AUTOMATED TESTING

Why do we need it?

OUTLINE

MODERN SOFTWARE PRACTICES

THE NEED FOR AUTOMATION

HOW TO AUTOMATE?

EXAMPLE + DEMO

MODERN SOFTWARE PRACTICES

When you look at the way how very successful companies build their software, there are a set of common patterns and practices.

MODERN SOFTWARE PRACTICES

Everything is version controlled

Continuous integration

Infrastructure as code

Automate everything

Continuous delivery

Configuration as code

MODERN SOFTWARE PRACTICES

Everything is version controlled

Continuous integration

Infrastructure as code

Automate everything

Continuous delivery

Configuration as code

EVERYTHING IS VERSION CONTROLLED

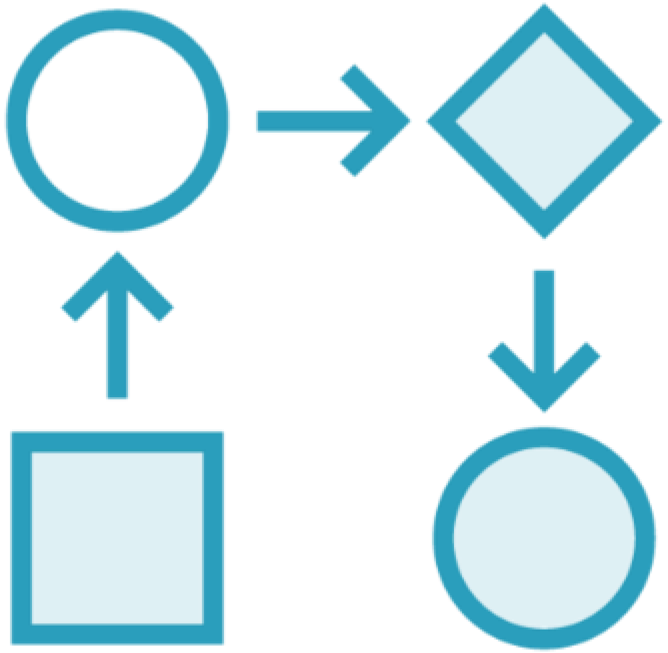
Put everything under version control and I mean everything!
Not just the source code!

- Specification
- Scripts to create a new development environment
- Scripts used to build the software
- Scripts to create Schema in a database

AUTOMATE EVERYTHING

Strive towards the goal of automating everything:

- We get speed from automation
- We get consistency and repeatability
- Get less dependant on who are the developers in the team
- Anyone in the team can do the work because regressions are caught sooner
- Automation acts as documentation to spread knowledge



CONTINUOUS INTEGRATION

Important to get feedback on the work we have done as soon as possible:

- Can our changes be integrated without introducing problems?
- Aim to have changes integrated into the product as soon as possible
- See if there are any failing tests based with the current changes

THE NEED FOR AUTOMATION

Speed of delivery of software has become much faster in many cases and we need to continuously test these changes

AGILE PROCESS

Requirements gathering

Writing software

Testing software

REQUIREMENTS GATHERING

- Gather requirements
- Write a functional specification
- Generate user stories

WRITING SOFTWARE

In an ideal world:

- Write unit tests
- Implement functionality
- Continuous integration guards against changes that potentially break the software.

The world is not ideal:

- Unit test coverage is often low
- Often a consequence of maintaining software written many years ago
- Retrofitting unit tests is not a simple task

TESTING

- Testers install the software
- Run their tests scripts which are most of the time documents
- Often a manual process which can take between days, weeks and months of work.
- Often tasks are repetitive

SPEED OF DELIVERY

- Software delivery speed has improved in the last decade:
  - Agile methodologies put an emphasis on fast continuous delivery to production
  - IDEs have become smarter which allows us to write code faster
  - Infrastructure can now be provisioned on demand in seconds with cloud

MANUAL TESTING TAKES A LONG TIME

If you look at the state of delivery and compare it to the past:

Manual testing does not take longer than it used to.

- The frequency of needing to manual test code changes has increased.

This often leads to compromises to increase the speed of software delivery:

- Not manually testing all changes
- Infrequently running the full suite of manual tests

This inevitably leads to unfound regressions and long stretches stuck in UAT as issues are found late.

HOW DO WE SPEED UP?

- Speed is everything!
- Early feedback is essential!
- Manual testing provides feedback days after implementation
- Need to deliver multiple times a day
- How to speed up?
  - Find alternative ways to validate if your software conforms to specification
  - Catch obvious issues like crashes before handing over to test
  - Test automation, replace ourselves as testers and let computers do it for us

HOW TO AUTOMATE?

What tools and practices do we need to follow to automate?

AUTOMATION TOOLS

cucumber

specflow

cucumber for .net

appium

cucumber

PROVIDES TOOLS THAT ENABLE:

Collaboration

Behaviour Driven Development

Automation

Living Documentation

Gherkin Syntax

COLLABORATION

- **Enable frictionless collaboration**
  - Empower the whole team to read and refine executable specifications without needing technical tools.
- **Single Source of Truth**
  - Git integration means your BDD documentation is always up to date
  - Stop sending documents around, and worrying if everyone has the latest version

BEHAVIOUR DRIVEN DEVELOPMENT

Behaviour-Driven Development (BDD) is revolutionizing the way people build software.

- Develop collaborative relationships between business and technical people by working together to develop executable specifications.
- Break down big problems into small ones using the power of examples.
- Use test automation to guide development and eliminate bugs.

AUTOMATION

Accelerate development using BDD specifications that double as automated tests.

- **Implement Agile Test Management:**
  - Automated tests build confidence and trust across the team. Give all stakeholders visibility into testing activity and results with analytics built for modern agile organizations

LIVING DOCUMENTATION

- **Shared Understanding:**
  - Describe how the system should behave in a way that everybody can understand
- **Documentation is never out of date:**
  - If the behaviour of a feature changes then the living specification needs changing otherwise the BDD tests will fail.
- **Publish Living documentation**
  - See your team’s feature files rendered as beautiful, easily understood documentation, automatically verified with every developer check-in

GHERKIN SYNTAX

```
# Comment
@tag
Feature: Eating too many cucumbers may not be good for you

  Eating too much of anything may not be good for you.

  Scenario: Eating a few is no problem
    Given Alice is hungry
    When she eats 3 cucumbers
    Then she will be full
```

Gherkin uses a set of special keywords to give structure and meaning to executable specifications.

specflow

cucumber for .net

What is SpecFlow?

- SpecFlow is the #1 .NET open source framework for Behavior Driven Development, Acceptance Test Driven Development and Specification by Example. With over 10m downloads on NuGet, SpecFlow is trusted by teams around the world

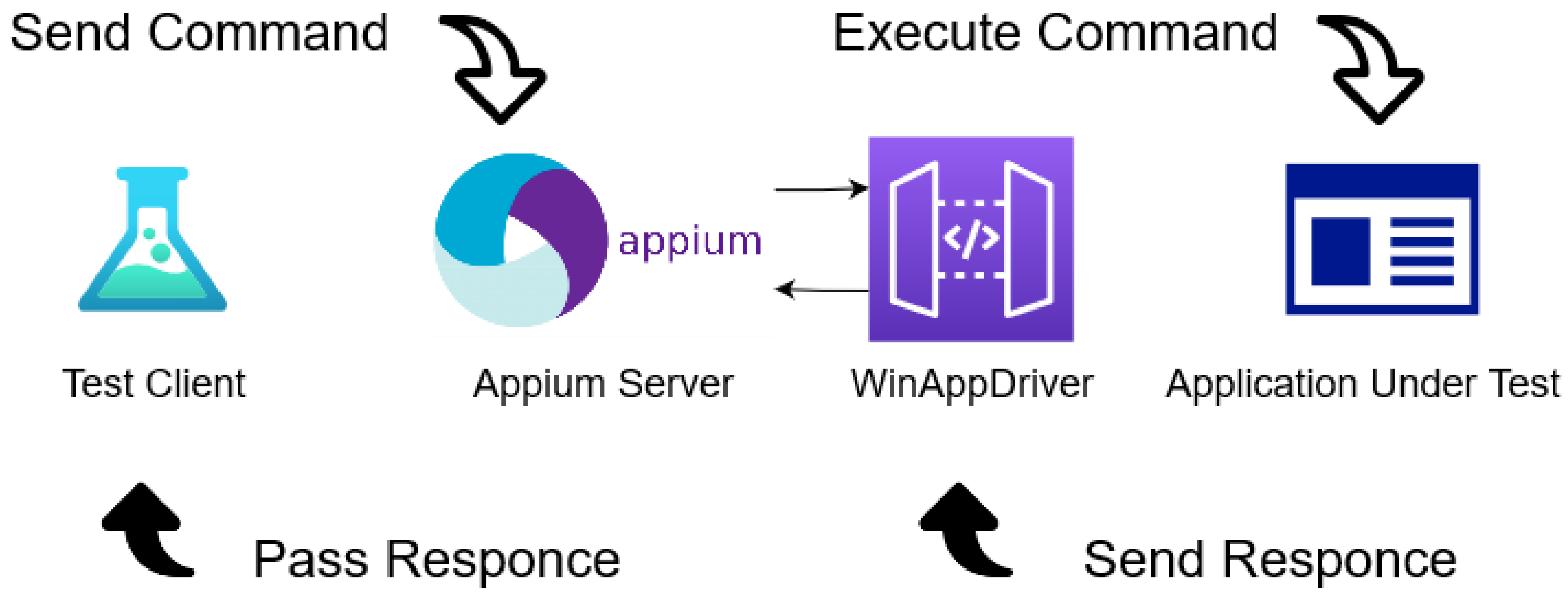
What is Appium?

- Appium is an open source test automation framework for use with native, hybrid and mobile web apps.

WinAppDriver

- Is a test framework developed by Microsoft
- Is an implementation of Appium, Itself based on Selenium
- WinAppDriver is a Selenium-like automation framework

WINAPPDRIVER



EXAMPLE + DEMO

Example of existing UAT tests converted over to Gherkin and demo of running the automated tests.

DEMO

QUESTIONS

Thank you for listening!