OpenCredo

Things I wish I'd known before I started with Microservices

GOTO Amsterdam - June 18, 2015

OpenCredo

Who are we?

Steve Judd - Lead Consultant
Tareq Abedrabbo - CTO

OpenCredo - an Open Source software consultancy

Agenda

OpenCredo

Things I wish I'd known before I started with Microservices

What?

Why?

How?

What's the story morning glory?

"Gotta have a definition, right?"

WHAT

OpenCredo

- It is an **architecture**
  - Independently deployable software components (services)
  - Stateless, loosely coupled, resilient
  - Communicate via explicit, published APIs
  - Each service fulfils a single business capability
  - Automated testing and deployment is essential
- It's a **choice**
  - It's not the only one
  - Commitment is key

So, what's the big deal?

Go on, convince me….

WHY

OpenCredo

- Encourages:
  - ★ loose-coupling
  - ★ separation of concerns
  - ★ single responsibility principle
  - ★ domain-driven design
- Good fit with Agile development practices
- Well-suited to a containerised infrastructure

Monoliths: friend or foe?

And are Microservices really the new black?

WHY

OpenCredo

Monoliths

- Familiar & well-understood
- Easy to develop, build & deploy
- Consequences of changing domain design are localised
- Limited scaling choices
- Long-term commitment to tech stack (technology lock-in)

Microservices

- Flexible scaling options
- Enables independence in development and deployment
- Reduces technology lock-in
- Better fault tolerance
- Build/deploy/execution infrastructure is complex (automation a must)
- Getting the domain (service) boundaries right can be difficult

WHY

OpenCredo

Monoliths

- Familiar & well-understood
- Easy to develop, build & deploy
- Consequences of changing domain design are localised
- Limited scaling choices
- Long-term commitment to tech stack (technology lock-in)

Microservices

- Flexible scaling options
- Enables independence in development and deployment
- Reduces technology lock-in
- Better fault tolerance
- Build/deploy/execution infrastructure is complex (automation a must)
- Getting the domain (service) boundaries right can be difficult

WHY

OpenCredo

Monoliths

- Familiar & well-understood
- Easy to develop, build & deploy
- Consequences of changing domain design are localised
- Limited scaling choices
- Long-term commitment to tech stack (technology lock-in)

Microservices

- Flexible scaling options
- Enables independence in development and deployment
- Reduces technology lock-in
- Better fault tolerance
- Build/deploy/execution infrastructure is complex (automation a must)
- Getting the domain (service) boundaries right can be difficult

The importance of contracts

"Until the contract is agreed, nothing is real"

HOW

OpenCredo

- Design your API contracts first
- Communicate them well
- Use tools to document them, e.g
  - apidocjs (http://apidocjs.com/)
  - swagger (http://swagger.io)
  - spring-restdocs (https://github.com/spring-projects/spring-restdocs)
- Be mindful of the impact of changing an API

HOW

OpenCredo

- If you don’t specify your contract, you end up with an implicit one anyway
- Use the power of resources (HTTP and REST)
  - ✤ Links and locations
  - ✤ Uniform interface and status codes
  - ✤ Representations

Does size matter?

Provide as many APIs and Services as you need but no more

HOW

OpenCredo

- Size - what really matters is quality not quantity
  - Services should be **decoupled conceptually** so that they can evolve independently
  - Services should be **decoupled technically** so that they can be managed independently
- What do I do, *practically*?
  - Co-locate services, but avoid implicit dependancies though shared common objects
  - Separate services but avoid sharing (domain) libraries

HOW

OpenCredo

- **Don’t** stress about how many APIs or Services
- **Do** stress about designing an appropriate domain models for your services
- **Don’t** separate your services based on technical boundaries
- **Do** separate your services based on self-contained functions

Separating the men from the boys

What does a good microservice look like?

HOW

OpenCredo

- Logging & monitoring
  - Centralised collection
  - Many more moving parts
- How the services are managed
- External configuration
- Handling failure
- Inter-process communication
  - Message serialisation/deserialisation
  - Network overhead

And finally….

Final takeaways

OpenCredo

1. Specify your **contracts first** AND communicate them
2. Design for **scale**: infrastructure, processes, services
3. You'll need to pay the **Distributed Service Tax**
4. **Everything is a Service** (aka eat your own dogfood)
5. Invest in **tooling** and **automation**

Thank you, any questions?

http://www.opencredo.com/blog
@OpenCredo
@cyberbliss
@tareq_abedrabbo