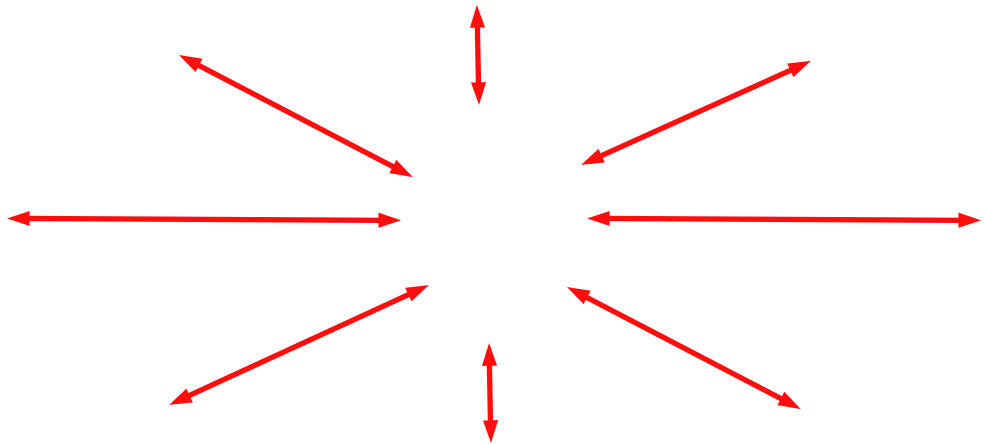
# Cloud Application Management for Platforms (CAMP) 1.0:  An Introduction

# Agenda

- Motivation, Introduction & Scope
- Why?
- What?
- Standardization: Issues & Missing Pieces
- Q&A



Elastic Beanstalk

# Existing Platform Application Mgmt APIs

- Most platforms (Elastic Beanstalk, GAE, Heroku, OpenShift, Oracle Java Service, …)
  - Command line interface
  - Web based console
  - Unpublished RESTful API that can change

# Why: Motivation (1)

- No existing PaaS Application Management standard
- Proprietary APIs are proliferating
- Each PaaS management API causes friction between clouds, limiting growth of the market
  - A standard alleviates customer concerns and increases adoption
- Customers will prefer early adopters of a Platform standard over the slower adopters
  - Implementation of the standard itself will be a “feature”

# Why: Motivation (2)

- Interoperability
- Portability
- Consensus management API
  - Enables providers to invest resources in other, more valuable areas
  - Serves as a basis for innovation and value-add

# Introduction (1)

§ Cloud Application Management for Platforms (CAMP)

- http://www.cloudspecs.org/paas/


# Introduction (2)

§ Simple API to manage applications in a PaaS cloud

- Resource & lifecycle model for application management
- Portable packages
- HTTP-based RESTful API
- JSON serialization
- Extensible
- Language- (Ruby, Java, Python, PHP, etc.), framework- (Rails, Spring, etc.), and platform- neutral (Java EE, .Net etc).

§

# Scope (1)

## A Humble Spec

- Management of applications and their use of the Platform
  - Upload
  - Manage lifecycle (configure/customize, deploy, undeploy, start, stop, snapshot, suspend, restart, delete)
  - Enable Monitoring
-

# Scope (2)

# Why: Goals (1)

- Simple
- HTTP/REST
- JSON
- Interoperability
- Portability
  - Application packaging
  - Tools
  - Skills

# Why: Goals (2)

§ Extensibility

- Platform vendors can still extend the standard for their own value added features (no lowest common denominator)

# Why: Use Cases (1)

- Moving applications between clouds
  - Public cloud A <> Public cloud B
  - On-premise/Private cloud <> Public cloud
- Managing App lifecycle consistently
  - Starting, stopping, snapshotting, suspending, resuming, patching, deleting an application
- Monitoring Support
- Changing configuration and runtime parameters

# Why: Use Cases (2)

# Why: Use Cases (3)

§ Develop App in an ADE running in the cloud
  - deploy to cloud

§

# What CAMP is not (1)

- Topology and orchestration
- Functional (non-management) interfaces to Platform and application services
  - E.g., interface to a message service bus or a database service offered by a Platform
- Facilities and interfaces that are language-, framework or platform-specific (e.g. .Net, Java EE)

# What CAMP is not (2)

- Management of underlying Platform resources
  - Management of underlying IaaS layer (if it exists)
    - Agnostic to underlying layer
  - Definition of Platform services (functional interfaces)
    - these may take many different shapes and offer a variety of APIs
    - E.g., Messaging service functional interfaces

# What: Lifecycle

- The lifecycle of an application proceeds through multiple stages:
  - Uploaded
  - Deployed
  - Instantiated
  - Suspended
- These transitions are controlled through the CAMP interface
-

# What: Packaging

§ CAMP includes a portable packaging format

- ADE ←→cloud
- Cloud A ←→ Cloud B

# What: Protocol

- Based on REST Principles
- HTTP
- JSON serialization
- Extensible to accommodate vendor extensions
- Authentication via TLS

# What: Protocol Example (Deploy Request)

- Request
  - POST /myPaaS HTTP/1.1
    Host: example.org
    Content-Type: ...
    Content-Length: ...

    {
      "pdp_uri": "/myPaaS/pkgs/1"
    }
-

# What: Protocol Example (Deploy Response)

- Response
  - HTTP/1.1 201 Created
    Location: http://example.org/myPaaS/templates/1
    Content-Type: ...
    Content-Length: ...

    ...
-

# What: Protocol Example (Start)

- Request
  - `POST /myPaaS/templates/1 HTTP/1.1`
    `Host: example.org`
- Response
  - `HTTP/1.1 201 Created`
    `Location: http://example.org/myPaaS/apps/1`
    `Content-Type: ...`
    `Content-Length: ...`

    `...`
-

# What: Protocol Example (Monitor Request)

- Request
  - `GET /myPaaS/apps/1 HTTP/1.1`
    `Host: example.org`

# What: Protocol Example (Monitor Response)

- Response
  - HTTP/1.1 200 OK
    Location: http://example.org/myPaaS/templates/1
    Content-Type: ...
    Content-Length: ...

```
{       "uri": "http://example.org/myPaaS/apps/1",
                    "name": "Hello Cloud App",
                    "applicationComponents": [
                    {"href": "/myPaaS/apps/1/acs/1”},
                    {"href": "myPaaS/apps/1/acs/2”}],
                    "platformComponents": [
                    {"href": "/myPaaS/pcs/1”},
                    {"href": "myPaaS/pcs/2”}],
                    "assemblyTemplate": "/myPaaS/templates/1",
                    "resourceState": {
                    "state": "RUNNING"
                }
            }
```

# What: Management Model

# What: Management Model

## Traversing Resources (Runtime Discovery Via URI Links)

# CAMP Standardization

## Missing Pieces

- Defining PDP
- Extensibility model and framework
  - Attributes
  - Resources
  - Protocol/API
  - Lifecycle
  - Extension and metadata discovery
- Testing
  -
-

# CAMP Standardization

## Issues

- Resource model
- Media type
- Life cycle
- Other minor/misc. issues

# Q&A