



**Corndel  
Digital.**

in association with

**Softwire**

## Workshop: Module 07

Continuous Integration

# Agenda

## 1000 Welcome and Introductions

### Part 1: GitHub Actions

- Recap (5 mins)
- New Stuff (5 mins)
- Exercise – GitHub Actions (110 mins)

## 1200 Lunch Break (1 hour)

## 1300 Part 2: Jenkins

- Recap (5 mins)
- Exercise – Jenkins (150 mins)
- Discussion

# Part 1

GitHub Actions

# Recap

## Continuous Integration

- Automated build and test
- Can prevent broken changes from being merged
- Makes it easy to spot when something is broken
- Often integrated with notification systems

# New Stuff

## GitHub Actions

- Easy to set up
- Used to run actions on *runners* (essentially VMs hosted by GitHub)
- Integrated with GitHub's pull requests
- Free for public repositories
- Workflow made up of terminal commands and actions
- Many official and community-maintained actions are available

```
Build and test
succeeded on 17 Sep in 1m 20s

> ✓ Set up job
> ✓ Run actions/checkout@v2
> ✓ Setup .NET Core SDK 3.1
v ✓ Dotnet build

1 ▶ Run dotnet build --configuration Release
2
3
4
5
6
7 Welcome to .NET Core 3.1!
8 -----
9 SDK Version: 3.1.402
10
11 Telemetry
12 -----
13 The .NET Core tools collect usage data in order to help us improve your experience.
14 community. You can opt-out of telemetry by setting the DOTNET_CLI_TELEMETRY_OPTOUT
15
16 Read more about .NET Core CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry
17 -----
18 Explore documentation: https://aka.ms/dotnet-docs
19 Report issues and find source on GitHub: https://github.com/dotnet/core
20 Find out what's new: https://aka.ms/dotnet-whats-new
```

# Exercise

GitHub Actions

## Exercise – GitHub Actions

In this exercise, we're going to add continuous integration to a small .NET Core application. To do this we'll be using GitHub Actions.

[GitHub Repo Link](#)

## Part 2

Jenkins



# Recap

## Jenkins

- Alternative to GitHub Actions
- Free, however need to provide machines to run builds on (called *agents or nodes*)
  - The Jenkins Controller (i.e. the central machine which manages Jenkins) is an example of an agent
  - Agents can also be separate machines/VMs
  - These agents have multiple *executors*, each of which can run separate Jenkins tasks (often in parallel)
  - Jenkins can create containers inside these “machines” using docker. These containers are also referred to as agents but are run on machine agents and do not have executors
- Works with most code management services, e.g. GitHub, GitLab, BitBucket
- Requires more setup than GitHub Actions



# Jenkins

# Jenkins

## Containerised pipeline

- Can specify docker image for each pipeline step or the whole pipeline
- Pros: fewer system requirements for machine running Jenkins, scalable, portable
- Cons: slightly more complicated to set up and debug, can slow down pipelines

```
pipeline {
  agent any

  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }
    stage('Build') {
      steps {
        sh "npm ci"
        sh "npm run build"
      }
    }
  }
}
```

```
pipeline {
  agent {
    docker { image 'node:14-alpine' }
  }

  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }
    stage('Build') {
      steps {
        sh "npm ci"
        sh "npm run build"
      }
    }
  }
}
```

# Exercise

Jenkins

## Exercise – Jenkins

In this exercise, we're going to use Jenkins to set up an alternative continuous integration pipeline for our .NET Core app.

[GitHub Repo Link](#)

**Thank You!**