Title:	Scalable Microservices APPROVED
Long Title:	Scalable Microservices

Module Code:	SOFT8025	**Duration:**	1 Semester

Credits:	5
NFQ Level:	Advanced
Field of Study:	Computer Software
Valid From:	Semester 2 - 2020/21 (January 2021)
Module Delivered in	4 programme(s)
Module Coordinator:	Sean McSweeney
Module Author:	Paul Davern
Module Description:	Microservices are singular in terms of their responsibility and can be independently scaled, tested and deployed. Services developed using this approach are built around business capabilities using best practices from domain-driven-design and are autonomous and isolated. In this module the student will learn to build, automate, scale and manage a distributed system created from a number of collaborating components represented as microservices.

Learning Outcomes	
On successful completion of this module the learner will be able to:	
LO1	Apply a framework technology to develop a microservices based application.
LO2	Design a distributed application composed of multiple collaborating services in the form of microservices.
LO3	Develop microservices in a distributed environment.
LO4	Interpret and compare microservice design architectures.
LO5	Design the deployment, scaling and management of microservice based applications.

Pre-requisite learning

Module Recommendations
This is prior learning (or a practical skill) that is strongly recommended before enrolment in this module. You may enrol in this module if you have not acquired the recommended learning but you will have considerable difficulty in passing (i.e. achieving the learning outcomes of) the module. While the prior learning is expressed as named CIT module(s) it also allows for learning (in another module or modules) which is equivalent to the learning specified in the named module(s).

Incompatible Modules
These are modules which have learning outcomes that are too similar to the learning outcomes of this module. You may not earn additional credit for the same learning and therefore you may not enrol in this module if you have successfully completed any modules in the incompatible list.

No incompatible modules listed

Co-requisite Modules

No Co-requisite modules listed

Requirements
This is prior learning (or a practical skill) that is mandatory before enrolment in this module is allowed. You may not enrol on this module if you have not acquired the learning specified in this section.

No requirements listed

Module Content & Assessment

Indicative Content
Core Concepts History of microservices - Service Oriented Architectures (SOA), Web Services, REST, Microservices. Monolithic service architecture versus micro-service architecture; Microservice characteristics. Overview of microservice development frameworks, technologies and programming languages; Single Microservice design. Building microservices using RESTful API and Web Services. Scalability of REST; Logging; Concurrency Control; Asynchronous and synchronous communication; event loops; Testability; Security; Runtime environments.
Developing Microservices Building a microservices application. Service Registration and discovery process. Load balancing across services and Inter-process communication in the Cloud. Server side load balancing and Client side load balancing. Messaging: HTTP, grpc. Message bus: scalability and performance.
Microservice design and distributed systems. Patterns for distributed sessions and distributed transaction management: Event sourcing, CQRS, Saga, state machines. Microservices and the CAP theorem. Reliable distributed coordination: RAFT, etcd, zookeeper. Logging. Concurrency issues. Fault tolerance in microservices.
Microservice architecture and scalability Lightweight containerisation – compare to traditional server virtualisation, approaches, benefits and implementations e.g. Docker. Linux kernel infrastructure for building containers. Security issues with containerisation; Unikernels and microVMs; Function based services: design, security. Automating deployment, scaling and management of containerised applications e.g. kubernetes.

Assessment Breakdown	%
Course Work	100.00%

Course Work

Assessment Type	*Assessment Description*	*Outcome addressed*	*% of total*	*Assessment Date*
Practical/Skills Evaluation	Mini-projects based on lectures, which will re-enforce the students learning. For example: build a microservices application.	1,2,3	30.0	Every Week
Written Report	Technical report on a topic related to microservices design.	2,3,4	10.0	Week 9
Essay	Essay on the architecture of microservices based applications with a distributed systems focus.	2,4,5	20.0	Week 12
Project	The purpose of this assessment is to design and develop an application composed of micro-services in a distributed environment.	1,2,3,5	40.0	Sem End

No End of Module Formal Examination

Reassessment Requirement
Coursework Only *This module is reassessed solely on the basis of re-submitted coursework. There is no repeat written examination.*

The institute reserves the right to alter the nature and timings of assessment

Module Workload

Workload: Full Time				
Workload Type	*Workload Description*	*Hours*	*Frequency*	*Average Weekly Learner Workload*
Lecture	Lecture underpinning learning outcomes.	2.0	Every Week	2.00
Lab	Lab supporting content delivered.	2.0	Every Week	2.00
Independent & Directed Learning (Non-contact)	Independent learning & study.	3.0	Every Week	3.00
			Total Hours	7.00
			Total Weekly Learner Workload	7.00
			Total Weekly Contact Hours	4.00

Workload: Part Time				
Workload Type	*Workload Description*	*Hours*	*Frequency*	*Average Weekly Learner Workload*
Lecture	Lecture underpinning learning outcomes.	2.0	Every Week	2.00
Lab	Lab supporting content delivered.	2.0	Every Week	2.00
Independent & Directed Learning (Non-contact)	Independent learning & study.	3.0	Every Week	3.00
			Total Hours	7.00
			Total Weekly Learner Workload	7.00
			Total Weekly Contact Hours	4.00

Module Resources

Recommended Book Resources

- **Chris Richardson 2018, *Microservices Patterns: With examples in Java*, Manning Publications [ISBN: 978-161729454]**

Supplementary Book Resources

- **Denis Kolodin 2019, *Hands-On Microservices with Rust: Build, test, and deploy scalable and reactive microservices with Rust*, Packt Publishing [ISBN: 978178934275]**
- **S. Newman 2019, *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*, O'Reilly Media [ISBN: 978149204784]**

Recommended Article/Paper Resources

- **Agache, Alexandru, et al. 2020, *Firecracker: Lightweight Virtualization for Serverless Applications.*, 7th {USENIX} Symposium on Networked Systems Design and Implementation NSDI 20**
- **Ongaro, Diego, and John K. Ousterhout. 2014, *In Search of an Understandable Consensus Algorithm*, USENIX Annual Technical Conference**
- **Kreps, Jay, Neha Narkhede, and Jun Rao. 2011, *Kafka: A distributed messaging system for log processing.*, Proceedings of the NetDB**
- **Howard, Heidi, et al. 2015, *Raft refloated: Do we have consensus?*, ACM SIGOPS Operating Systems Review**
- **Dragoni, Nicola, et al. 2017, *Microservices: yesterday, today, and tomorrow.*, Present and ulterior software engineering. Springer.**
- **Combe, Theo, Antony Martin, and Roberto Di Pietro. 2016, *To docker or not to docker: A security perspective.*, IEEE Cloud Computing 3.5**

Other Resources

- **Website: *Microservices***
  http://www.martinfowler.com/articles/mic roservices.html
- **Website: *Cloud Design Patterns***
  https://docs.microsoft.com/en-us/azure/a rchitecture/patterns/
- **Website: *Patterns for Distributed Systems***
  https://martinfowler.com/articles/patter ns-of-distributed-systems/

Module Delivered in

Programme Code	Programme	Semester	Delivery
CR_KDNET_8	Bachelor of Science (Honours) in Computer Systems	7	Mandatory
CR_KCLDC_9	Master of Science in Cloud Computing	1	Elective
CR_KCLDC_9	Master of Science in Cloud Computing	2	Elective
CR_KSADE_9	Master of Science in Software Architecture & Design	1	Mandatory