Practical CQRS with Windows Azure

**Lokad Flavor**

Simplified practical approach to build and maintain complex cloud solutions with small teams.

**Time-Proven Design**
CQRS architecture principles are based on DDD\(^2\), Event Sourcing and cloud experience.

**By Cloud Enthusiasts**
Lokad team continuously learns and shares Azure experience via conferences and blogs.

With OSS Framework
Lokad relies on Lokad.CQRS for complex integrations, high-load data processing and business workflows.

1. CQRS = Command Query Responsibility Segregation
2. DDD = Domain Driven Design

Limitations
Every approach has these

We are still learning
Based on the current practical experience

Only 3 cloud projects
Not all scenarios are covered

Limited teaching experience
Only with distributed teams in EU/Russia

Startup Survival Mindset
Limited resources, low-friction development

Unlearning Curve
A bit of unlearning is needed to build for cloud
CQRS
Command-Query Responsibility Segregation

Design Principle
Separation of reads from writes

Architecture Model for Cloud
Patterns + Design + Project Guidelines

- Messaging
- Domain-Driven Design
- CQRS
- Event Sourcing
Simple Organizational Model deriving from Command-Query Responsibility Segregation

Native cloud distribution for fault-tolerance (cloud is cheap but volatile) and performance.

Windows Azure
Platform as a Service

- Pay-as-you-go
- Elastic Scaling
- Native .NET

Azure Fabric
Platform as a Service

- Web Role
- Custom VM

SQL Azure
Cloud MS SQL

- Table
- Queue
- Blob

Azure Storage
NoSQL

- **Client**: Web Role, ASP.NET MVC 2/3, WinForms, WCF REST API, AJAX
- **Service Bus**: Azure Worker, Service
- **Persistence**: Azure SQL, NoSQL, Azure Blob/Table
- **Domain Log**: Azure SQL, Table Storage
- **Views**: Azure Blob/Table Storage, NoSQL
- **View Handlers**: Azure Worker Role, Service
- **Publish Events**: Send Commands
- **Query Views**: Publish Events
Lokad.CQRS
for Windows Azure

CodePlex Project
- CQRS¹ App Engine for Azure
- Native Azure Queues
- Atomic Storage
- Streaming Storage
- Project Testability
- System.Transactions
- Decoupled Interface
- Lokad.Cloud Support

Cloud Architecture Guidance
- Tutorials
- Samples
- Article Series
  - Architecture Patterns & Practices
  - Development and deployment
  - Debugging, upgrades and maintenance
  - Full auditability
  - Scalability Theory
  - ASP.NET MVC Web Client Guidance
- Lokad Feature Studies
- Ask.Lokad Community

1. CQRS = Command Query Responsibility Segregation

Case 1: Salescast

Cloud integration engine capable of processing millions of product references, tailored for large retail networks.

**Lokad.CQRS** significantly simplifies the development while preserving cloud scalability. It allows to have less than one hour interval between committing code and reliably deploying latest changes into production.

**Features:**
- Multi-tenant
- Tenant-specific ad-hoc integration logic.
- Full audit logs.
- Auto detection of 3rd party business apps.
- API.

---

Case 2: Callcalc

Email-based forecasting client for call centers. You send an Excel spreadsheet with call volumes and Callcalc replies with forecasts.

Lokad.CQRS provides simple and reliable foundation for a heavily verticalized solution that tailors our raw Forecasting API for the very specific needs of call centers.

Features:
- Multi-tenant
- Multiple calling queues
- Erlang-C staffing optimization
- IMAP interface

Case 3: Lokad Hub

Platform unifying metered pay-as-you-go forecasting offered by Lokad.

Lokad.CQRS was key to transition a pre-cloud business app toward a decoupled and efficient design. It provided additional reliability and allowed much faster iterations.

Features:
- Multi-tenant.
- Full audit logs.
- Flexible reporting, including ad-hoc reports and temporal queries.
- Reliable failure handling.
- No-downtime upgrades.
- Integration with payment, Geo-Location and CRM systems.

Conclusion: Diverse Projects leveraging CQRS + Azure

**Lokad Salescast**
- Targeting retailers:
  - Massive Scalability – CQRS
  - Complexity – DDD
  - Handling integrations with unreliable legacy

**Lokad Callcalc**
- Targeting call centers:
  - Parsing Excel spreadsheets on the cloud
  - Blending automation and human support
  - Azure seamlessly integrated into existing IT

**Lokad Hub**
- Centralized SaaS subscription management:
  - Metered pay-as-you-go billing
  - Kept simple and lean with DDD
  - Full auditability to reduce dev & support friction

Same Resources and Experience
- Same Azure Stack
- Same Architecture
- Same Tools
- Same Teams
- Same Design Principles
- Same Scalability

References on CQRS with Windows Azure

- CQRS Info: http://cqrsinfo.com/
- CQRS Starting Point: http://abdollin.com/cqrs/

Presentation

- Rinat Abdullin: http://abdollin.com
- Lokad SAS: http://lokad.com

Questions?

- DDD/CQRS: http://groups.google.com/group/ddd cqrs
- CQRS + Azure: http://groups.google.com/group/lokad
- Private: rinat.abdollin@gmail.com