

Gluing Together Amazon Web Services with Perl

Timothy Appnel
tima@cpan.org
<http://appnel.com/>

Perl is the glue language
of the Web

Perl is the glue language
of ~~the Web~~
“Amazon Cloud”

EC2

- COMPUTE!
- Virtual computing environment
- Requisition machines on-demand
- Load custom software setups (images) and configure them however you like
- Run as many or as few as you want for what ever time you need them

EC2

- Images (AMIs)
- Instances (Machines running an image)
- Also manage registration, security, "size", addressing
- The ultimate sandbox tool
- Examples: `Net::Amazon::EC2` & `Net::Amazon::EC2::Metadata`

S3

- Storage on-demand from KB to TB
- Buckets (Folders)
- Keys (Files)
- Buckets use a flat namespace
- Example: Amazon::S3

SQS

- Highly scalable, hosted queue service for distributed messaging
- Unlimited number of queues
- Messages can be up to 8K each
- Messages can be queued for up to 4 days
- Example: `Amazon::SQS::Simple`

Oh by the way...

- Amazon::SimpleDB
 - “...create and store multiple data sets, query your data easily, and return the results.”
 - Requires no schema
 - Automatically indexes your data
 - Still in limited beta (and so is my code)

Introducing Amazon::Bezos

- A Perl toolkit for working with the Amazon Web Services cloud in an integrated and seamless manner.
- (And eventually) a more powerful and easier to use command-line tool

Objectives

- Less code to wire together multiple AWS services
- Foundation for more specific systems utilizing the AWS cloud
- Command line shell tool(s) for working with services -- pipelining etc -- that can be extended
- Easy to install

Milestones

M1

- Shared and multiple account information across service clients

M2

- Command tool(s) for working with cloud services using App::Cmd

Milestones

M3

- Amazon::SimpleDB support.

M4

- Consistent uniform interface across all service interfaces
- Uniform error handling
- Reduce number of dependencies

Issues & Future Considerations

- Different service interface styles
- Dependencies are more broad than they probably needs to be
- EC2 module isn't entirely object oriented and make subclassing hard
- EC2 module == dependency hell
- A lot of redundant utility code

Going Forward

- Just Scratching The Surface
- Try it out. Participate!
- Google Code Project Page:
<http://code.google.com/p/bezos/>
- Amazon::Bezos Project Page
<http://groups.google.com/group/bezos-dev>

