Building Scalable .NET Websites on Amazon Web Services

Part 1 - Understanding the AWS Cloud

* Please, spare us from “The Cloud”!
* Introduction
  + Why AWS?
  + Why .NET
  + Windows as a Second Class Citizen
  + What is different
  + What is the same
  + Competitors
    - Azure
    - Traditional Hosting
* Cloud Architectural Concepts
* The Development Process
* Using the Amazon Services
* Embracing the non-Windows Culture
* Creating an Architectural Mission Statement

Part 2 - Concepts

* Design for Failure
* Security
* Responsible Data
* Cost Models
* Open Source Projects
* Testing

Part 3 - Getting Ready to Code

* Breaking Away from Enterprise IT Practices
* Choosing and Creating a Windows AMI
* The public cloud and ALM solutions
  + Source Control Options
  + ~~Coming to terms with TFS loss~~
* Using elasticity for Continuous Integration and Build( Life without TFS)
* Non windows machines

Part 4 - Development

* Definitions of Scalability
  + Enterprise scalability
  + Cost Effective Scalability
  + Startup
* Dude, where's my data?
  + Architectural Considerations
  + Technologies
    - S3
    - RDS
    - SimpleDB

NoSQLPart 6-Security

* Security in the public cloud
* Working with Corporate security controls
* ~~Considerations of threats for traditional data~~ centres
* ~~Security practices for traditional data centres~~
* Considerations for the public cloud (in general)
  + Security on PaaS
  + Security on IaaS
* Comparing AWS security and traditional hosting
  + Physical security
  + Ownership of assets (e.g. Routers)
  + Removal of the DMZ
    - Assumptions about public access
* Risks of self managed/owned security assets compared to AWS
  + Patching
  + Intrusion detection
  + DOS
* Amazon commitment to security
  + Security certifications
  + Practice and guidance
* What you get 'for free' on AWS
* Elements of Security
  + Data security
  + Authentication
    - Federated security
  + Authorization
  + Encryption
  + Non-repudiation
* Principles for applying security across architecture
  + Database
  + Shared data store (e.g. S3, RDS)
  + Owned data store (e.g. mongoDB)
  + Application
    - Web app
    - Services
  + Administration
    - AWS console
    - Restrictions on remote desktops and root access
    - Monitoring data
  + Security groups
  + Using and deploying certificates
* Steps for setting up basic security on AWS
  + Developing simple threat model
  + Creating an account and protecting account information
  + Creating logins for the development team
  + Separating dev, test and production environments
  + RDS security
  + Locking down an instance
    - Linux
    - Windows
  + Security groups
  + S3 security
    - Buckets
    - Authorisation keys with expiry
  + Integration with existing authentication mechanisms (how would we do the equivalent to Azure federated identity)
  + Creating, installing and using certificates
    - SSL
    - X.509
  + Monitoring
    - For intrusion detection
    - For DOS attacks
* Websites
  + State and Caching
* Services
* Batches

Part 5 - Operations

* Scalability

Using AWS cloudwatch, ELB and autoscaling to effect scalability

Designing applications for scalability

* Elasticity
* Deployment
* Relationship to elasticity
* Configuration Managementt
* Monitoring
  + Need to say something about SCOM here not sure what though apart from the fact its big & bad but crucially you may stll be able to use it if the instances are part of your domain ( nasty though)
  + Cloudwatch
  + Integration with std 3rd party tools( e.g nagios, prgt)
  + 3rd party solutions e.g NewRelic
* Automation