Immutable Infrastructure in the Mesos Ecosystem

Kevin Sweeney (@kts)
Software Engineer
Twitter
What is Immutable Infrastructure?

- Snapshot-equivalent
- Destroy, Recreate and Scale “at-will”
- Single-specification
- Automated
Ideal State

- Failure Recovery (Rare)
- Software Deploys (Common)
- Capacity Add/Remove
Packaging

- Machine Images
- Flattened Container Images
- Layered Container Images
- App Bundles (WAR, PEX, etc)
- Layered App Bundles (POM, requirements.txt, etc)
Machine Images

- The full machine and all the software on it
  - Linux Kernel
  - Mesos Agent
  - Framework Executors?
Machine Images

- Pretty immutable
- General Purpose
- Slow deploys
  - Fully out-of-band of Mesos!
Flattened Container Images

- Contains Executor, Application, all Libraries
- Depends on Kernel and Agent APIs
Flattened Container Images

• Mostly immutable except:
  • Linux Kernel
  • Mesos Agent

• Slow startup
  • High Fetch Time
  • Not very cacheable
Layered Container Images

• Contains Executor, Application, all Libraries
  • Depends on Kernel and Agent APIs
  • Depends on Containerizer Runtime
Layered Container Images

- Mostly immutable except:
  - Linux Kernel
  - Mesos Agent
  - Containerizer Runtime
- Slow startup
  - High Fetch Time
  - Sometimes very cacheable
App Bundles

- Contains Application, all Libraries
- Depends on Kernel and Agent APIs
- Depends on Executor, Language runtime
App Bundles

• Mostly immutable except:
  • Linux Kernel
  • Mesos Agent
  • Containerizer Runtime
  • Language Runtime

• Faster startup
  • Medium Fetch Time
  • Not cacheable (but runtime installed out-of-band)
Layered App Bundles

• Contains Application
  • Depends on Kernel and Agent APIs
  • Depends on Executor, Language runtime
  • Depends on All Libraries
Layered App Bundles

• Mostly immutable except:
  • Linux Kernel
  • Mesos Agent
  • Containerizer Runtime
  • Language Runtime
  • Fetcher Runtime

• Faster startup
  • Short Fetch Time
  • Very Cacheable (but runtime installed out-of-band)
Pitfalls

- “Configuration Service”
- Mutable labels
- Puppet/Chef/Automatic OS Updates
- Non-concrete dependencies
Improvements

• Command-Line Flags
• Content-Addressed
• Master Organization Image
• Pinned dependencies
Pitfalls

• Combinatorial explosion
  • Linux Kernel A -> B
  • Mesos Agent C -> D
  • Containerizer E -> F
Example - Aurora

- “update” - Move from current config to new config slowly and with health checks

- Creates Mesos tasks from config snapshot.
Example - Aurora

- “Binding helpers” - leave unresolved references in a master config file.
- {{package.version[latest]}}
- {{docker.image[ubuntu][14.04]}}
Example - Aurora

- Optional “compilation” step

- aurora config read < job.aurora > job.json
Conclusions
Tradeoffs

- Startup time!

- VM >> Container >> App Archive >> Native Binary
Tradeoffs

• Generalizability

• Better performance + reliability if you can cache common libraries
Questions
Get Involved

- https://aurora.apache.org
- #aurora on irc.freenode.net
- dev@aurora.apache.org
- user@aurora.apache.org