BUILDING STABLE AGILE CLOUD APPLICATIONS

Dan Piessens

Twitter: @dpiessens
ABOUT ME

- Senior Agile Consultant
- 13 Years Experience as Developer, IT Consultant, Architect, Trainer, Coach
- 2008 — 2013 MS Patterns & Practices Champion
TALES FROM THE TRENCHES

“It was a good idea at the time . . .”
THE SUPPLY CHAIN COMPANY

- Large software vendor for supply chain solutions
- Software was designed for on premise installation
- Until one day . . .

They tried to go to the cloud!
THE INSURANCE COMPANY

- SaaS software for insurance brokers
- Large stable private cloud
- Until one day . . .

Open Enrollment Started
They fired the 3rd party vendor!
WHY DO WE CARE?

“It works fine most of the time”
IT’S ALL ABOUT PERCEPTION

- Response times affect perception
  - 0.1s - Users feeling that they are directly manipulating the UI
  - 1.0s - Users feeling that they are freely navigating the UI
  - 2.0s – User feel a noticeable delay in the UI
  - 10s - Users feeling that their experience is impaired

- This was from a paper in 1968!

http://www.nngroup.com/articles/response-times-3-important-limits/
CURRENT HARDWARE IS STALE
TO THE CLOUD!
CLOUD ADVANTAGES

- Agility
- Secure*
- Scalable
- Cost Effective
- Full Automation Available

* Easy to screw up
BUT YOUR APP LOOKS LIKE THIS!
Azure Active Directory

Active Directory

Active Directory

Azure Storage

LDAP query

Database

Azure SQL DB

Integration service

SQL Server

Web site

Browser

User profile

Windows event log

File share

Scans service

Azure VPN

Payment system

Input file

Output file
AVOIDING THE “BIG BANG”

- You don’t need to migrate all at once
- Start with what comes “out of the box”
  - Cloud Databases (SQL Azure)
  - Caching Providers (SQL Cache, Redis Cache)
- Setup a VPN
  - Extremely easy, script is done for you
  - Azure has ExpressRoute (L2 connection)
See Fowler’s “Inversion of Control Containers and the Dependency Injection pattern” at http://martinfowler.com/articles/injection.html for a great discussion on choices and tradeoffs here.
EXAMPLE: STORAGE

User Uploads Image

Website

IStorageService

File System

Azure Tables

Configuration Change
WHERE THIS HELPS

- **Hybrid Application Development**
  - One provider in local datacenter, the other in the cloud

- **Testability**

- **Separation of Concerns**
ADDING RESILIENCY

“I’m not quite dead yet...it’s just a flesh wound!”

-Monty Python
TRANSIENT ERRORS

- No dependency is available 100% of the time
- Need to separate application failures from transient failures
- Retrying on transient errors produces less error logs
  - BUT it increases wait time! More to come on that…
## TYPES OF RETRY POLICIES

<table>
<thead>
<tr>
<th>Retry strategy</th>
<th>Example (intervals between retries in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed interval</td>
<td>2, 2, 2, 2, 2, 2</td>
</tr>
<tr>
<td>Incremental intervals</td>
<td>2, 4, 6, 8, 10, 12</td>
</tr>
</tbody>
</table>
WHAT MAKES THIS WORK

- Asynchronous methods
- Isolated operations
- Known transient failures
- Recording when you “give up”
- Make retry strategies global (use IoC)
public async Task<ActionResult> Index()
{
    // Step 1 - Setup retry
    var retryStrategy = new ExponentialBackoff(10, TimeSpan.FromSeconds(2), TimeSpan.FromSeconds(20), TimeSpan.FromMilliseconds:

    // Step 2 - Create a retry policy
    var retryPolicy = new RetryPolicy<CustomTransientErrorDetectionStrategy>(retryStrategy);

    StockQuote quote;
    try
    {
        // Step 3 - Attempt the action
        quote = await retryPolicy.ExecuteAsync(() => this._stockService.GetQuote("MSFT");
    }
    catch (Exception ex)
    {
        // Log error here
        Debug.WriteLine("The call failed!, Details: {0}", ex);
        throw;
    }

    return View(quote);
}
Detecting errors can be difficult
- Dig into errors, check status codes, details in messages, etc.

Extensions exist to help
- Caching
- Database
- Storage
- Service Bus

What about authorization?

```csharp
public class CustomTransientErrorDetectionStrategy : ITransientErrorDetectionStrategy
{
    public bool IsTransient(Exception ex)
    {
        return (ex is SocketException);
    }
}
```
WHAT ABOUT RESPONSE TIME?

- Retry is good for availability but can make the user wait
- “What happens if we don’t want it to fail?”
- “I don’t want the site to tip over if things get busy”
- “Auto-scale doesn’t work for my application”
COMMAND & QUERY RESPONSIBILITY SEGREGATION

A CQRS Journey

1. Our domain
2. Decomposing the domain
3. Bounded context #1: Orders and Registrations
4. Extending and enhancing the Orders and Registrations bounded context
5. Preparing for the V1 release
6. Versioning our system
7. Adding resilience and optimizing performance

Testing
Task-based UIs
8. Lessons learned

Event sourcing
Domain-driven design

CQRS

Sagas and process managers
Bounded contexts
Domain events
HOW IT WORKS

- Updates write to an event store
- Workers process events
- Aggregate state is persisted
- System reads aggregate state
 DON’T use this for every system

Do research and read

Build the pattern incrementally
  - Pro Tip: Try a small application

Think about deployment
OTHER PERFORMANCE TIPS

- Caching
  - Static Content
  - Slow-Moving Results

- Compression and Optimization

- Call Separation
DEPLOYMENT AND THE CLOUD
<table>
<thead>
<tr>
<th>Continuous...</th>
<th>What it Does</th>
<th>Started By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>Builds and Asserts Code Quality</td>
<td>No-One (Automated)</td>
</tr>
<tr>
<td>Deployment</td>
<td>Manages Application Releases to an Environment</td>
<td>Anyone (Dev, Qa, Business)</td>
</tr>
<tr>
<td>Delivery</td>
<td>Releases New Functionality</td>
<td>Business</td>
</tr>
</tbody>
</table>
WHY DO DEPLOYMENTS FAIL?

- Large Work Batches
- Large Batches $\neq$ Deployment Size
- How often do you deploy your software?
MOST DEPLOYMENTS TODAY

- Develop
- Build Locally
- Wait for IT to Yell
- Hack settings
- Smoke Test
- XCopy
THE IDEAL DEPLOYMENT CYCLE

Develop

Gather Feedback

CI Build

Deploy

CI Test

Package

You should be doing these already!
WHAT MAKES THIS WORK?

- Small Work Batches
- Automated Quality Gates
  - Unit Tests, Code Coverage, Quality Checkers (FxCop, etc.)
- Repeatable Process
  - Goal: No Manual Steps During Deployment
DEPLOYMENT: OCTOPUS DEPLOY

- Flexible Deployment Tool
- Deploys both On-Premise and to the Cloud
- Focused on Windows Apps
  - Web Applications
  - Windows Services
  - Click Once Apps
  - Databases
  - PowerShell
**WHY CARE ABOUT DEPLOYMENT?**

- Deployment to the cloud can be complicated
- Think about deploying **everything** each time
  - Application
  - Database
  - Service Bus Topics / Queues
  - Storage Container
- Treat your application settings like your code
IT’S NOT JUST ABOUT SOFTWARE

- Automate Your Infrastructure
  - New Development Environments
  - Automated Testing
  - Disaster Recovery / Scalability

- Tooling
  - Puppet
  - Chief
  - PowerShell Desired State
FLEXIBLE RELEASES
FEATURE TOGGLS

- A mechanism to switch between features at runtime
- Separates delivery from deployment
- Typically done at the UI / Service layer
- Scary?

You already have this in your application… User Login and Authorization!
Brittle!

```csharp
HomePageModel homePageModel;
if (ConfigurationManager.AppSettings["ToggleMyFeature"] == "true")
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Well this is different...",
        SubMessage = "Something changed, not sure what",
        Title = "Base Page"
    };
}
else
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Welcome to my page!",
        SubMessage = "Now this is cool :)",
        Title = "Home Page"
    };
}
```
Easier to Refactor

```csharp
HomePageModel homePageModel;

if (ToggleManager.IsEnabled<ToggleMyFeature>())
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Well this is different...",
        SubMessage = "Something changed, not sure what",
        Title = "Base Page"
    };
}
else
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Welcome to my page!",
        SubMessage = "Now this is cool :)",
        Title = "Home Page"
    };
}
```

It's a Class!
Toggles and Deployment

- Toggles do stuff! = Has a performance impact
- Must correlate changes to runtime feedback
- Flip toggles via deployments
3 STAGE UPDATES

Orders

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10001</td>
</tr>
<tr>
<td>2</td>
<td>10002</td>
</tr>
<tr>
<td>3</td>
<td>10003</td>
</tr>
</tbody>
</table>

Web Service
3 STAGE UPDATES

STEP 1: DEPLOY NEW FUNCTIONALITY DISABLED

<table>
<thead>
<tr>
<th>Orders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Web Service
No Urgent Code
Uses Urgent Code
3 STAGE UPDATES

STEP 2: TOGGLE NEW SERVICE CODE

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
<th>Urgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10001</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>10002</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>10003</td>
<td>F</td>
</tr>
</tbody>
</table>
3 STAGE UPDATES

STEP 3: REMOVE OLD FUNCTIONALITY

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
<th>Urgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10001</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>10002</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>10003</td>
<td>F</td>
</tr>
</tbody>
</table>

Web Service
Uses Urgent Code
GETTING FEEDBACK
RUNTIME FEEDBACK

- Instrument your applications at runtime!
- Many tools available
  - New Relic
  - Application Insights
  - Splunk
  - Raygun.io
- Include User Analytics
  - Google
  - All above tools
DIG DEEPER FOR DATA

- Monitoring tools have APIs for tracing / logging
- Instrument key transactions in your system
- Track performance end-over-end for deployments
- Choose a logging framework and leverage it!
  - Log to a common location
Find errors during development

Make performance reviews part of your DoD

Example: Glimpse
- Trace your application
- Many plugins available
- http://www.getglimpse.com
TAKE AWAY CONCEPTS

- Scaling enhancements can always be done incrementally
- Deploy in small batches!
- Provide fast feedback
- Treat deployment settings like code
- Separate deployments from releases

"Deployments are like exercise, the more you do them the less it hurts"

-Dan Piessens
RESOURCES

- Moving Applications to the Cloud 3rd Edition

- Building Hybrid Applications in the Cloud on Microsoft Azure

- CQRS Journey

- Transient Fault Handling Core

- Octopus Deploy
  - http://www.octopusdeploy.com
QUESTIONS?

Thank You!