PORTKEY

Kimmo Koskinen, Christophe Grand; ClojuTRE 2017

Warning #1

The sensation of travelling by Portkey is universally agreed to be **uncomfortable**, if not **downright unpleasant**, and can lead to **nausea**, **giddiness** and **worse**.

–J.K. Rowling

Warning #2

This talk is unappropriate for Ents, Huorns, Groot, Idefix or tree-huggers as violent tree-shaking is depicted.

A crazy idea

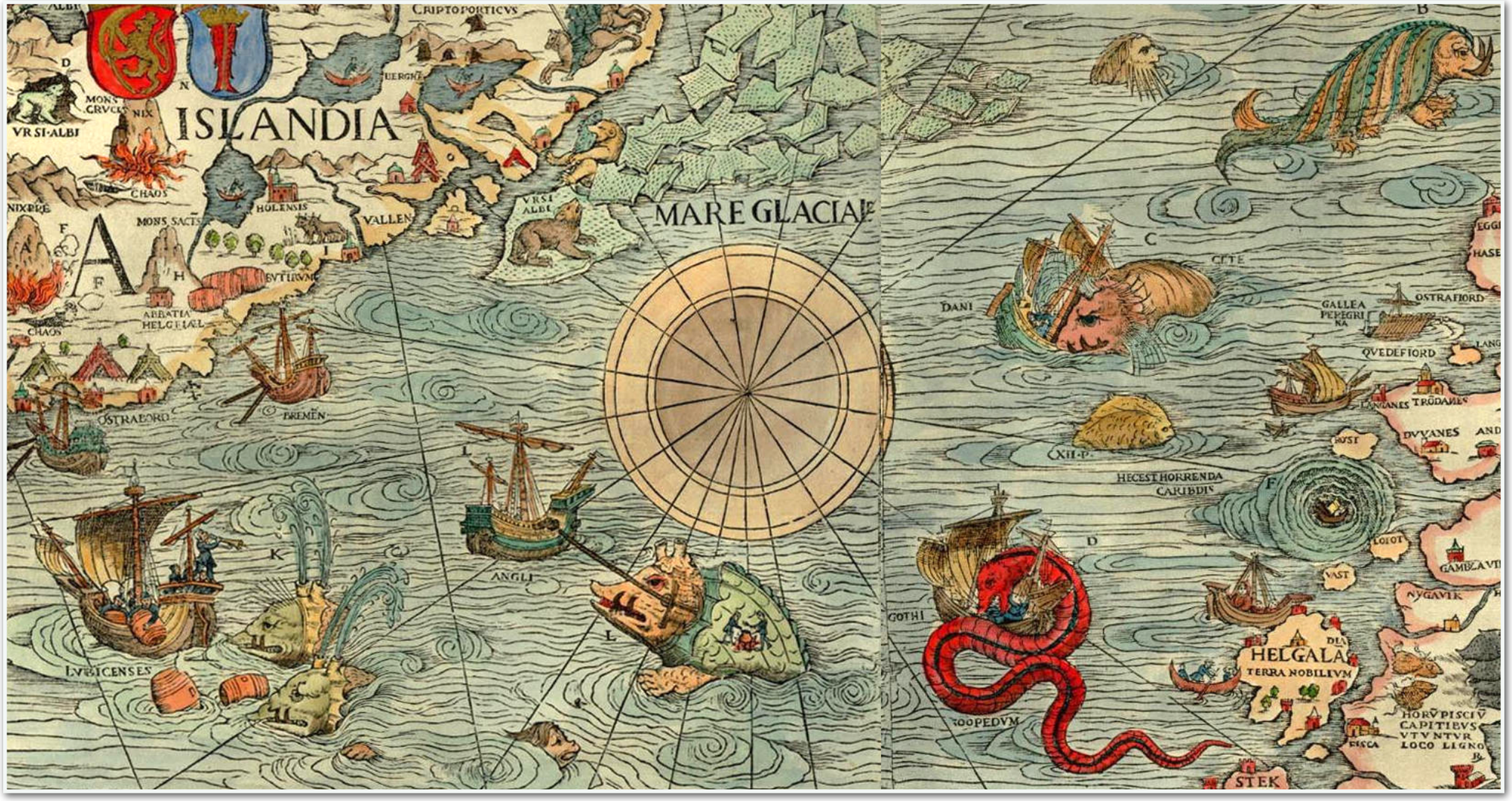
Live-code the cloud!

What can go wrong?

A minute plan

- Scavenge Ouroboros from Powderkeg
- Package classes for Lambda
- …
- Profit!

Uncharted waters



We ain’t afraid of no dragons!

TRE: 🐍🐯🌳 , no 🐲 , 😅


A
KYRO
PALIO
KYLA
LAYA
B
C
MOSTESUR
PERKALA
PIET MADO
MAÑA
SALA
VERO
KORSHOLM
K
HOLELA
LACVS
SVRPESI
MAXIMA HIC MVLTITVDO GENTIV
VESIKULA
HIC FABRICATVR
NAVES
TAVASTIA VEL
TAVESTHIA
REVZE
NERPIS

Aligned expectations

Hello World tree-shaked!

Nope, doesn’t work!

What’s in your Lambda?

Hadoop, what else?

Demo

Step #1 Scavenge Ouroboros

- Part of Powderkeg github.com/HCADatalab/powderkeg
  - Live-coding transducers on Spark cluster
- Ouroboros: connects the JVM to itself as a Java Agent
  - Super-powers: read and redef any class (and more)
  - No need for AOT to get classes -> live packaging

Step #2 Packaging

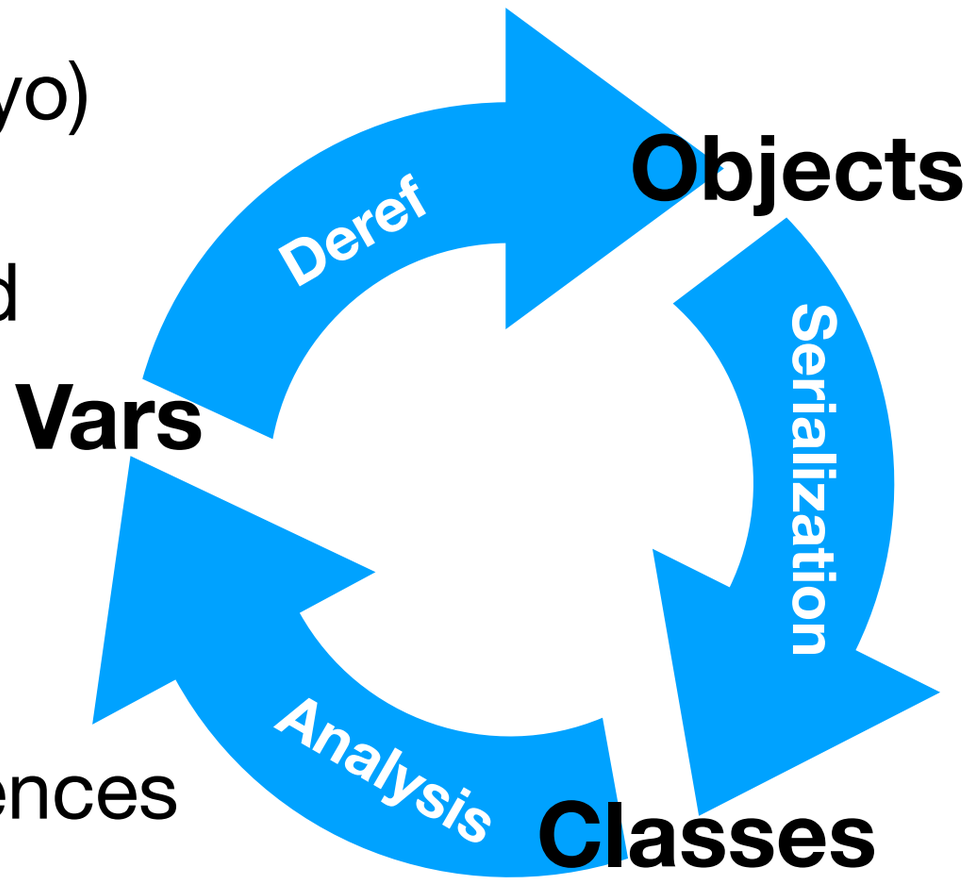
- Powderkeg: Pack’em all, Spark will know its own
  - Not applicable because of Lambda 50MB limit
- Tree-shaking!
  - ProGuard
    - Doesn’t know a thing about Clojure
    - GPL → cumbersome integration
    - NIH

Our Ghetto Tree-Shaker

- The root is a live object, not a static entry point.
    - Gives us an edge over ProGuard
    - Makes for our naivety
- Based on org.objectweb.asm.tree.analysis.Analyzer

Our Ghetto Tree-Shaker

- Serialization of live objects (Kryo)
    - Side-product: list of required classes
- Analysis of required classes
    - More classes, and var references (→ more live objects)



Abstract interpretation

- Dynamic code (classloading, var lookup) blinds analysis
- Approximate interpretation to narrow dynamic code behavior in some cases, e.g.:
  - `Class.forName("foo.bar.Baz")`
  - `(Foo) Class.forName(x)`
  - `RT.var("foo.bar", “f")`
  - …

Limitations

- Analyzis may be blinded by `some.lib.Utils/loadClass`
    - Manual hinting (“keeps”, should add profiles for common libs)
- Big Utils classes bring a lot of unrelated deps
    - Impacts package size

And yet it works…

	Before	After tree-shaking
Original	90MB	14MB
Manually Pruned	49.7MB	12MB

Lambda joy

- No build, What You Run Is What You Deploy
- just `(deploy! f "lambda-name")`, that's all!
- API Gateway helper: `(mount! f "uri")`
  - Data ❤️ Clojure, Swagger under the hood
    - Spec support planned
  - URI templates only for now

Are we done yet?

- No!
- Fluent REPL experience = fast feedback
  - Automatic deployment on redef
  - The latency dragon has to be slain!
- Better tree-shaking
- Wider integration

Latency

- Dominated by upload (unless repl running on EC2…)
    - Initial deploy: no satisfying idea yet (public classes repo, background optimistic upload)
    - Redeploys: send deltas and create new artifact online
- JVM/Clojure cold start
    - Better tree-shaking, shaking Clojure itself 🐲 🐲 🐲
    - Skip traditional deployment with custom classloader🤔

Better tree-shaking

- Cross-class
  - to understand class loading utils
  - less « keeps » (manual inclusions)
- Method granularity
  - to cut dependencies knots (Utils classes)
  - to remove unused methods
- Being brave enough to unleash it on Clojure proper

Wider integration

- Spec
  - Validate/conform inputs
  - Infer swagger
- Tests
  - Redeploy when tests pass, Extreme CD
- More and better AWS helpers (API Gateway, Cron, S3, Dynamo, RDS…)

AWS bindings

- Manual JSON and AWS signature is tedious
- aws-java-sdk is interop
    - generated from machine-readable specifications
- amazonica wraps aws-java-sdk
    - generates vars by reflection on SDK classes
    - converts data (clj) to objects to data (json|xml)
- portkey/aws-clj-sdk a new hope
    - generated, spec-based

Example

```
=> (clojure.spec.alpha/conform
     :portkey.aws.lambda/create-function-request
     {:function-name "name"
      :handler "portkey.LambdaStub"
      :code {:zip-file (.getBytes "random bytes")}
      :role "arn:aws:iam::012345678901:role/pk"
      :runtime "java8"
      :memory-size 1536
      :timeout 30
      :environment {:variables {}}})
{"FunctionName" "name",
 "Handler" "portkey.LambdaStub",
 "Code" {"ZipFile" "cmFuZG9tIGJ5dGVz"},
 "Role" "arn:aws:iam::012345678901:role/pk",
 "Runtime" "java8",
 "MemorySize" 1536,
 "Timeout" 30,
 "Environment" {"Variables" {}}}
```

Spin them off!

- Tree shaker
    - Little coupling with Lambda
- aws-clj-sdk
    - Already spun at github.com/portkey-cloud/aws-clj-sdk

Thanks

- Code lives at github.com/portkey-cloud/portkey
- If you ain’t afraid of dragons, try it!
    - After-sales: #portkey on Clojurians’ Slack
- Stalk us on Twitter @KimmoKoskinen @cgrand