Speaker Notes: Ammonite: Succinct Scala Shell Scripting

Markus Dale, medale@asymmetrik.com

October 2019

Setup

- Open Ammonite site https://ammonite.io
- Download page: https://github.com/lihaoyi/Ammonite/releases
- Requests lib: https://github.com/lihaoyi/requests-scala
- CSV data: https://catalog.data.gov/dataset?res_format=CSV&page=2, https://data.ny.gov/api/views/e8ky-4vqe/rows.csv?accessType=DOWNLOAD

Intro, Slides And Code

- Bio:
  - mostly Java, big data with Hadoop
  - big data with Spark, Databricks, Scala
  - Now Asymmetrik - Scala, Spark, Elasticsearch, Akka...
  - Data Engineer
  - Slides: https://github.com/medale/prez-ammonite-scala-shell/blob/master/presentation/SparkDataEngineering.pdf
  - Code Examples: https://github.com/medale/prez-ammonite-scala-shell/code

Examples of Shell Scripting

- Automate a small but labor-intensive task
- Run clean-up jobs with cron
- Refresh all git repositories (git pull)
- build and publish all projects under a common dir

That Bash Syntax

- hard to remember if not used all the time
- each command has its own set of options
- Can pipe things together - very useful
- No types - mostly treated as strings (other than exit code)
- Dr. Google - but fragile to maintain

Scala: file exists?

- Use java import
- More verbose

Scala: find and delete

- from 1 liner to 8 (and don’t forget close)

Ammonite: file exists?

- Domain-specific language for Scala shell scripting
- Make common tasks as terse as possible (but typed)
- imports Scala commands and implicits to convert strings to paths
- additional ease of use to remove need for full project/build system

Installing Ammonite

- Open https://github.com/lihaoyi/Ammonite/releases

Other ways of installing Ammonite

Configure the shell ~/.ammonite/predef.sc

Major improvements over Scala REPL

- open Scala REPL and Ammonite
- Syntax highlighting input/output
- output valid Scala code: `Seq.fill(10)("Hello, Ammonite")`
- multi-line editing (up arrow and search)
- Classpath search: `StandardCharsets<TAB>` (if not imported yet but on classpath)

Importing functionality from scripts

```
import $file.CommonImports
CommonImports.printCwd //accesses Paths OK
Paths.get("foo") //no imports - error
CommonImports.printWithNewlines(List(1,2,3))

import $file.scripts.Utils
Utils.du()
import Utils._
du()

import $file.^.print
print.message()
```

exec - bring in the defs and imports

- all definitions AND import statements

```
//contains: import java.nio.file._

import $exec.CommonImports
Paths.get("foo")
import CommonImports._
printWithNewlines(List(1,2,3))
```