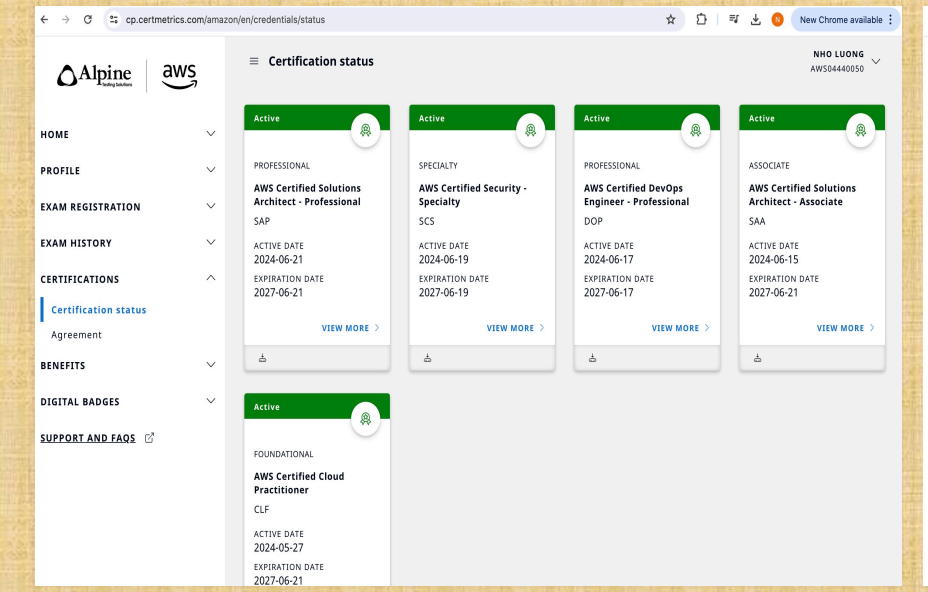
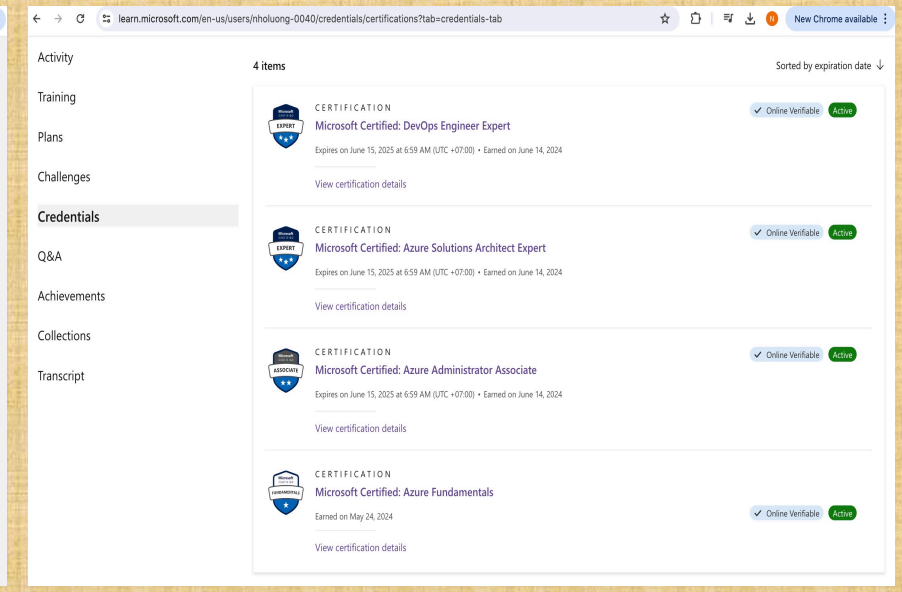
Test Driven Development

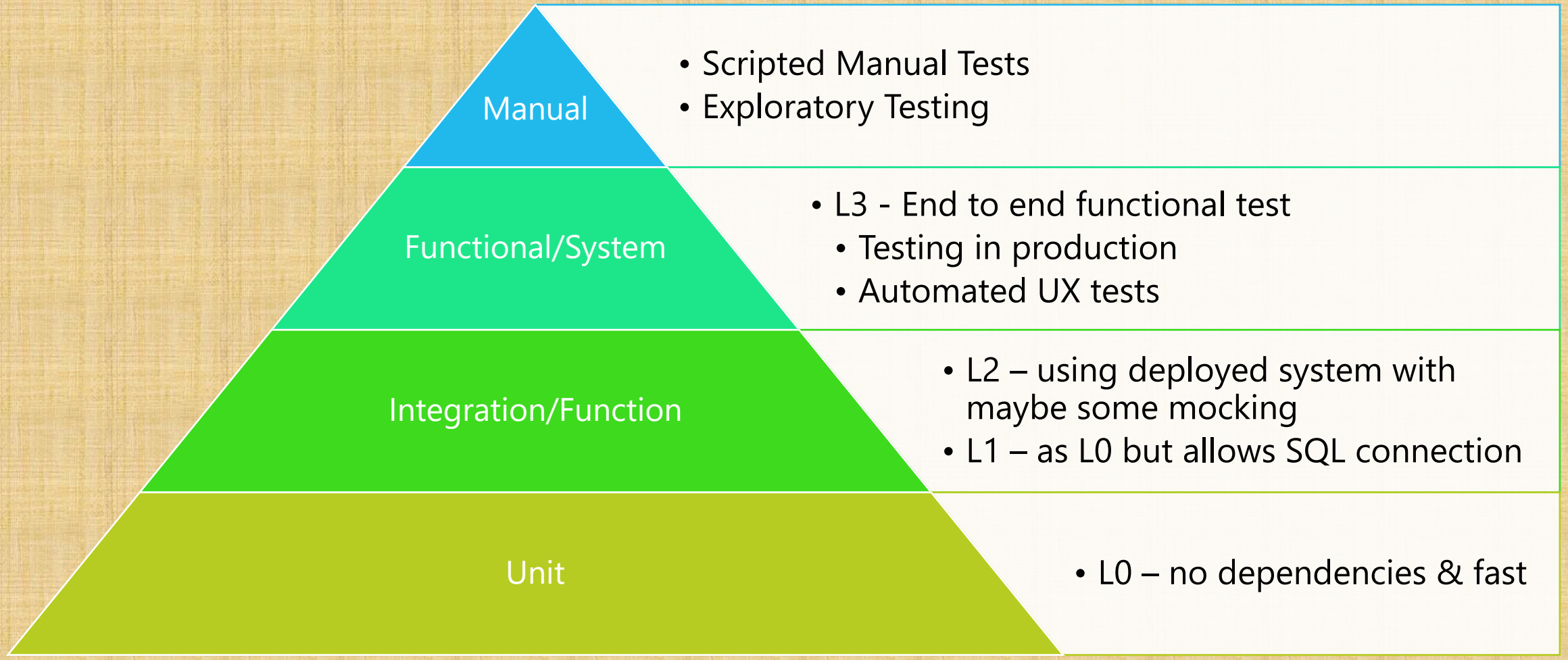
Author: Nho Luong

Skill: DevOps Engineer Lead





Testing Pyramid



Author: Nho Luong
Skill: DevOps Engineer Lead

Test Principles

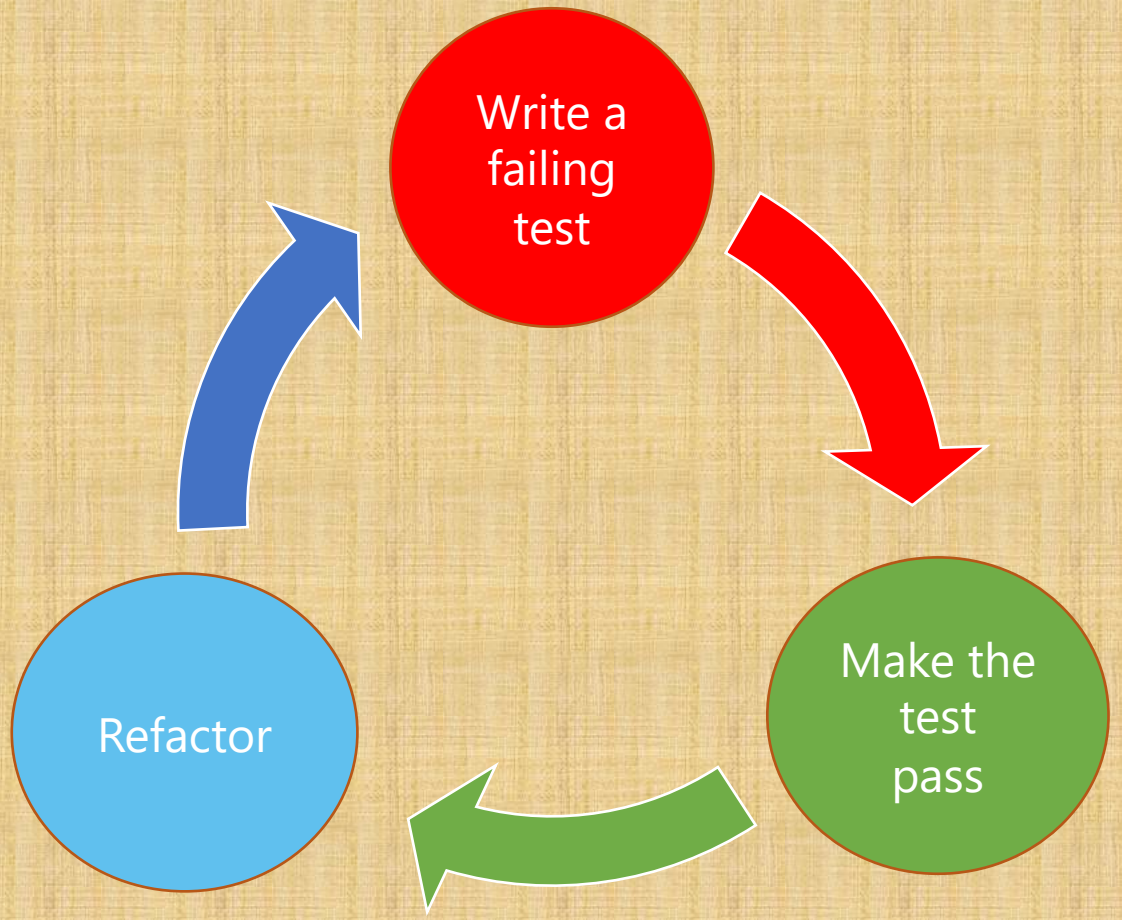
- Tests should be written at the lowest level possible
- Tests written once, run anywhere including production system
- Product is designed for testability
- Test code is product code, only reliable tests survive
- Testing infrastructure is a shared Service
- Test ownership follows product ownership

Author: Nho Luong
Skill: DevOps Engineer Lead

How do I created my Unit Tests?

Author: Nho Luong
Skill: DevOps Engineer Lead

The Test Driven Development Cycle



Author: Nho Luong
Skill: DevOps Engineer Lead

Demo
TDD Kata

https://osherove.com/tdd-kata-1

1. An empty string returns zero
2. A single number returns the value
3. Two numbers, comma delimited, returns the sum
4. Two numbers, newline delimited, returns the sum
5. Three numbers, delimited either way, returns the sum
6. Negative numbers throw an exception

Author: Nho Luong
Skill: DevOps Engineer Lead

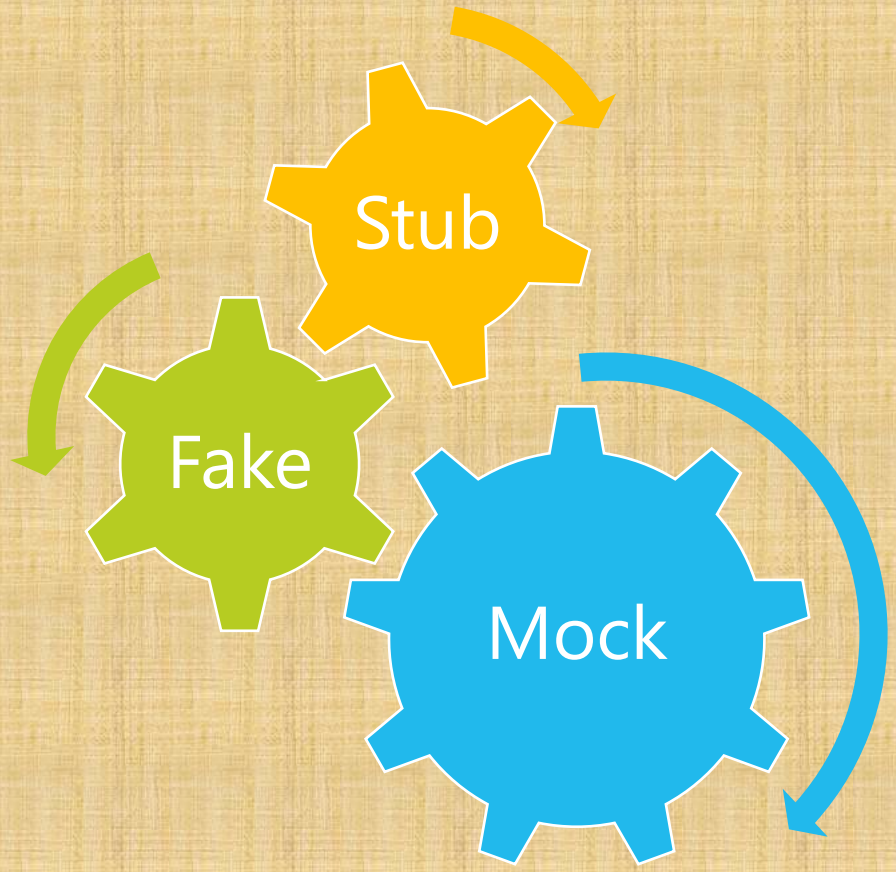
So with TDD we get…

- To write the test before the code
- Emergent design of architecture
- Excellent test coverage

But what happens when we need systems that are not available to the developer?

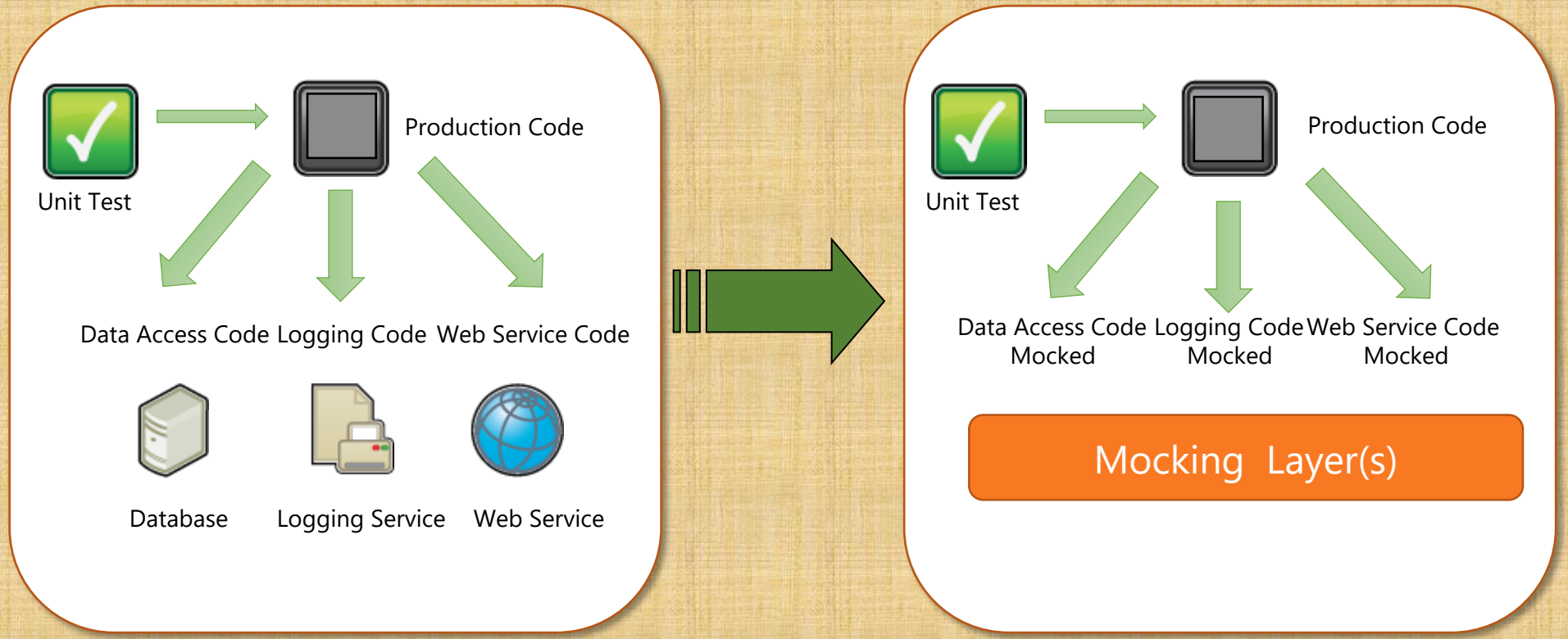
Author: Nho Luong
Skill: DevOps Engineer Lead

What is Mocking



Author: Nho Luong
Skill: DevOps Engineer Lead

Mocking the 'Principal of Isolation'



Author: Nho Luong
Skill: DevOps Engineer Lead

Demo
Mocking with Moq

Author: Nho Luong
Skill: DevOps Engineer Lead

Frameworks for common languages*

Language	Testing Framework	Mocking Framework
.NET	MSTest, xUnit, NUnit	Moq, NMock
Java	JUnit	Mockito, EasyMock
JavaScript, TypeScript, Node.JS	Jest, Mocha, Jasmine	Mocha, Jasmine
C++	CppUnit, Google Test, Boost	CppUnit, Cmock
Python	PyTest, PyUnit	PSBoundParameters

* a far from exhaustive list

Author: Nho Luong
Skill: DevOps Engineer Lead

But what is the argument against TDD?

Pros.

- Only write code needed
- Easier to maintain/refactor
- High test coverage
- Tests are the documentation

Cons.

- Not a silver bullet
- Can be slow to develop
- Usually needs a lot of mocking
- All the team has to adopt
- Tests must be maintained

Author: Nho Luong
Skill: DevOps Engineer Lead

Summary

- TDD is a way to get high code coverage
- High code coverage makes refactoring safer
- Mocking is a key tool to enable TDD
- Practice your skills with Coding Katas

Author: Nho Luong
Skill: DevOps Engineer Lead

Azure

Azure DevOps

Azure Integration Services

android

GitHub

Windows

Microsoft HoloLens

Author: Nho Luong
Skill: DevOps Engineer Lead

Thank You

Author: Nho Luong
Skill: DevOps Engineer Lead