Functional Infrastructures

Toni Batchelli, @disclojure
PalletOps
Clojure/West 2013
Infrastructure Automation

- Write programs that will build and operate computing environments
- Increase repeatability and reliability, reduce time and resources
- **Manage complexity**
Complexity

- Dev, QA, Perf Tests, Production
- Cloud, Containers, VMs, Hardware
- Clusters, hot stand-by, replica sets
- OS, services
Pallet

A Functional Infrastructure Automation Platform written in Clojure, 3+ years of development, 30+K lines of code

Design Constraints:
- Works in today’s environments
- Scales well with complexity
- Works everywhere:
  - Cloud, VM, Hardware, Containers...
  - Ubuntu, Centos, RedHat...
- 1st class support for Clusters
- Extensible and Embeddable
Managing Complexity

- Abstractions
- Reusable Components
- Stateless operation
- Purely functional code
- Library (vs. a service)
(println "hello world!")

echo hello world!
(let [files ["a" "b" "c"]]
  (actions/exec-script
   (doseq [file ~files]
     ("ls" @file)))))

for file in a b c; do
  ls ${file}
done

---

**Script DSL**

**Script Lib**

**Action**

**Spec**

**Plan**

**your domain**

**Script DSL**
(println (~lib/user-home tbatchelli))

Ubuntu:
```bash
echo $(getent passwd tbatchelli | cut -d: -f6)
```

OSX:
```bash
echo $(dscl localhost -read /Local/Default/Users/tbatchelli \ 
  dsAttrTypeNative:home | cut -d ' ' -f 2)
```
(actions/user "test-user" :groups ["group-a" "group-b"])

**Ubuntu:**
if getent passwd test-user;
  then /usr/sbin/usermod --groups "group-a,group-b" test-user;
  else /usr/sbin/useradd --groups "group-a,group-b" test-user;
fi

**Centos:**
if getent passwd test-user;
  then /usr/sbin/usermod -G "group-a,group-b" test-user;
  else /usr/sbin/useradd -G "group-a,group-b" test-user;
fi
(require '[pallet.crate.java :as java])

(plan-fn (java/install))

where:

(defplan install
  "Install java. OpenJDK installs from system packages by default."
  [:keys [instance-id]]
  (let [settings
        (get-settings
          :java {:instance-id instance-id
                :default ::no-settings})]
    (debugf "install settings %s" settings)
    (crate-install/install :java instance-id)
    (set-environment (:components settings))))
ACTION: pallet.actions/package of type script executed on target
FORM:
  (pallet.actions/package ("openjdk-7-jdk"))

SCRIPT:
  |
  |  
  |  { debconf-set-selections <<EOF
  |  debconf debconf/frontend select noninteractive
  |  debconf debconf/frontend seen false
  |  EOF
  |  } && apt-get -q -y install openjdk-7-jdk+ && dpkg --get-selections
  |  } || { echo '#> [install: install]: Packages : FAIL'; exit 1; } >&2

ACTION: pallet.actions/exec-script* of type script executed on target
FORM:
  (pallet.actions/exec-script* "echo 'install: set-environment: system-environ...'")

SCRIPT:
  |
  |  echo 'install: set-environment: system-environ...';
  |
  |  
  |  if ! ( [ -e /etc/environment ] ); then
  |  { cat > /etc/environment <<EOFpallet
  |  # environment file created by pallet
  |  
  |  EOFpallet
  |  }
  |
  |  pallet_set_env() {
  |  k=$1; v=$2; s=$3
  |  if ! ( grep "${s}" /etc/environment 2>&- ); then
  |  sed -i -e "/${k}=/ \ d" /etc/environment && sed -i -e "$ a \"
  |  ${s} " /etc/environment || exit 1
  |  fi
  |  } && vv="$(dirname $(dirname $(update-alternatives --query javac | grep 
  |  Best: | cut -f 2 -d ' ' )))"
  |  pallet_set_env "JAVA_HOME" "${vv}" "JAVA_HOME="/"${vv}""
  |  } || { echo '#> install: set-environment: system-environment: plan-when: Add java
  |  environment to /etc/environment : FAIL'; exit 1; } >&2
(def web-server-node
  (node-spec{:image {:os-family :ubuntu
    :os-version "10.04"}}
  {:hardware {:cpu-count 12
    :min-ram (* 64 1024)}}))

(def web-servers
  (group-spec "web-server"
    :node-spec web-server-node
    :phases
    {:configure (plan-fn
      (java/install)
      (tomcat/install))}
  (converge
    {web-servers 5}
    :compute-service (compute-service
      :aws-ec2 ...))
(defn web-server-node [cpus ram os-family os-version]
  (node-spec
   {:image {:os-family os-family
            :os-version os-version}}
   {:hardware {:cpu-count cpus
              :min-ram (* ram 1024)}})

(defn web-servers [cpus ram os-family os-version]
  (group-spec
   :node-spec
    (web-server-node cpus ram os-family...)
   :phases
    {:configure (plan-fn
                  (java/install)
                  (tomcat/install))}
  (converge {(web-servers 12 32 :centos "6.3")
             :compute-service
             (compute-service :aws-ec2)}})
(def platforms [[:centos "6.3"]
               [:ubuntu "10.04"]
               [:rhel "7"]])

(defn webservers-to-build [ps]
  (zipmap (map (fn [[os-family os-version]]
                (web-servers 12 32
                              os-family os-version))
             ps)
           (repeat 1)))

(converge (webservers-to-build platforms)
           :compute-service ec2)

(converge (webservers-to-build platforms)
           :compute-service virtualbox)
(require 'pallet.crate.cassandra :as cassandra))

(group-spec cassandra
  :node-spec { :hardware { :cpu-count 12
                      :min-ram (* 64 1024) }
           :extends [ (cassandra/server-spec {}) ]})

(converge { cassandra 6 } :compute aws-ec2)
(converge { cassandra 3 } :compute virtualbox)
plan = lift( current, plan-fns )
\[\text{plan} = \text{lift} \left( \text{current, plan-fns} \right)\]

\[\text{desired system} = \text{exec}(\text{plan})\]
Data

- Data allows to perform heavy lifting operations very simply
- Data is easy to test, inspect, debug, log
- Defer execution as much as possible
- Pallet internals are built around data manipulation
  - Coupling between components is data
  - All intermediate representation is data, until right before the execution
Where are we now?

- Functional and programmatic infrastructure automation
- Works on most cloud providers and target OSs (as long as they’re *nix)
- Build complex and flexible clusters
- Fast development paths
- Easy to build your domain abstractions on infrastructure
- Sometimes we wish we had static typing...
Infrastructure Automation

~

Clojure Development

~

http://palletops.com