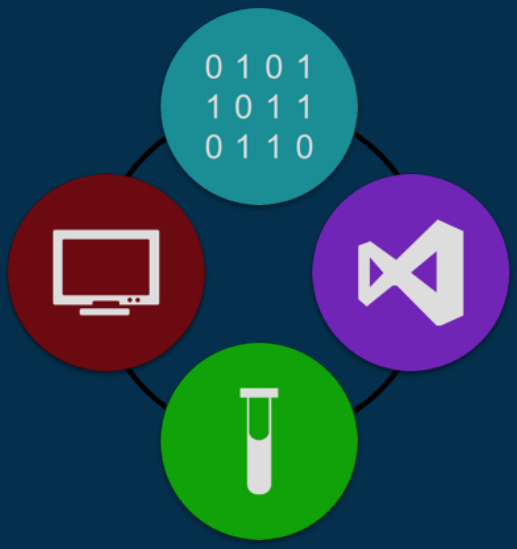
Xamarin Evolve 2014

Software that Builds Software

Tom Opgenorth
Documentor
tomo@xamarin.com

Continuous Integration Overview

- What is Continuous Integration (CI)?
- What Does it Do For Me?
- What Do I Need for CI?
- Setup
- Xamarin Test Cloud
- Guidance



Demo

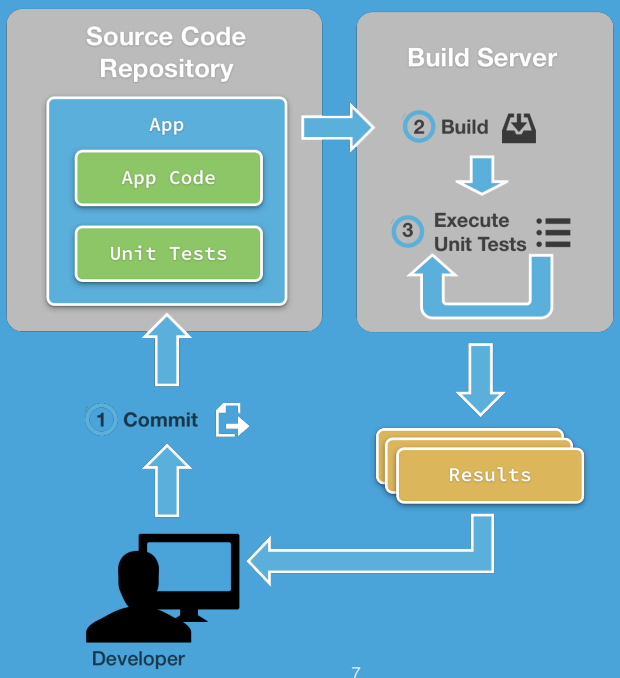
What Is Continuous Integration?

What Is Continuous Integration?

"Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early. "

Textbook Definition from ThoughtWorks

The Continuous Integration Workflow



Xamarin Evolve 2014

What will CI do for me?

Xamarin Evolve 2014

"Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove."

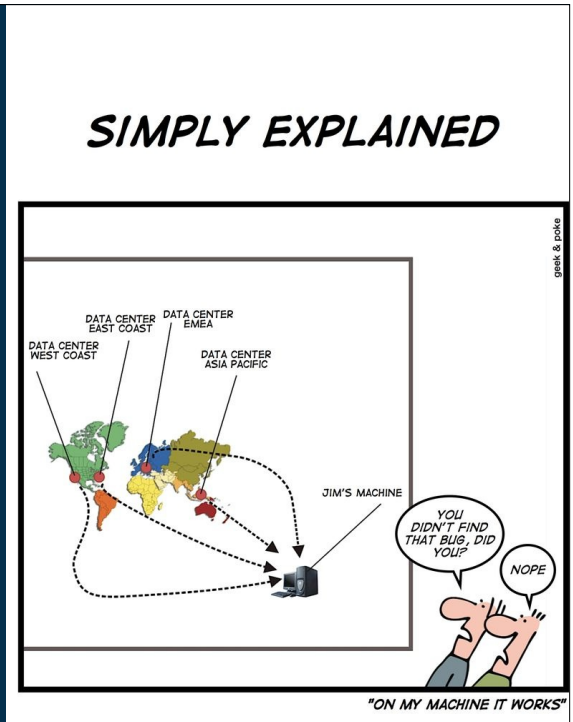
Martin Fowler
Chief Scientist, ThoughtWorks

Minimizes the "Merge Migraine"

- Frequent integration results in smaller commits
- Easier to rollback small commits

Find Problems Quicker

- Detect errors close to the source
- Cheaper and easier to fix

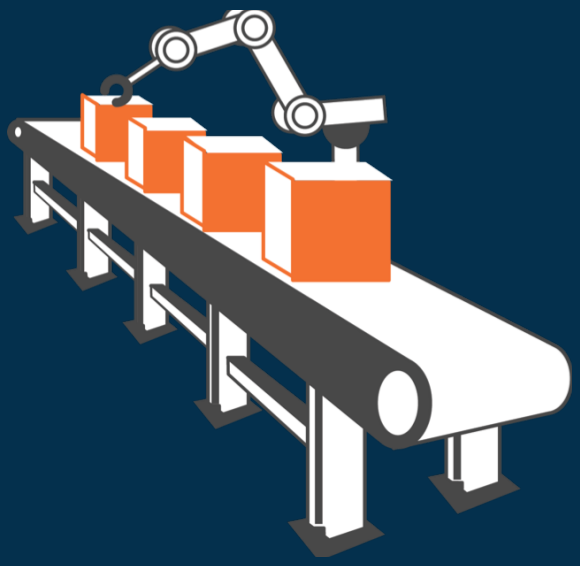


No more "Works on my Machine!"

- Consistent, repeatable builds
- App will work everywhere
- Less chance of *build contamination*

Deployable Software Any Time

- "We need to show this to a customer NOW!"
- Get bug fixes out quicker
- Release new features easier
- Confidence in the Software



What Do I Need for CI?

Source Code Management

Team Foundation Server
or
Visual Studio Online

Git / Github

Subversion

CI Server Software

Jenkins

- Open Source
- Windows & OS X
- Easy to setup
- Lots of plug-ins

CI Server Software

TeamCity

- Free Edition & Commercial Licenses
- Windows & OS X
- Easy to Install
- Scales Out
- Lots of Plugins

CI Server Software

TFS

- Not Just CI
- Windows Only - Still Requires A Mac Build Host
- Requires Xamarin Pro or Business

Setup

Xamarin Evolve 2014

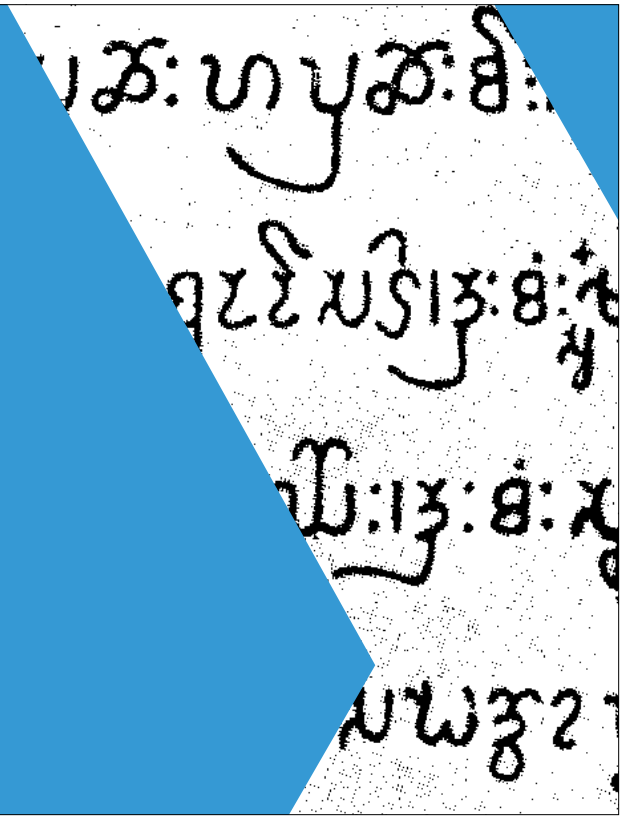
Setup Checklist

- Get Source Code Server
- Get CI Server Up & Running
  - Dedicated Computer
- Install & Activate Xamarin Studio
  - Authenticate with Component Store
- Write Build Script
- Configure Job or Build in CI Server

By Ragesoss via Wikimedia Commons

Why Script the Build?

- Documentation
- Traceability
- Developers can use it too!
- Easier to Debug and Maintain
- Simplifies Setting up the CI Server



Build Scripts

Option #1: Rake

- A DSL for Builds
- Lots of Gems to support builds
- Works on both Windows & Mac

Sample: A Rakefile

```
desc "Compiles the Android and iOS projects."
task :build => [:clean, :build_android, :build_ios] do

end

desc “Builds a Debug and Release IPA.”
task :build_ios do
  puts "Build the IPA:"
  system("#{@mdtool} \"--configuration:Debug|iPhone\" TaskyPro-Calabash.sln")
  system("#{@mdtool} \"--configuration:Release|iPhone\" TaskyPro-Calabash.sln")
end

desc "Build a Release APK."
task :build_android do
  system("#{@xbuild} /verbosity:diagnostic /t:SignAndroidPackage /
p:Configuration=Release ./Tasky.Droid/Tasky.Droid.csproj")
end
```


Build Scripts

Option #2: Powershell (psake)

- A DSL written in Powershell for Builds
- Great for Windows
- Not as many plugins (gems) as Rake

Sample: PSake

```
task default –depends Test

task Test –depends Compile, Clean {
  $testMessage
}

task Compile –depends Clean {
  $compileMessage
}

task Clean {
  $cleanMessage
}
```


Build Scripts

Option #3: FAKE - the F# make

- A DSL for Builds
- Works on Windows & Mac
- Very Rake-like

Sample: FAKE

```
#r "tools/FAKE/tools/FakeLib.dll" // include Fake lib

open Fake

Target "Test" (fun _ ->
    trace "Testing stuff..."
)

Target "Deploy" (fun _ ->
    trace "Heavy deploy action"
)

"Test"            // define the dependencies
   ==> "Deploy"

Run "Deploy"
```

Xamarin Test Cloud

Setting Up Test Cloud

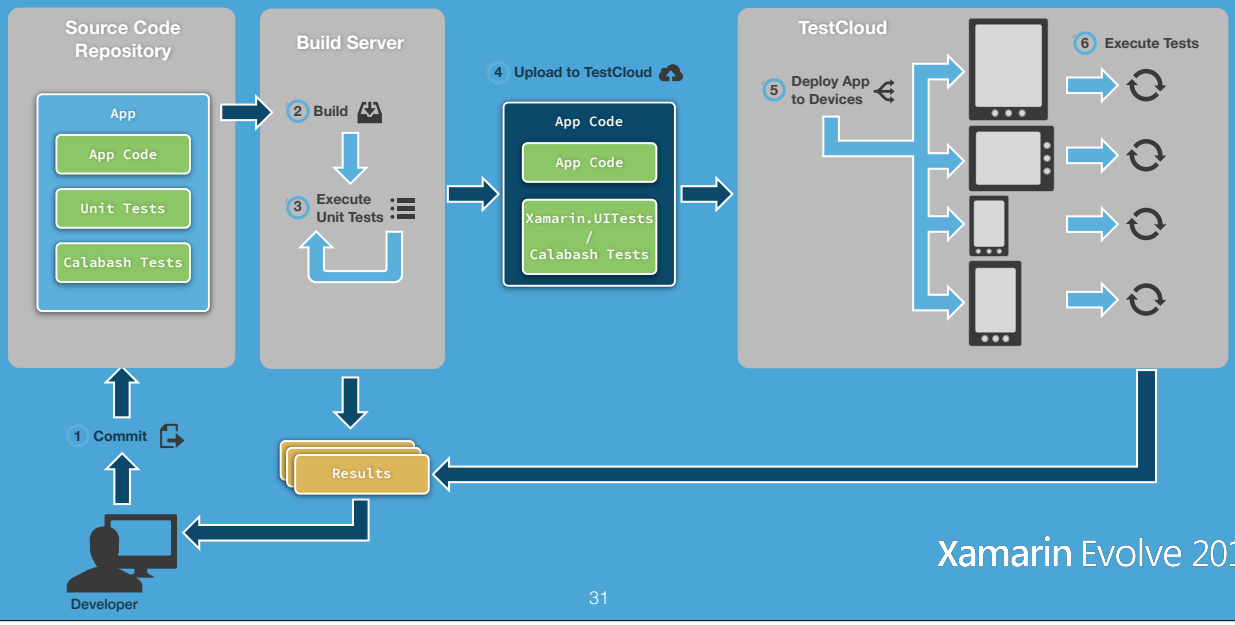
- Get a Test Cloud API key
- Create a Test Run to get the Device ID's for iOS & Android
- Install Calabash or Xamarin.UITest
- Install Calabash Component (for iOS)
- Update the Build Script

Sample Rake Task for Xamarin Test Cloud

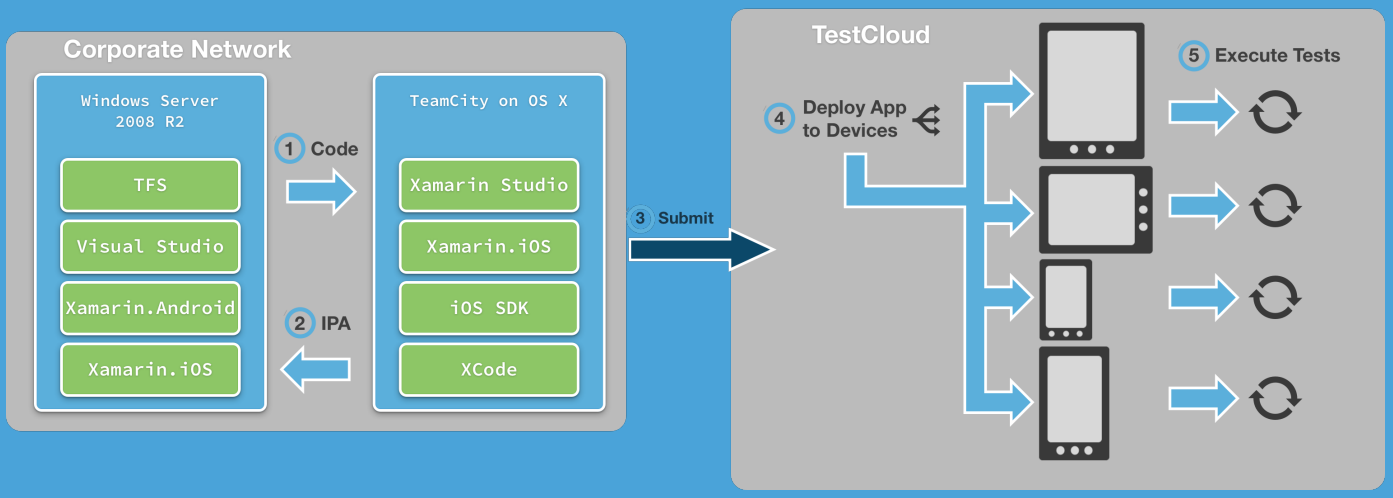
```
desc "Submits the IPA to Test Cloud."
task :xtc_submit_ios => [:require_environment] do
  raise "Missing the IPA #{@ipa}." unless File.exists?(@ipa)
  tests_passed = system("test-cloud submit #{@ipa} #{@testcloud_api_key}
                         -y #{@dsym} --devices=#{@ios_device_id}
                         --config=config/cucumber.yml --profile=ios
                         --pretty  --app-name=\"Tasky Pro\"")
  raise "Some tests failed in test cloud - check the build log. #{$?}" unless
tests_passed
end
```

New and Improved CI WorkFlow

Now With More Xamarin Test Cloud!



Team Foundation Server / TeamCity



Demo 2

Xamarin Evolve 2014

Advice From the Trenches

Xamarin Evolve 2014

CI Commandments

Thou Shalt:

- Automate
- Commit *Early*, Commit *Often*
- Build *Fast* - Fail *Fast*
- *Never* Commit *Broken* Code
- Fix Build Failures *Immediately*
- Build in *every* target environment

Xamarin Evolve 2014

Go Forth And Integrate Continuously

Tom Opgenorth
@topgenorth