow similar patterns for both Azure and GCP. Azure continues to perform better on download throughputs too for all filesizes against GCP with an average of 14 time better throughput.



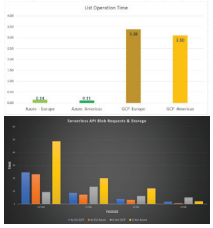
2. List and Concurrent Read/Write Operations:

In 5 runs, to list the 4 files of different sizes, Azure took around less than 15 milliseconds whereas GCP took around 3 secs. Azure takes around 2 secs compared to upto 12-17 secs for GCP for concurrent Write Read operation. Azure performs similar in Europe and Americas , whereas GCP performs better in Americas.



3. Serverless API:

By using **Azure App Service** and **GCP App Engine** serverless features, Azure Europe hosted Serverless API , File stored in Azure is taking similar download time to that in GCP for bigger file size sample whereas Azure taking lesser time than GCP for smaller file sizes. For GCP America hosted Serverless API, File stored in GCP is taking only less than half the time to download than its Azure counterpart for all file sizes.



Security

Azure	GCP
DATA PROTECTION	
• Versioning • Soft Delete • Point in time restore • Blob change feed • Snapshots	• Versioning • Object Holds • Retention policy • Object Lifecycle Policies
BACKUP	
Azure Backup Center • Continuous backups (Operational backup) • Periodic backups(vaulted backup)	• No similar dedicated backup option for an object storage in GCP. • Scheduled backups using scripts or Google Cloud Functions.
BACKUP COST	
No backup storage charges. However, you'll incur the source side cost, associated with Object replication.	No additional fee , but cost increases due to increase in replicating the objects.

Data Backup and Protection: Above summary table illustrates that data protection options in Azure Storage Accounts and GCP Buckets exhibit similarities, albeit expressed through their respective terminologies. From test experience backup option in Azure stands out for its user-friendly interface, offering a more intuitive experience compared to GCP.

Azure		GCP	
Storage Account Level	• Access Key • Shared access signature • Encryption • Access Control(IAM)		
Container Level	• Shared access tokens • Access Policy • Access Control(IAM)	Bucket Level	• Identity and Access management (IAM) • Public Access • Access Control ❖ Uniform
Blob Level	• Shared access tokens	Object Level	• Access Control ❖ Fine-grained
Encryption Types for data at rest	• 256-bit AES Encryption Key(DEK) Service-side encryption (SSE) • DEKs Encrypted with Key Encryption Key (KEK). ❖ Microsoft-managed keys ❖ Customer-managed keys ❖ Customer-provided keys • AES Client-side encryption	Encryption Types for data at rest	• AES-256 Data Encryption Key (DEK). • DEKs Encrypted with Key Encryption Key (KEK). ❖ Google- managed keys ❖ Customer-managed keys ❖ Customer-supplied keys • Client-side encryption
Transport Protocol for data in transit	HTTPS	Transport Protocol for data in transit	HTTPS

Data Security and Access Control: Above Security and Access Control options summary table shows the different security and access control settings for blob data at container/bucket, storage account level in Azure and GCP. Both exhibit a strong dedication to data security, employing comparable encryption strategies.

Conclusions and Future Work

Both GCP and Azure offer reliable and scalable storage options regardless of region. If you prioritizes performance in cloud blob content downloads and uploads from a client application running from cloud virtual machine as a key factor, Azure stands out as a preferred choice regardless of region. Businesses prioritizing efficient and speedy file retrieval may find GCP Americas App engine more favorable as a serverless solution. If cost-effectiveness is a primary consideration, especially storage expenses, GCP proves to be more economical, particularly in the America, where costs are lower across all tiers.
Future work: Performance tests of additional storage tiers offered by the Azure Cool and Archive tiers, GCP Nearline, Coldline and Archive tiers. CPU/Memory, scalability tests can be considered.

QR Code for Recording

Amazon Textract vs Google Document AI vs Azure Document Intelligence.

Ian Bruwer
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00142242@myTUDublin.ie

Introduction

This paper conducts a comparison of Azure AI Document Intelligence, Amazon Textract, and Google Document AI by evaluating price, ease of use, performance, and accuracy. It looks at the mechanisms by which data is collected, from designing a task appropriate form with which to gather hand writing samples, but also discussed the need for redaction of the data being sent for analysis. After sending the data for analysis through the various providers the responses are analysed by comparing actual and expected results using the Ratcliff-Obershelp similarity technique. This measure of accuracy is used to measure how accurately each of the sample documents is captured, but also to compare various sections of the sampled forms to ascertain if certain processors perform better with certain form elements. Ultimately, the paper concludes with a recommendation that if one is already utilising AWS or Azure workloads, then one can comfortably stick with using their respective document processing solutions. However, if starting a greenfield project, overall pricing, ease of use, performance, and accuracy mean that Azure AI Document Intelligence should be your first choice.

Approach

1. Data collection:

A form was design to collect handwriting samples to represent subset of the types of input fields contained in the Irish Naturilsation Process, consisting of various input types. i.e. Text boxes, Check boxes, Tables. The forms were designed to provide expected text such that the research did not need to be concerned with data protection legislation.

2. Data Preparation:

72 Forms where scanned to PDF, with each form also undergoing redaction to ensure that results were not poisoned due to form containing both typed and written versions of the responses.

3. Model Training:

Where appropriate Neural and Template based AI models were trained for each provider; 10 documents used for unredacted documents, 15 documents used for redacted documents.

4. Document Processing:

Primary objective was to compare Generic Pre-trained models for each provider, however, Neural and Template custom models were included in order to ascertain if future research was warranted regarding custom trained models or if generic pre-trained models were adequate for most processing tasks. Documents were sent for processing using REST APIs provided by each provider. This allowed the evaluation of both the performance of the respective APIs, but also for analysis of responses. Though the responses from the providers varied, none were overtly more complicated to process. All results where parsed into consistent structure and were compared with the expected responses using Ratcliff-Obershelp similarity score.

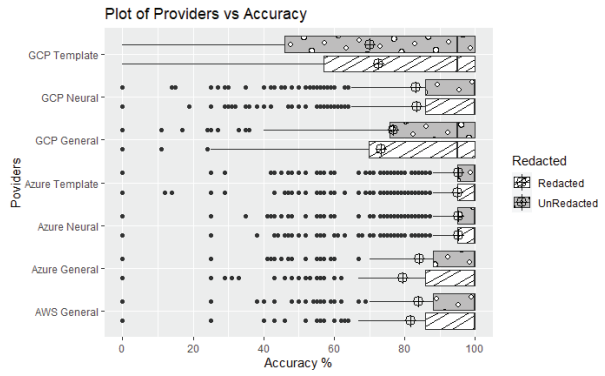
Performance

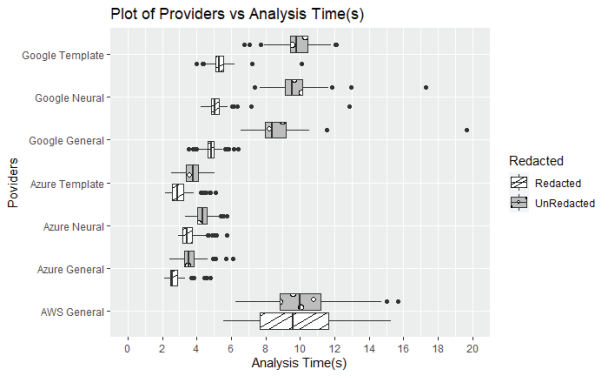
Performance metrics collected for the various providers indicated that Azure is the most performant and consistent processor across all models for both redacted and unredacted documents. Google showed the widest variance in performance between processing of redacted and unredacted documents. AWS and Google processors showed relatively equal processing times for unredacted documents.

Accuracy

When utilising Pre-trained general models, Azure and AWS proved to be the most accurate of the processors with median accuracy of 100% followed closely by Google. The most accurate processsing was deliverd by Azure using Neural and Template custom models, with the Google template custom model showed surprising poor performance.

Results





Conclusions and Future Work

The research shows that if you are looking to perform document analysis, the choice of either Azure AI Document Intelligence or Amazon Textract are suitable option if embedding into an existing Azure or Amazon eco system respectively, however, if only just starting out, the accuracy, cost. and ease of use of the Azure AI Document Intelligence makes it the prime candidate for new endeavours. Future work needs to focus on an in depth comparison of the Neural and Template models provided by the providers in order to ascertain if Google can achieve parity with Azure if provided with enough training data. Generative AI is another area that needs to be investigate in this context as the providers have mechanisms to integrate natural lanuage queries into document analysis.

QR Code for Recording

Comparative Analysis of Google Cloud Deployment Manager and Terraform

Brian Ryan
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193204@myTUDublin.ie

Introduction

This research project examined Terraform and Google Cloud Deployment Manager in modern IT orchestration. It was an evaluation mission, placing the two side-by-side to compare capabilities for organisations navigating cloud options. A research case study was completed where a simple but very effective cloud architecture was created using Terraform and Google Cloud Deployment Manager. The researcher created a survey to enhance the research, and a group of experts in the field answered it. The survey results were dissected and examined to further enlighten the reasearch about the nuances of these IaC deployment tools.

Research Question

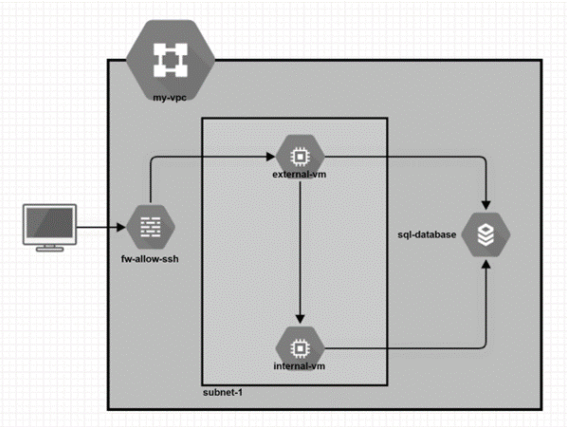
With the search for innovative ways to handle the infrastructure that software is built on, and due to cloud computing becoming the norm for delivering software, the research question arises:

"How do Google Cloud Deployment Manager and Terraform differ in performance, features, and community support, and what insights can be obtained from a survey of developers to inform selecting either Google Cloud Deployment Manager or Terraform as the IaC tool of choice?"

IaC Tool Performance

Speed is everything when getting resources up and running in the cloud. The performance metrics for Google Cloud Deployment Manager and Terraform were tracked and analysed. The average build times are close, but Terraform destroys infrastructure more consistently every time - the range of times it takes is much smaller. With Deployment Manager, some deletes happen faster, and some take longer. We can see from the results that there is no considerable difference in build times. However, the time it takes to delete could make a difference when selecting a tool.

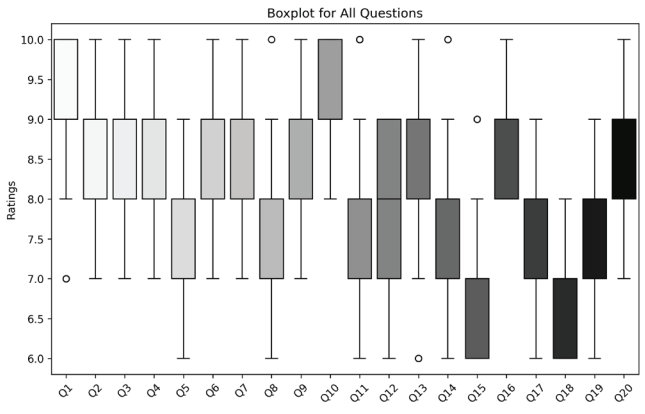
Case Study



Architecture Definition

The architecture is a baseline cloud architecture demonstrating some key IaC at play. The architecture, which includes a VPC, subnet, two compute instances, a database, a database instance, and a firewall rule to access the instances outside the VPC securely. These are the core resources in any cloud app or service, networking, compute, storage, and security. It is not complex, but it is enough to showcase some cloud capabilities in this research project. This architecture could also be extended based on many future complex use cases.

Survey Responses



Survey Results Comparative Analysis

Terraform has an edge over Google Cloud Deployment Manager in most critical areas, particularly infrastructure as code support and multi/hybrid cloud support. The broader responses for Deployment Manager could be due to users having more variation with cloud experience, which might mean different use cases or a project tailored for specific functionality.
Fundamental concepts, such as handling large complex deployments and using templates and modules, are almost on par for both tools. Users responded very similarly to these questions. Terraform seems better for projects focused on solid IaC management and multi/hybrid cloud deployment support. Meanwhile, Google Cloud Deployment Manager is boosted through tight integration with GCP.

Conclusions and Future Work

For organisations invested in GCP, unless a transformation to hybrid cloud is planned, there is no benefit of moving away from Deployment Manager. Terraform is the all-round utility king for managing cloud services. Its primary benefit, which ensures it stands out from the crowd, is the ability to work with all the major cloud platforms. The research shows that each tool differs slightly in each domain, and Terraform comes out on top in its performance, features, and the greater community support available. In the authors opinion the research has shown that Terraform has risen above Google Cloud Deployment Manager as the Infrastructure as Code tool of choice. Future research in Infrastructure as Code for Google Cloud Deployment Manager and Terraform could include studies on incorporating new features and investigating how each tool handles emerging technologies and integrates with the industry enhancements that will come.

QR Code for Recording

ANALYSIS OF IMAGE SECURITY VULNERABILITIES OVER TIME.

AUTHOR: DARRAGH CLERKIN

School Of Enterprise Computing and Digital Transformation, TU Dublin, Ireland

X00193261@myTUDublin.ie

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

INTRODUCTION

A container image is a standalone executable piece of software that includes everything needed to run the application. This includes code, system libraries, runtimes and settings.
With the rise in containerized environments and images, there comes security risks, image vulnerabilities which include vulnerabilities in their dependencies can leave applications both privately and publicly used open to being exploited by bad actors. This can lead to sensitive data about users being used or sold to 3rd parties intending harm.
The motive for this thesis is the need for research into the number of vulnerabilities present in many of the top Linux images on Dockerhub.

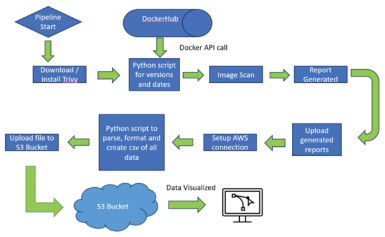
RESEARCH QUESTIONS

Question 1: Do we know when the best time/ version to take after a major release?
Hypothesis 1: It would be assumed the later into minor versions would be better to take an image realistically if you are trying to avoid vulnerabilities in the image used. This would come with downsides like not having the newest major features available to you until months after others potentially disadvantaging you or your company.
Question 2: Does the number of vulnerabilities decrease or increase over time in images?
Hypothesis 2: The number of vulnerabilities will decrease over time but increase during major releases. This increase is due to major updates in the software which may be rushed to get to the customer. It would be expected that the number of vulnerabilities will decrease in the minor releases and issues are fixed and new content is delivered. The number will be dynamic is certain instances also as new vulnerabilities are discovered they may be fixed in the next major or minor release based on their severity it would be assumed.

METHODOLOGY

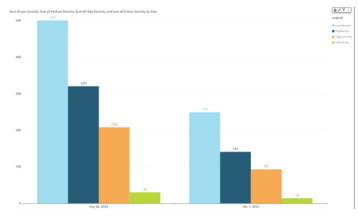
Primary Research was carried out by the way project. Dockerhub Api was used to gather version data and dates last updated and Trivy image scanner was used to scan images for vulnerabilities. This was carried out over 3 months.
Secondary Research by way of a Literature review was carried out. This took papers and articles published in the last 3 - 4 years on the topic of image security and vulnerabilities.

PROJECT ARCHITECTURE

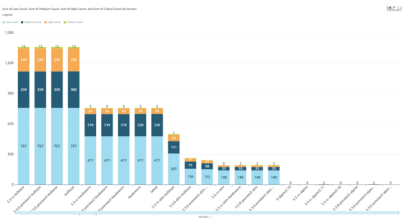


RESULTS

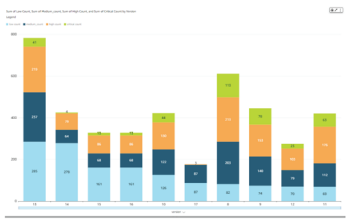
Plone Results



Ruby Results

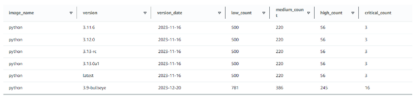


Odoo Results



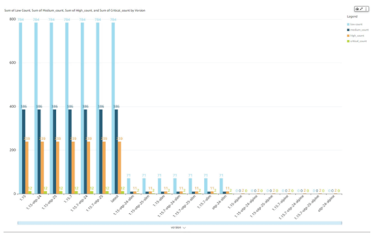
In the cases of Ruby, Plone and Odoo and others present in the study we see the vulnerabilities are decreasing over time. We can see in Ruby that the slim and alpine versions have the lowest number of vulnerabilities present

Python Results



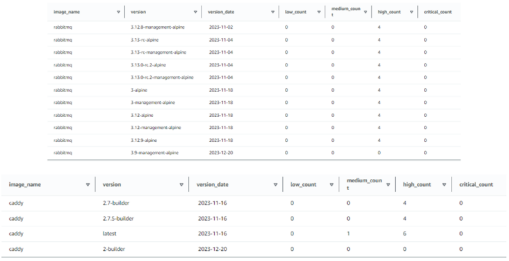
Python like many others contains a bulleye version which is built using debian 11 (oldstable). It can been seen that these versions tend to contain more vulnerabilities in them than the bookworm versions which are debian 12 which are the current (stable) versions.

Elixir Results



In the Elixir example we can see that the between versions 1.15 , 1.15.7 and latest there no change in the number of low, medium, high and critical risk vulnerabilities.

Caddy and RabbitMq Results



14 out of the 50 images scanned have little to no vulnerabilities present in them, 2 examples of this are rabbitmq and caddy which can be seen above

CONCLUSION

- 14 / 50 Images scanned had little / no vulnerabilities detected.
- For those that have data for multiple versions, the vulnerabilities are decreasing over time.
- If alpine versions are available they should be taken unless the latest stable versions are required
- Inconclusive on whether major or minor versions are best to take.

QR Code for Recording

FUTURE WORK

- Longer time frame for the project
- Larger data set > 50 images
- Scan other types of images (Windows images)
- Use different Image Vulnerability scanners
- Gather more types of Data from Analysis
- Use Data Analytics to make more accurate conclusions

Navigating Quantum Realities: A Comprehensive Analysis of Quantum Computers, Providers, and Qiskit Compatibility Challenges and Opportunities

Weverton de Souza Castanho
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00193205@myTUDublin.ie

Introduction

Analysis of the degree of availability, capacity, and compatibility of quantum infrastructure service providers and types of quantum computers running a compatible application to assess the maturity level of available services and infrastructure. Since quantum computers are a promise for the present and future, it is necessary to analyze their compatibility, identify the best development framework that can run on all of them, preserving the investment applied in code..

The renowned scientist Richard Feynman was one of the pioneers in recognizing the importance of quantum computing and understanding that nature could resemble a vast quantum computer. In his paper "Quantum mechanical computers" (1986), he laid the initial foundations for the next step: quantum computing. Although quantum computers are a reality today, the question that remains is how far we can advance with them and what features and compatibilities among them can drive scientific and human progress.

Quantum Photosynthesis

Photosynthesis is a vital process for plants and life on the planet. However, its efficiency, approaching 100%, cannot be fully explained by classical physics. One explanation lies in the use of quantum processes within plant leaves. A simulation based on a quantum circuit was developed and tested on various computers to assess the maturity of Qiskit on these platforms.

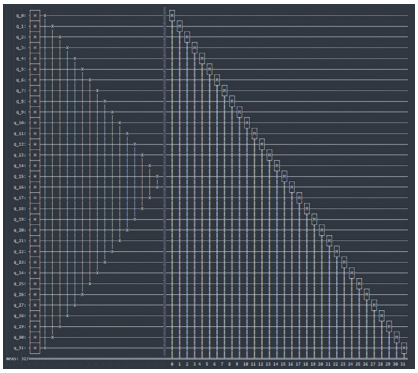
Quantum Bitcoin Mining

The Bitcoin attracts significant investments from banks and companies as a substantial store of value. However, the energy consumption in processing the currency is equivalent to that of some countries. Quantum computers provide a more efficient solution, consuming less energy and processing cryptography rapidly through algorithms designed for this purpose. An address generator was tested using the Grover algorithm on various quantum computing platforms.

Quantum Circuits

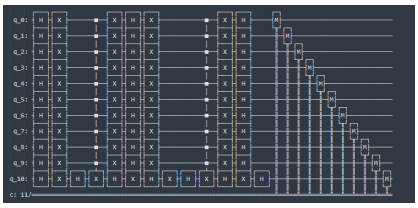
1. Quantum Photosynthesis Simulation:

The quantum circuit was implemented using the Quantum Fourier Transform (QFT), for which the SWAP command was employed as a simplification of the QFT.



2. Bitcoin Simulation:

This code is a basic implementation of the Grover's algorithm for a fictional Bitcoin address search scenario. Grover's algorithm is particularly powerful for unstructured search problems, providing a significant quantum advantage over classical algorithms.



Topic Overview

The thesis aims to analyze the availability, capacity, and compatibility of quantum infrastructure service providers and types of quantum computers to assess the maturity of currently available services and infrastructure. The importance of choosing a development framework compatible with various providers is emphasized, with IBM's Qiskit being the initial choice. Despite testing on various quantum computers in the market, the availability of these machines still does not fully meet the demands of an emerging market seeking to understand and leverage the advantages of this new technology. Several tests could not be completed due to limitations in containers running Jupyter IDEs, presenting significant memory constraints. Additionally, the dependence of quantum computers on classical computers for data input and output is highlighted. More complex constructions, such as Shor and Grover algorithms, often result in failures due to memory overflows.

Conclusions and Future Work

The Qiskit language, a promising tool for the future, opens up possibilities for compatibility among various quantum processor architectures. However, there is still no complete integration among the different available quantum computers. The shortage of machines for testing complicates the debugging and improvement process for Qiskit applications, especially when migrating to other quantum platforms. While the opportunities that quantum computing can offer to humanity are significant, the journey for those embarking on it is challenging, filled with efforts and incremental results.

Looking ahead, it would be interesting for universities to engage in creating a quantum computer model based on Nitrogen Vacancy (NV) Diamonds. Companies in the market already offer these synthetic diamonds for use in quantum computing. NV Diamonds enable the construction of quantum computers at room temperature, are less susceptible to noise, consume less energy, take up little space, and could be the key to developing an open-source computer. This would open up broad opportunities for advancements in the field of quantum computing within educational institutions.

QR Code for Recording

Effectiveness of Microservice and Token based Security access control method.

Emanual Alby - X00193250@myTuDublin.ie
School of Enterprise computing and Digital transformation, TU Dublin, Ireland
Supervisor: Dr. John Burns

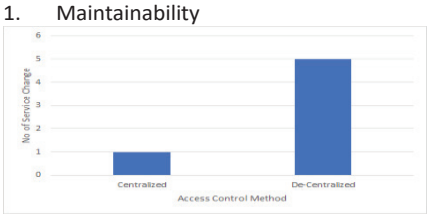
Introduction

Microservices architecture is an architectural style that divides an application into small, independent services that are loosely coupled and deployed independently. Each service is typically organised around a specific business capability. Though there are significant advantages using microservices architecture, it increases the security risk complexities. Due to the distributed nature of the application, to manage the access control is an additional exercise and to maintained for each distributed service. There are two types of access control approaches namely Centralized and de-centralized access control architecture patterns.

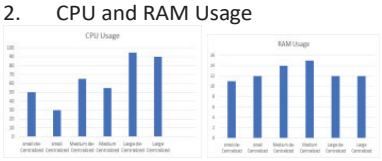
- RQ1 - Whether Centralised Access control pattern is preferable over De-Centralised Access control pattern.
- RQ2 -. Will external access controls increase the efficiency of the application.
- RQ3 - Whether microservices pattern more advantages over the legacy architecture pattern.

Centralized and De-Centralised

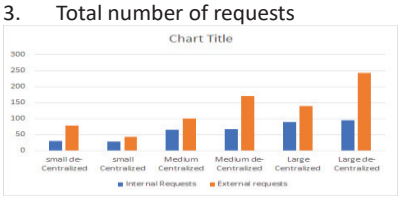
1. Maintainability



2. CPU and RAM Usage

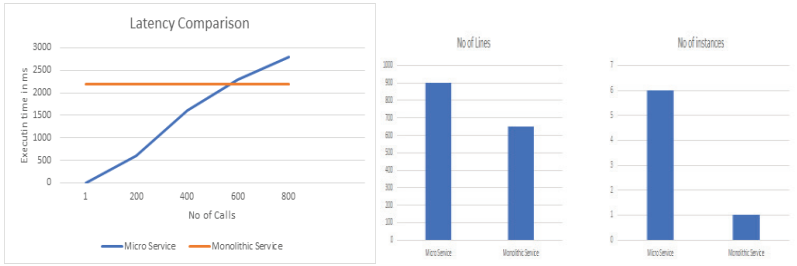


3. Total number of requests

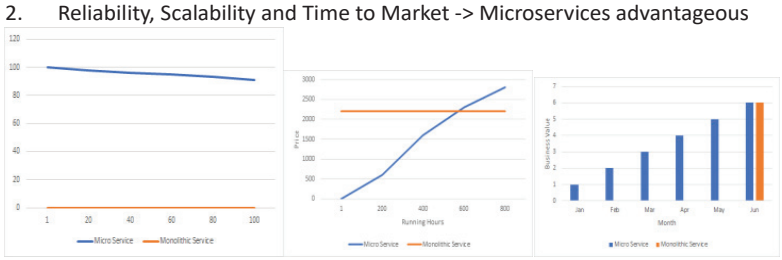


Advantages of Microservices over monolithic architect pattern

1. Latency & Complexity -> Monolithic advantageous



2. Reliability, Scalability and Time to Market -> Microservices advantageous



Topic overview

RQ1 -> Whether Centralised Access control pattern is preferable over De-Centralised Access control pattern.

- Experimental results denote security of the application is increased using De-Centralised access control patter due the distributed nature. Single point of failure can be avoided using de-centralised access control pattern.
- Applications with Central access control can be easily maintained using centralised access control pattern.
- Complexity of the application reduces using centralised access control pattern.

RQ2 -> Will external access controls increase the efficiency of the application.

- Externalizing the access control increase the application security as the resource details will not be exposed to outside world.

RQ3 -> Whether microservices pattern more advantages over the legacy architecture pattern.

- Latency and complexity denote monolithic pattern is advantageous.
- Reliability, Scalability and Time to Market parameters denote microservices is advantageous

Conclusions and Future work

Conclusion:

- De-Centralised access control is preferred over Centralized access control as this increase the security scalability, granularity of the security implementation.
- Access control using external token increase the system security.
- Microservices is preferred over Monolithic.

Future work

- The load test that is used can be extended to include more complex load test scenarios.
- We have a used a set of microservices for this experiment using spring boot and can use other technologies.

QR code for recording