FIND THIS!
SEARCHING IN THE CLOUD WITH CLOUDSEARCH

AWS Charlotte Meetup /
Charlotte Cloud Computing Meetup
Bilal Soylu
June 2013

Agenda

- Hola!
- Housekeeping
- What is this use case
- What is Amazon CloudSearch
- What can we do with this
- How much is it going to cost me
- Let’s Brake it Down
  - Concepts, Examples, etc.

Hola! Guten Tag! Bonjour!

- Bilal Soylu
  - CTO Verian Technologies LLC (www.verian.com)
  - Of course, we are looking for peeps ! What kind of question is that!
  - Like to play with AWS stuff
  - Open Source Contributor
  - I really, really learn from my mistakes ;o)
- Blog
  - http://BonCode.blogspot.com
- Contact
  - @BmanClt
  - bilal.soylu@gmail.com

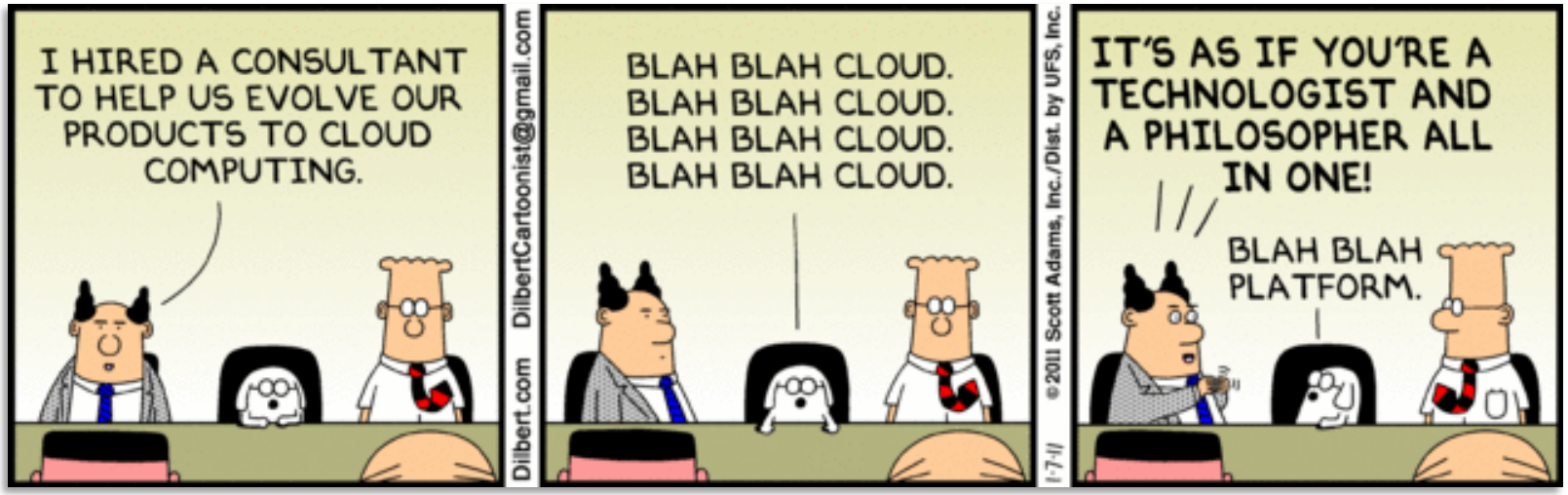
Housekeeping

- Meeting Place
- Meeting Time
- Meeting topics (we're scheduling ahead)
- The reoccurring meeting or specific meeting?
- Speakers please. Now is the time to think about change. We are glad to cycle in your topic.
- Communication
  - More / Less
  - What medium to use

Let’s get started

Find This!

Cloud Humor


I HIRED A CONSULTANT TO HELP US EVOLVE OUR PRODUCTS TO CLOUD COMPUTING.
Dilbert.com DilbertCartoonist@gmail.com
BLAH BLAH CLOUD. BLAH BLAH CLOUD. BLAH BLAH CLOUD. BLAH BLAH CLOUD.
© 2011 Scott Adams, Inc./Dist. by UFS, Inc.
IT'S AS IF YOU'RE A TECHNOLOGIST AND A PHILOSOPHER ALL IN ONE!
BLAH BLAH PLATFORM.

What is the use case

- Use Case Examples
  - Index your website documents and make them available for search.
  - Search log files that have been uploaded and parsed by AWS DataPipeline.
  - Make common office documents searchable to allow discovery of organizational knowledge.
  - Combine, files and database records into an index to allow for faceted search experience a la Amazon.

- Technical Requirements
  - Highly reliable
  - Good set of base features
  - Auto-Scalable

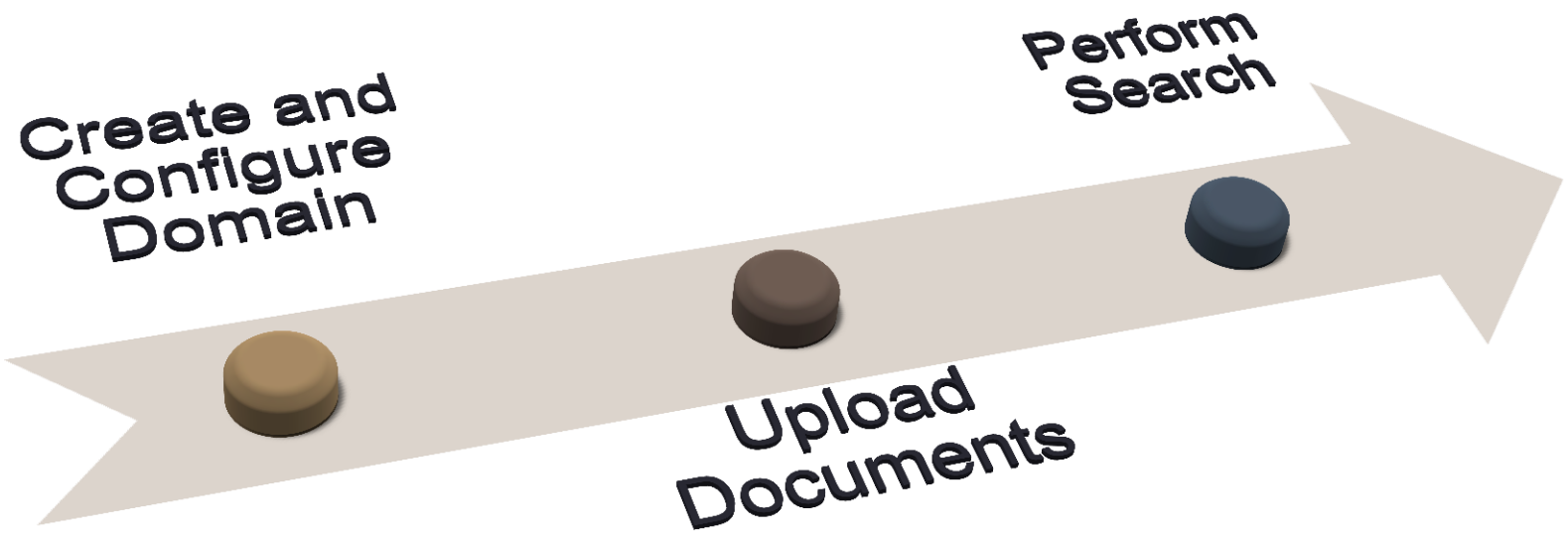
What is AWS CloudSearch

"Amazon CloudSearch is a **fully-managed** service in the cloud that makes it **easy** to set up, manage, and **scale** a search solution for your website. Amazon CloudSearch enables you to search **large** collections of data such as web pages, document files, forum posts, or product information. …Amazon CloudSearch **automatically scales** to meet your needs."

Some other noted features

- Simple to configure
- Automatic scale for data and traffic
- Low latency, high throughput
- Easy Admin
- Rich Search Features
- Low Costs
- Secure
- Structured and unstructured data
- Based on A9 Search (not Lucene)

Basic getting started

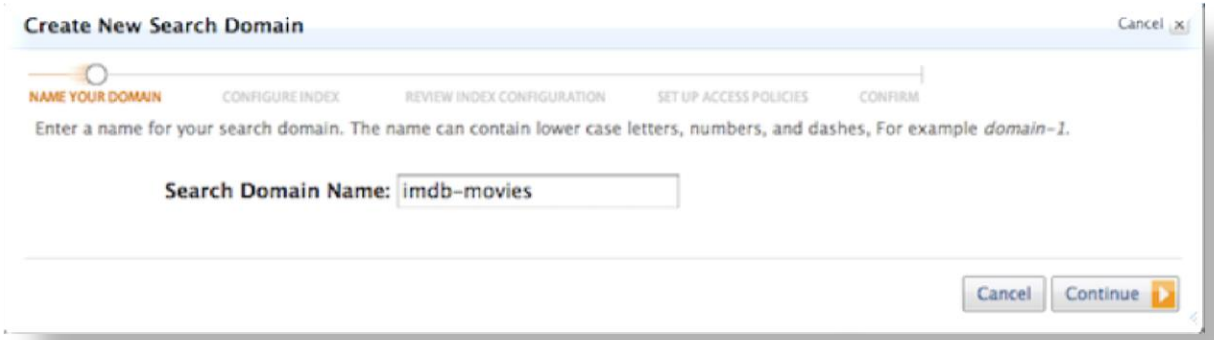


1. Create Domain

- Three sub steps:
  - A) Create Domain Name
  - B) Define Index
  - C) Setup Access Policies

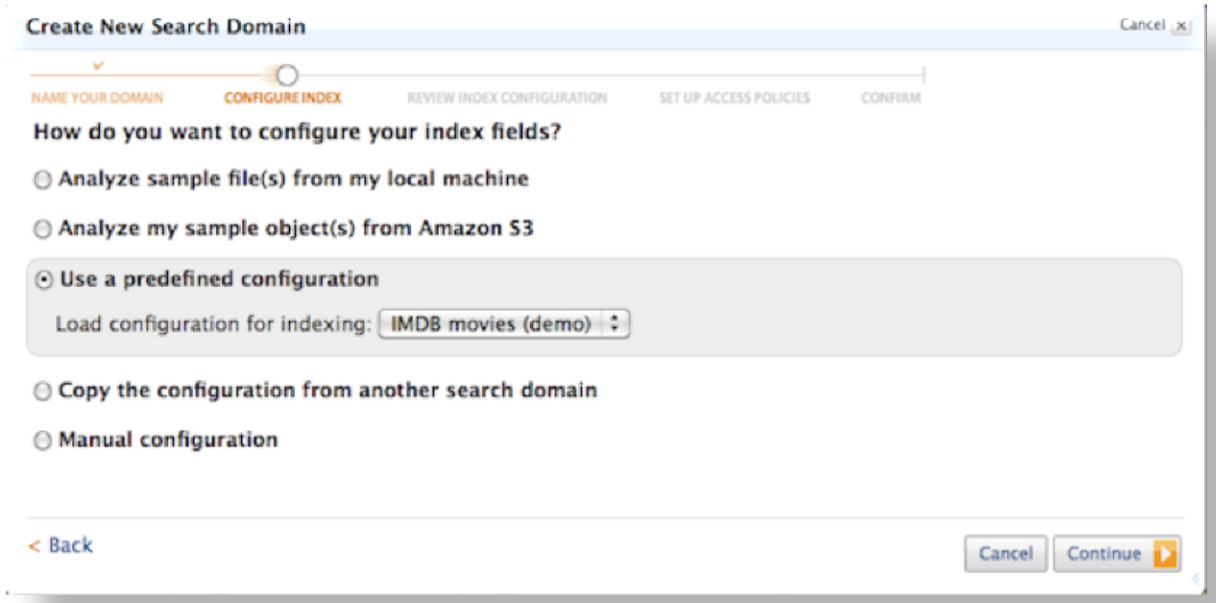
1A. Create Domain

- Use console or API



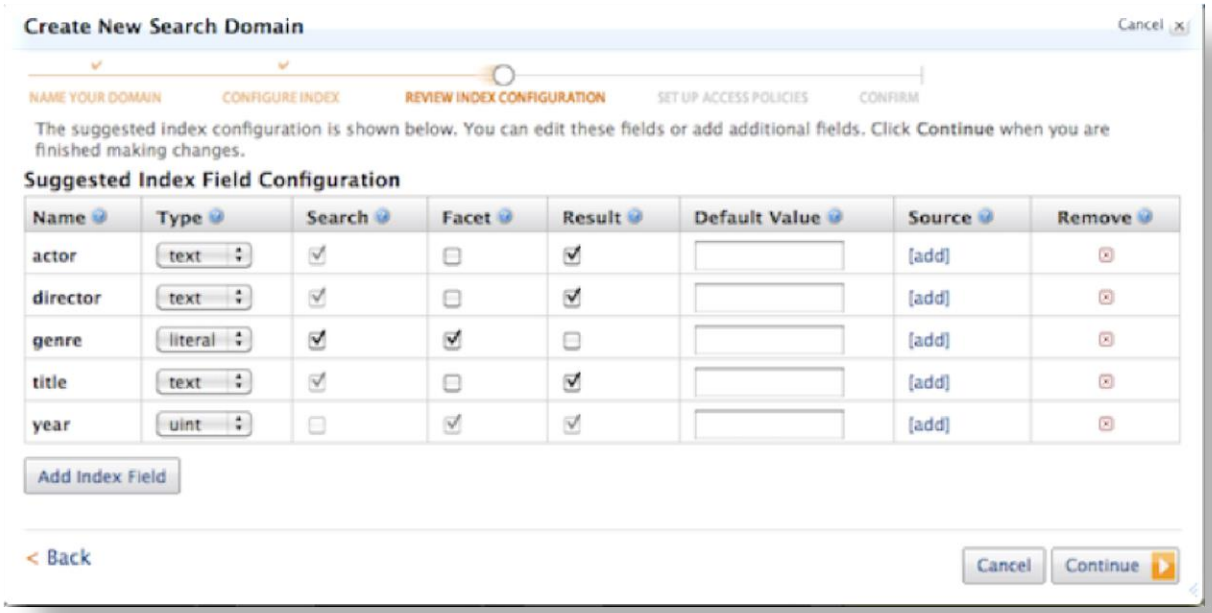
1B. Define index via console

- From sample file (json,xml,csv,pdf,htm,xls,ppt,doc,txt,json,xml)
- Predefined (optimized for task)
- Copy from other domain
- Manual



Define/View Index Fields

- This is where you refine what can be searched vs what is display only. Any change later will require complete rebuild of index.



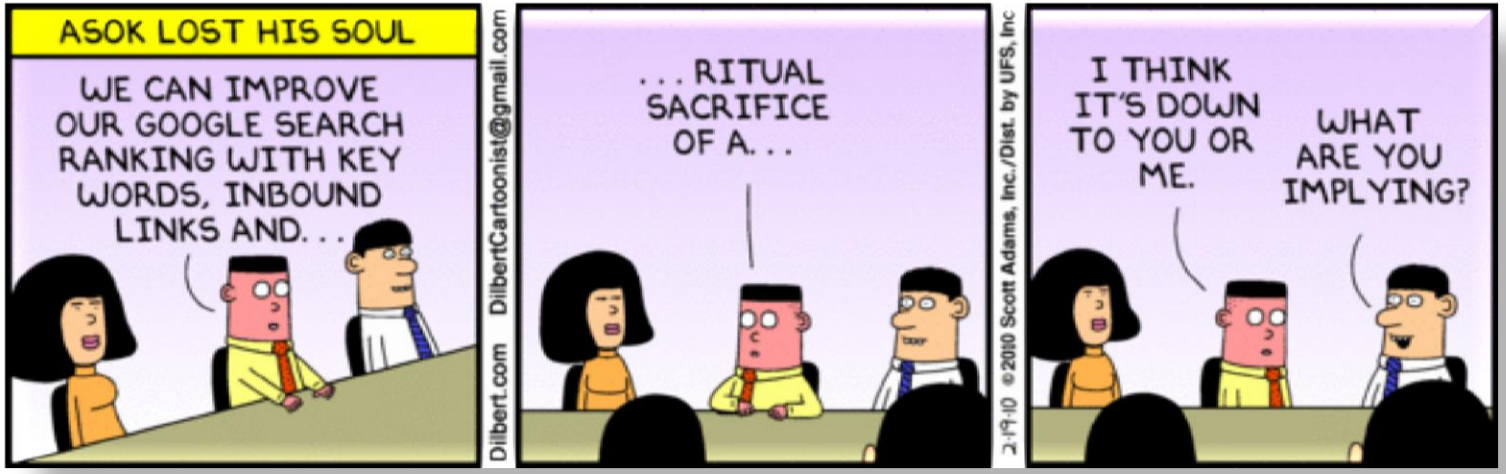
Index Field Elements in Console

- Name: What it says, should match your source name
- Status: Tells you field level info: active (can be searched), pending, pending deletion, being deleted
- Type: uint, literal (id or exact match criteria), text
- Search: can we search for this field's data
- Facet: can we build facets using this field's data
- Result: Is this field returned with the result package
- Default Value: what it says
- Source: Mapping/Copying/Transform options

Guide to Index Fields

- **Uint: Does the field contain numeric values?**
  - **If you can convert it to to unsigned int (uint) you can construct rank expressions and range searches.**
- **Text: Do you need to search for individual words within the field?**
  - Text fields are used for free text searches of data such as names, descriptions, or even the entire body of a document.
- **Literal: Does the field contain string values that you want to match exactly or use as facets?**
  - Literal fields are often used for fields that have a small set of possible values, as well as for more arbitrary values like email addresses or brand names where an exact match is important. Literal fields are frequently used to enable faceted searches where you want to count the number of exact matches for a particular value.

Search is a challenge for all



The SDF file

- The heart of your index is your Search Definition File (SDF)
  - Create 5MB sized batches (1 doc cannot be larger than 1MB)
  - JSON or XML
  - UTF-8 only (Unicode and ISO/IEC 10646. FFFE, FFFF, and the surrogate blocks D800DBFF and DC00DFFF are invalid) caused me many headaches.
- For each document
  - Type -- (action type): add (=update with diff version), delete
  - Id -- unique doc id
  - Version -- incremented version number (unix time, 32bit only)
  - Lang -- (en is the only supported one)
  - Fields – name / value pairs

SDF Example

```
[
 {
    "type": "add",
    "id": "sogqtrzl2a8cl3c8b0",
    "version": 1,
    "lang": "en",
    "fields": {
      "title": "I Can't Stop Loving You",
      description: "A country standard performed live
        by Martina McBride."
      "artist_name": "Martina McBride",
      "year": 2005,
      "price": 100,
      "genre": ["country",  "pop",  "ballad"]
    }
  },
  {
    "type": "add",
    "id": "sobhvzql2ac9618285",
    "version": 1,
    "lang": "en",
    "fields": {
      "title": "I'm Gonna Love You Through It",
      description: "An emotional track written by
        Ben Hayslip, Jimmy Yeary and his
        wife, Sonya Isaacs."
      "artist_name": "Martina McBride",
      "year": 2011,
      "price": 100,
      "genre": [ "country",  "pop",  "ballad"]
    }
  }
]
```

Doing the Searches

- REST based API
  - Endpoint is: [Endpoint] + *"2011-02-01/search?"*
- The Query Syntax
  - q=[keywords]
  - bq= (boolean query)
  - Samples
    - q=star|wars matches movies that contain either *star* or *wars* in the default search field.
    - bq=title:'story funny|underdog' matches movies that contain both the terms *story* and *funny* or the term underdog in the title field.
    - bq=title:'red|white|blue' matches movies that contain either *red*, *white*, or *blue* in the title field.
    - bq=actor:'"evans, chris"|"Garity, Troy"' matches movies that contain either the phrase *evans, chris* or the phrase *Garity, Troy* in the actor field.
    - bq='title:-star+war|world' matches movies whose titles do not contain *star*, but do contain either *war* or *world*.
- Ranking
  - rank-expression1=(formula) etc.
- Return Fields (otherwise only hit id and relevance is returned)
  - fields=actor,title,text_relevance

Search Results

- JSON or XML

```
search?q=star+wars&return-fields=actor,title,text_relevance
```


```
{
    "id":"tt1185834",
    "data":{
        "actor":["Abercrombie, Ian","Baker, Dee Bradley","Burton, Corey",
                 "Eckstein, Ashley","Futterman, Nika","Kane, Tom",
                 "Lanter, Matt","Taber, Catherine","Taylor, James Arnold",
                 "Wood, Matthew"],
        "text_relevance":["308"],
        "title":["Star Wars: The Clone Wars"]
    }
}
```

The Dark Side

- Maintenance
  - Versioning (this is a killer)
  - Updating
  - Deleting
  - Knowing or Saving document Id (do not forget keys)

- Permissions
  - Not granular (IP only for search)
  - Full root access to manage index (scary)

What about the cost

Region: US East (N. Virginia)

Search Instance Type	Pricing
Small Search Instance	$0.10 per hour
Large Search Instance	$0.39 per hour
Extra Large Search Instance	$0.55 per hour

More on Cost

- No reservations
- Cannot determine size, it is sized as Amazon sees fit
- Min $72/month

Let’s put it all together

- Using the Command Line Tools
  - Create Index
  - Add Entries
  - Delete from Index
- Using Console
  - Search and Query language
- Using Ajax
  - Geospatial Search Example
    - Translate and store latitude and longitude in linear units as uint

What I don’t like

- No easy way to just clear the index, you will need the document Ids (and versions).
- Cannot auto maintain anything, e.g. connect to S3 bucket and suck up the differences.
- Relatively expensive even for small indexes.
- Complete index rebuilds even for minor changes.
- Search language is not compatible with SOLR
- Access policies are basic
- To do any index tasks root access is needed

What I like

- Easy to add docs of different kinds
- Query language and options are good (for me)
- XML or JSON
- Fast Responses
- It auto-scales people!

Summary

- You can build a nice, scalable search service
- You need to plan versioning from beginning
- Keep track of them Ids
- Nothing is auto mode: Plan to do maintenance

&

We always look for good people to join our team.

On that note…


HI, MY NAME IS. . .
DON'T BOTHER.
CLICK
Dilbert.com DilbertCartoonist@gmail.com
MY APP DOES FACIAL RECOGNITION AND SEARCHES ALL SOCIAL MEDIA TO GIVE ME YOUR FULL BIOGRAPHY.
12-17-10 ©2010 Scott Adams, Inc./Dist. by UFS, Inc.
HOW'S THAT WORKING OUT?
YOU'RE EITHER BART SIMPSON OR A HUGE DRY-ERASE MARKER.

Next Meeting

- When: July 25th
- Where: Packard Place
- Who: Addy
- What: Deep dive into network and application monitoring

THANK YOU

@BmanClt
http://BonCode.blogspot.com