



Masterclass

....

Amazon DynamoDB

Ian Massingham – Technical Evangelist

 @ianMmmm

Masterclass

A technical deep dive beyond the basics

Help educate you on how to get the best from AWS technologies

Show you how things work and how to get things done

Broaden your knowledge in ~45 mins

Amazon DynamoDB

A fast, fully managed NoSQL database service

Simple & cost effective to store and retrieve any amount of data

Reliable throughput & single-digit millisecond latency

Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,
Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall
and Werner Vogels

Amazon.com

ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems.

This paper presents the design and implementation of Dynamo, a highly available key-value storage system that some of Amazon's core services use to provide an "always-on" experience. To achieve this level of availability, Dynamo sacrifices consistency under certain failure scenarios. It makes extensive use of object versioning and application-assisted conflict resolution in a manner that provides a novel interface for developers to use.

Categories and Subject Descriptors

D.4.2 [Operating Systems]: Storage Management; D.4.5 [Operating Systems]: Reliability; D.4.2 [Operating Systems]: Performance;

General Terms

Algorithms, Management, Measurement, Performance, Design, Reliability.

1. INTRODUCTION

Amazon runs a world-wide e-commerce platform that serves tens of millions customers at peak times using tens of thousands of servers located in many data centers around the world. There are strict operational requirements on Amazon's platform in terms of performance, reliability and efficiency, and to support continuous growth the platform needs to be highly scalable. Reliability is one of the most important requirements because even the slightest

One of the lessons our organization has learned from operating Amazon's platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. Therefore, the service responsible for managing shopping carts requires that it can always write to and read from its data store, and that its data needs to be available across multiple data centers.

Dealing with failures in an infrastructure comprised of millions of components is our standard mode of operation; there are always a small but significant number of server and network components that are failing at any given time. As such Amazon's software systems need to be constructed in a manner that treats failure handling as the normal case without impacting availability or performance.

To meet the reliability and scaling needs, Amazon has developed a number of storage technologies, of which the Amazon Simple Storage Service (also available outside of Amazon and known as Amazon S3), is probably the best known. This paper presents the design and implementation of Dynamo, another highly available and scalable distributed data store built for Amazon's platform. Dynamo is used to manage the state of services that have very high reliability requirements and need tight control over the tradeoffs between availability, consistency, cost-effectiveness and performance. Amazon's platform has a very diverse set of applications with different storage requirements. A select set of applications require a storage technology that is flexible enough to let application designers configure their data store appropriately based on these tradeoffs to achieve high availability and guaranteed performance in the most cost effective manner.

There are many services on Amazon's platform that only need

Easy Administration

Fast, Predictable Performance

DynamoDB

Scalable

Built-in Fault Tolerance



Getting Started



Provisioned Throughput



Data Modeling



Programming DynamoDB



Partitioning



Working with the AWS CLI and SDKs



Reporting & Analysis



Getting Started

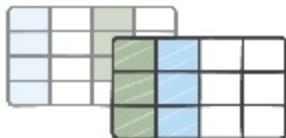
DynamoDB is a managed NoSQL database service

Designed to store and retrieve any amount of data &
to serve any level of request traffic

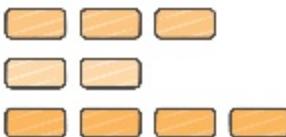
Getting Started



Seamless Scaling



Secondary Indexes



Schema-less



Strong Consistency, Atomic Counters

Getting Started

Durability

Consistent, disk only writes

Replication across data centers and availability zones

Creating a DynamoDB table via the AWS Console

AWS Management Console

Services VPC EC2 S3 RDS CloudFront Edit Janmas @ Janmas-aws Ireland Help

Amazon Web Services

Compute & Networking

- Direct Connect
- EC2 Virtual Servers in the Cloud
- Route 53 Scalable Domain Name System
- VPC Isolated Cloud Resources

Storage & Content Delivery

- CloudFront Global Content Delivery Network
- Glacier Archive Storage in the Cloud
- S3 Scalable Storage in the Cloud
- Storage Gateway Integrates On-Premise IT Environments with Cloud Storage

Database

- DynamoDB Predictable and Scalable NoSQL Data Store
- ElastiCache In-Memory Cache
- RDS Managed Relational Database Service
- Redshift Managed Petabyte-Scale Data Warehouse Service

Deployment & Management

- CloudFormation Templated AWS Resource Creation
- CloudTrail User Activity and Change Tracking
- CloudWatch Resource and Application Monitoring
- Elastic Beanstalk AWS Application Container
- IAM Secure AWS Access Control
- OpsWorks DevOps Application Management Service

Analytics

- Data Pipeline Orchestration for Data-Driven Workflows
- Elastic MapReduce Managed Hadoop Framework
- Kinesis Real-time Processing of Streaming Big Data

Mobile Services

- Cognito User Identity and App Data Synchronization
- Mobile Analytics Understand App Usage Data at Scale
- SNS Push Notification Service

App Services

- AppStream Low Latency Application Streaming
- CloudSearch Managed Search Service
- Elastic Transcoder Every-Frame Scalable Media Transcoding
- SES Email Sending Service
- SQS Message Queue Service
- SWF Workflow Service for Coordinating Application Components

Applications

- WorkSpaces Desktops in the Cloud
- Zocalo Secure Enterprise Storage and Sharing Service

Additional Resources

Getting Started See our documentation to get started and learn more about how to use our services.

Trusted Advisor Best practice recommendations to save money, improve fault tolerance, increase performance, and close security gaps.

Service Health

All services operating normally.

Updated: Jul 28 2014 14:05:00 GMT+100

Service Health Dashboard

Set Start Page

Console Home

AWS Marketplace Find & buy software, launch with 1-click and pay by the hour.

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

Creating a DynamoDB table via the AWS Console

DynamoDB · AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Ianmas @ Ianmas-aws Ireland Help

Amazon DynamoDB Getting Started

Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability. Learn More about Amazon DynamoDB.

To start using Amazon DynamoDB, create a table

Create Table

Note: Your table will be created in the EU West (Ireland) region.

How do I create a table?

- 1 Pick Primary Key.
- 2 Set Provisioned Throughput.
- 3 Create your table with alarms.

Learn More Learn More Learn More

Additional Resources

- Getting Started Guide
- FAQ
- Release Notes
- Developer Guide
- Forums
- Report an Issue

Watch the video

Learn more about how to setup and use DynamoDB databases.

Amazon DynamoDB Overview, a fully ma... 

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

Selecting the Primary Key Type

DynamoDB · AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Ianmas @ Ianmas-aws Ireland Help

Amazon DynamoDB Getting Started

Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability. Learn More about Amazon DynamoDB.

To start using Amazon DynamoDB, create a table

Create Table

PRIMARY KEY ADD INDEXES (optional) PROVISIONED THROUGHPUT CAPACITY THROUGHPUT ALARMS (optional) SUMMARY

Table Name: enter table name...
Table will be created in eu-west-1 region

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: Hash and Range Hash

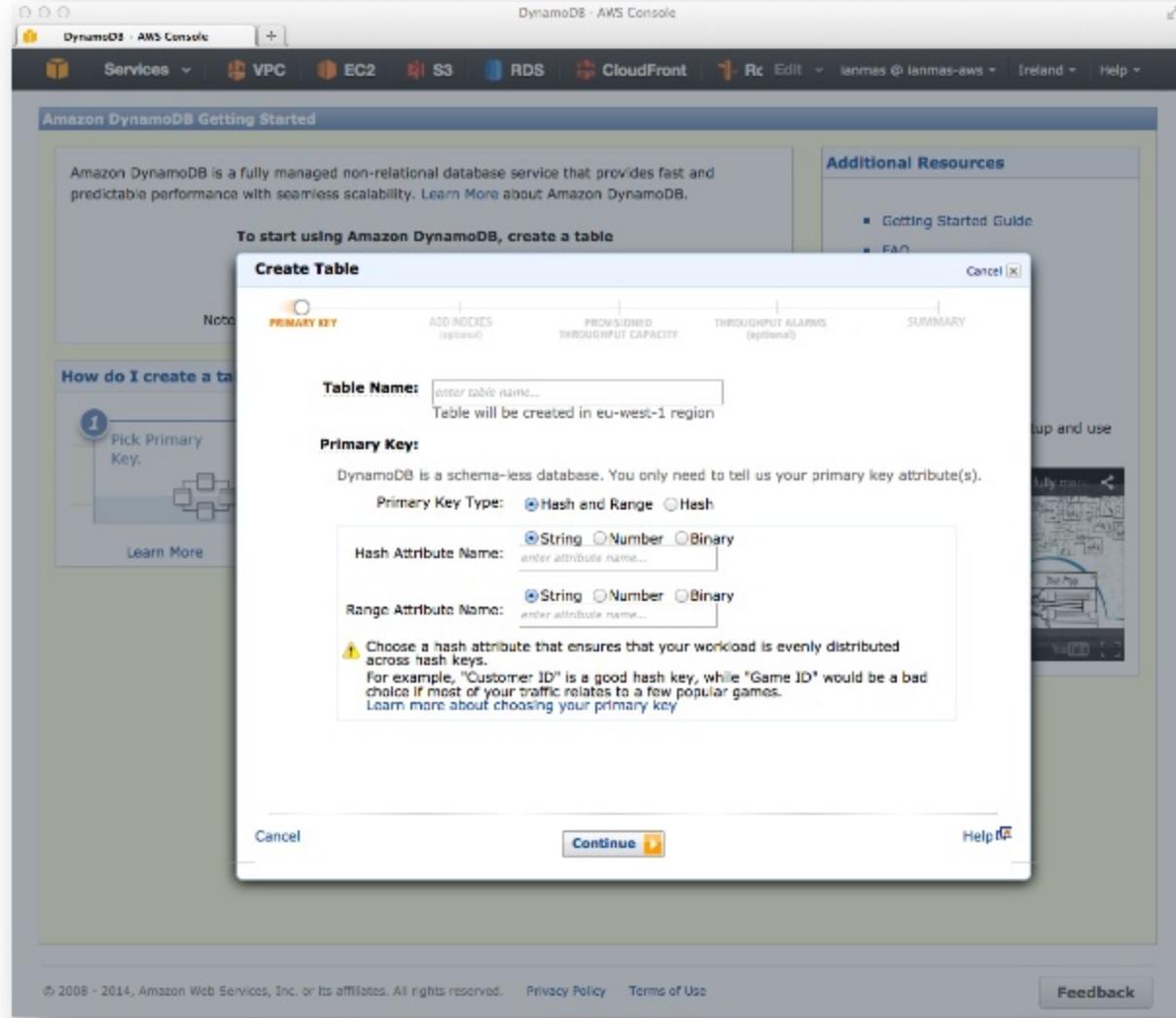
Hash Attribute Name: enter attribute name...
 String Number Binary

Range Attribute Name: enter attribute name...
 String Number Binary

Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.
For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.
Learn more about choosing your primary key

Cancel Continue Help

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback



Specifying a Hash Primary Key Type

DynamoDB · AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Ianmas @ Ianmas-aws Ireland Help

Amazon DynamoDB Getting Started

Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability. Learn More about Amazon DynamoDB.

To start using Amazon DynamoDB, create a table

Create Table

PRIMARY KEY ADD INDEXES (optional) PROVISIONED THROUGHPUT CAPACITY THROUGHPUT ALARMS (optional) SUMMARY

Table Name: Forum
Table will be created in eu-west-1 region

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: Hash and Range Hash

Hash Attribute Name: String Number Binary

Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.
For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.
[Learn more about choosing your primary key](#)

Cancel Continue Help

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

Additional Resources

- Getting Started Guide
- FAQ

Add optional Indexes to the new table

DynamoDB · AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Ianmas @ Ianmas-aws Ireland Help

Amazon DynamoDB Getting Started

Create Table

Additional Resources Cancel X

PREVIOUS STEP | NEXT STEP | BACK | FORWARD | HELP

How To Create a Table

1. Add Primary Key

2. Add Indexes (optional)

3. Provisioned Throughput Capacity

4. Throughput Alarms (optional)

5. Summary

Add Indexes (optional)

An index is a data structure that maintains an alternate hash and range key. You can use it to Query an item the same way you use the table hash and range key.

Index Type: Global Secondary Index

Index Hash Key: String

Index Range Key: String

Index Name:

Projected Attributes: All Attributes

Add Index To Table

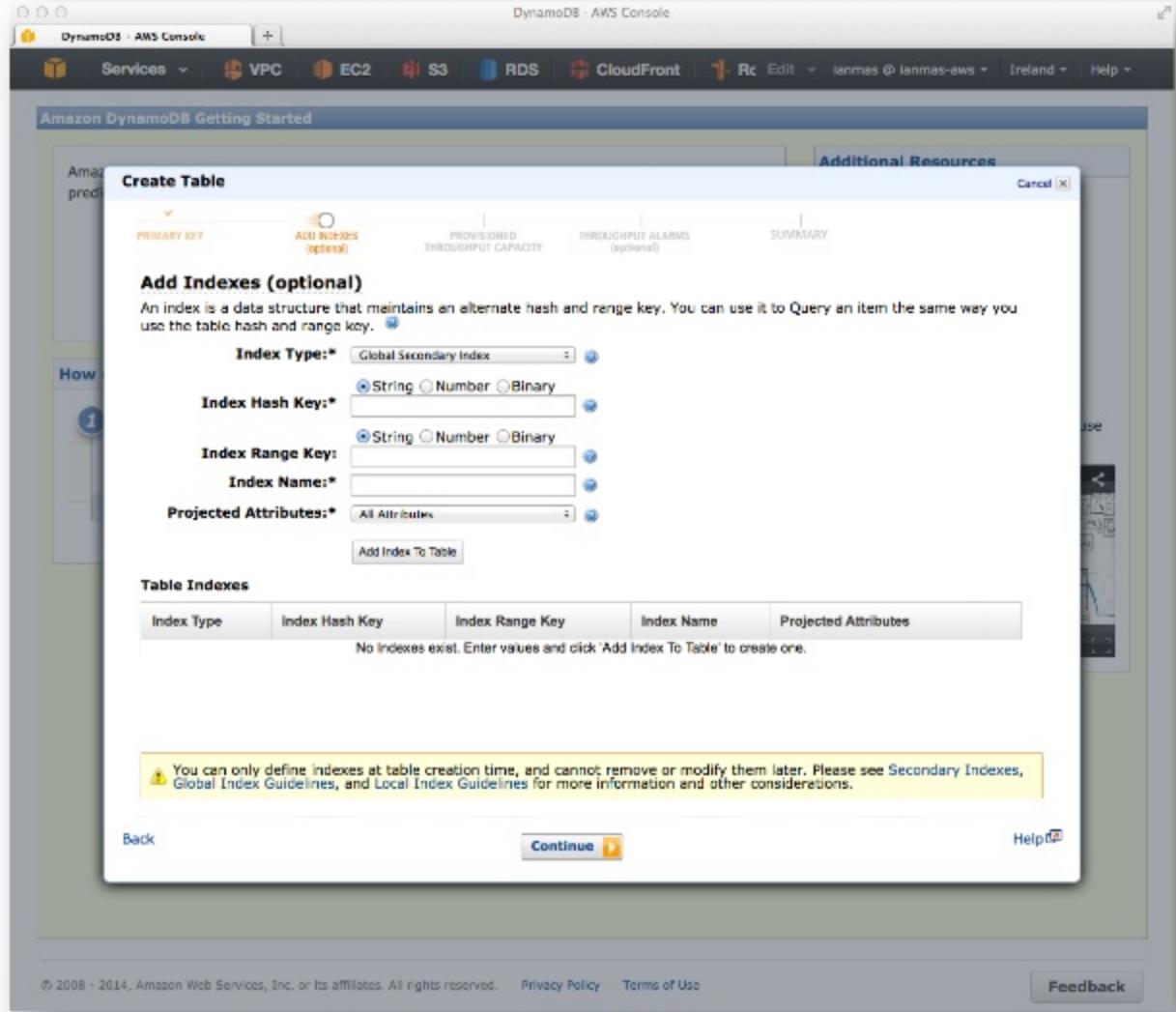
Table Indexes

Index Type	Index Hash Key	Index Range Key	Index Name	Projected Attributes
No indexes exist. Enter values and click 'Add Index To Table' to create one.				

You can only define indexes at table creation time, and cannot remove or modify them later. Please see Secondary Indexes, Global Index Guidelines, and Local Index Guidelines for more information and other considerations.

Back Continue Help

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback



Specify Provisioned Throughput Capacity

DynamoDB - AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Janneke @ Janneke-aws Ireland Help

Amazon DynamoDB Getting Started

Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability. Learn More about Amazon DynamoDB.

Additional Resources

Create Table

PRIMARY KEY ADD INDEXES PROVISIONED THROUGHPUT CAPACITY THROUGHPUT ALARMS SUMMARY

Provisioned Throughput Capacity:

Help me calculate how much throughput capacity I need to provision

Throughput capacity to provision:

Amazon DynamoDB lets you specify how much read and write throughput capacity you wish to provision for your table. Using this information, Amazon will provision the appropriate resources to meet your throughput needs. [More Information](#)

Read Capacity Units: 1

Write Capacity Units: 1

⚠️ Throughput capacity for this table will cost up to \$0.66 per month if you have exceeded the free tier.
*Taxes may apply.

⚠️ If you exceed the free tier you are charged for the provisioned throughput capacity of your table even if you do not actively use your provisioned capacity. Learn more about DynamoDB's free tier and pricing.

Back Continue Help

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

The screenshot shows the 'Create Table' wizard in the AWS DynamoDB console. The current step is 'PROVISIONED THROUGHPUT CAPACITY'. It has two input fields: 'Read Capacity Units' set to 1 and 'Write Capacity Units' set to 1. Below these fields are two yellow warning boxes. The first box states: 'Throughput capacity for this table will cost up to \$0.66 per month if you have exceeded the free tier.' followed by a note: '*Taxes may apply.'. The second box states: 'If you exceed the free tier you are charged for the provisioned throughput capacity of your table even if you do not actively use your provisioned capacity. Learn more about DynamoDB's free tier and pricing.' At the bottom of the wizard are 'Back', 'Continue', and 'Help' buttons.

Add optional Throughput Alarms

DynamoDB - AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Janneke @ Janneke-aws Ireland Help

Amazon DynamoDB Getting Started

Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability. Learn More about Amazon DynamoDB.

To start using Amazon DynamoDB, create a table

Create Table

PRIMARY KEY AUTO INCREASES (optional) PROVISIONED THROUGHPUT CAPACITY THROUGHPUT ALARMS (optional) SUMMARY

Throughput Alarms (optional)

Use Basic Alarms

Notify me when my table's request rates exceed 80% of Provisioned Throughput for 60 minutes.

Notification will be sent when:

- Read Capacity Units consumed > 8
- or
- Write Capacity Units consumed >4

Send notification to:

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch Management Console.

Back Continue Help

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

The screenshot shows the 'Create Table' wizard in the AWS DynamoDB console. The 'Throughput Alarms' section is open, displaying basic alarm configuration options. The 'Use Basic Alarms' checkbox is selected, and the threshold is set to 80% of provisioned throughput for a 60-minute period. The notification will be sent when either read or write capacity units are consumed above these thresholds. A text input field is provided for sending notifications to a specific recipient. Below the form, a note states that additional charges apply for exceeding AWS Free Tier levels for CloudWatch or Simple Notification Service, with advanced settings available in the CloudWatch Management Console. Navigation buttons for 'Back', 'Continue', and 'Help' are at the bottom, along with standard AWS footer links for privacy and terms of use.

Review

DynamoDB - AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Janneke @ Janneke-aws Ireland Help

Amazon DynamoDB Getting Started

Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability. Learn More about Amazon DynamoDB.

To start using Amazon DynamoDB, create a table

Create Table

Cancel

PRIMARY KEY ADD INDEXES (optional) PROVISIONED THROUGHPUT CAPACITY THROUGHPUT ALARMS (optional) SUMMARY

Review

Review the specifications for the table. Be aware that the hash key, range key, and local secondary index details cannot be changed after the table is created.

Table Name: Forum

Primary Key Type: Hash

Hash Key Attribute: Name (String)

Provisioned Read Throughput: 10 units

Provisioned Write Throughput: 5 units

Estimated Provisioned Throughput Cost: \$3.83 / month *Taxes may apply.

Use Basic Alarms: No

Indexes:

Index Type	Index Hash Key	Index Range Key	Index Name	Projected Attributes
This table will not have any indexes.				

Back Create Help

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

Confirmation that
the new table is
now active

DynamoDB - AWS Console

Services VPC EC2 S3 RDS CloudFront Re Edit Ianmas @ Ianmas-aws Ireland Help

Amazon DynamoDB Tables

Name	Status	Hash Key	Range Key	Read Throughput	Write Throughput
Forum	ACTIVE	Name	-	10	5

Table Items

Details Indexes Monitoring Alarm Setup

CloudWatch Alarms: No alarms configured Create Alarm

Alarm History (past 24 hours): No alarms triggered

Time Range: Last Hour

Below are your CloudWatch metrics for the selected resources. Click on a graph to see an expanded view. > View all CloudWatch metrics

Read Capacity Units/Second - 5 Minute Average Throttled Read Requests Count

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Feedback

The screenshot shows the AWS DynamoDB console interface. At the top, there's a navigation bar with links for Services, VPC, EC2, S3, RDS, CloudFront, Re Edit, and user information (Ianmas @ Ianmas-aws, Ireland, Help). Below the navigation is a search bar labeled 'Filter:' and several buttons: Explore Table, Create Table, Modify Throughput, Delete Table, Export / Import, and Access Control. A message '1 to 1 of 1 tables' is displayed. The main area is titled 'Amazon DynamoDB Tables' and contains a table with one row. The table columns are Name, Status, Hash Key, Range Key, Read Throughput, and Write Throughput. The single row shows 'Forum' as the Name, 'ACTIVE' as the Status, 'Name' as the Hash Key, '-' as the Range Key, '10' as the Read Throughput, and '5' as the Write Throughput. Below the table is a 'Monitoring' section titled 'Table Items'. It has tabs for Details, Indexes, Monitoring (which is selected), and Alarm Setup. Under Monitoring, there are sections for 'CloudWatch Alarms' (showing 'No alarms configured') and 'Alarm History (past 24 hours)' (showing 'No alarms triggered'). A 'Create Alarm' button is also present. At the bottom of the monitoring section, there's a 'Time Range' dropdown set to 'Last Hour'. Further down, a note says 'Below are your CloudWatch metrics for the selected resources. Click on a graph to see an expanded view. > View all CloudWatch metrics'. Two metrics are listed: 'Read Capacity Units/Second - 5 Minute Average' and 'Throttled Read Requests Count'. The footer of the page includes copyright information (© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.) and links for Privacy Policy and Terms of Use, along with a Feedback button.

Getting Started

Ready for use with just three clicks,
but you do have to make two decisions first.

Getting Started

Ready for use with just three clicks,
but you do have to make two decisions first.

Getting Started

Ready for use with just three clicks,
but you do have to make **two decisions** first.

Level of throughput

Primary keys



Provisioned Throughput

Reserve IOPS for reads and writes

Scale up or down at any time

Pay per capacity unit

Priced per hour of provisioned throughput

Write throughput

Size of item x writes per second

\$0.0065 for 10 write units

Consistent writes

Atomic increment and decrement.

Optimistic concurrency control: conditional writes.

Transactions

Item level transactions only

Puts, updates and deletes are ACID

Strong or eventual consistency

Read throughput

Strong or eventual consistency

Read throughput

Provisioned units = (size of item / 4) x reads per second

\$0.0065 per hour for 50 units

Strong or eventual consistency

Read throughput

$$\text{Provisioned units} = \frac{(\text{size of item} / 4) \times \text{reads per second}}{2}$$

\$0.0065 per hour for 100 units

Strong or eventual consistency

Read throughput

Same latency expectations.

Mix and match at ‘read time’

Provisioned throughput is
managed by DynamoDB

Data is partitioned and
managed by DynamoDB

Reserved capacity

Up to 53% for 1 year reservation.

Up to 76% for 3 year reservation.

Authentication

Session based to minimize latency

Uses the Amazon Security Token Service

Handled by AWS SDKs

Integrates with IAM

Monitoring

CloudWatch metrics: latency, consumed read
and write throughput, errors & throttling

Data Modeling

Data Modeling

id = 100	date = 2012-05-16-09-00-10	total = 25.00
id = 101	date = 2012-05-15-15-00-11	total = 35.00
id = 101	date = 2012-05-16-12-00-10	total = 100.00
id = 102	date = 2012-03-20-18-23-10	total = 20.00
id = 102	date = 2012-03-20-18-23-10	total = 120.00

Data Modeling

Table

id = 100	date = 2012-05-16-09-00-10	total = 25.00
id = 101	date = 2012-05-15-15-00-11	total = 35.00
id = 101	date = 2012-05-16-12-00-10	total = 100.00
id = 102	date = 2012-03-20-18-23-10	total = 20.00
id = 102	date = 2012-03-20-18-23-10	total = 120.00

Data Modeling

id = 100	date = 2012-05-16-09-00-10	total = 25.00	Item
id = 101	date = 2012-05-15-15-00-11	total = 35.00	
id = 101	date = 2012-05-16-12-00-10	total = 100.00	
id = 102	date = 2012-03-20-18-23-10	total = 20.00	
id = 102	date = 2012-03-20-18-23-10	total = 120.00	

Data Modeling

		Attribute
id = 100	date = 2012-05-16-09-00-10	total = 25.00
id = 101	date = 2012-05-15-15-00-11	total = 35.00
id = 101	date = 2012-05-16-12-00-10	total = 100.00
id = 102	date = 2012-03-20-18-23-10	total = 20.00
id = 102	date = 2012-03-20-18-23-10	total = 120.00

Where is the schema?

Tables do not require a formal schema

Items are an arbitrarily sized hash

Indexing

Items are indexed by primary and secondary keys

Primary keys can be composite

Secondary keys index on other attributes

Data Modeling



id = 100	date = 2012-05-16-09-00-10	total = 25.00
id = 101	date = 2012-05-15-15-00-11	total = 35.00
id = 101	date = 2012-05-16-12-00-10	total = 100.00
id = 102	date = 2012-03-20-18-23-10	total = 20.00
id = 102	date = 2012-03-20-18-23-10	total = 120.00

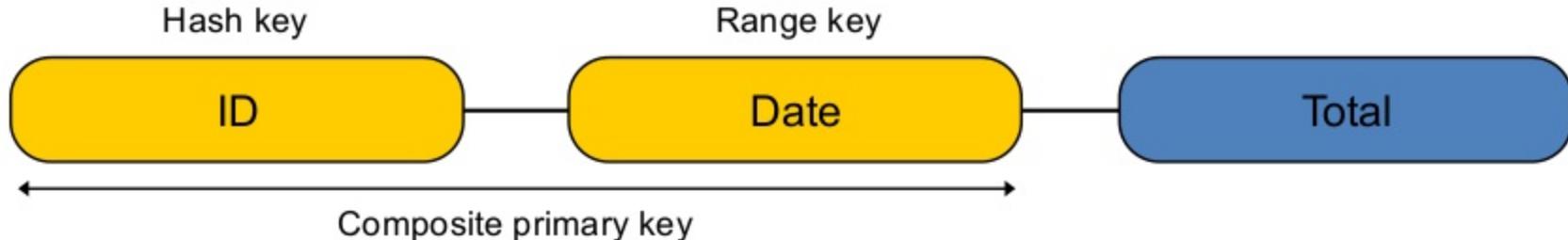
Data Modeling

Hash key



id = 100	date = 2012-05-16-09-00-10	total = 25.00
id = 101	date = 2012-05-15-15-00-11	total = 35.00
id = 101	date = 2012-05-16-12-00-10	total = 100.00
id = 102	date = 2012-03-20-18-23-10	total = 20.00
id = 102	date = 2012-03-20-18-23-10	total = 120.00

Data Modeling



id = 100	date = 2012-05-16-09-00-10	total = 25.00
id = 101	date = 2012-05-15-15-00-11	total = 35.00
id = 101	date = 2012-05-16-12-00-10	total = 100.00
id = 102	date = 2012-03-20-18-23-10	total = 20.00
id = 102	date = 2012-03-20-18-23-10	total = 120.00

Data Modeling



id = 100	date = 2012-05-16-09-00-10	total = 25.00
id = 101	date = 2012-05-15-15-00-11	total = 35.00
id = 101	date = 2012-05-16-12-00-10	total = 100.00
id = 102	date = 2012-03-20-18-23-10	total = 20.00
id = 102	date = 2012-03-20-18-23-10	total = 120.00

Programming

DynamoDB

CreateTable	PutItem
UpdateTable	GetItem
DeleteTable	UpdateItem
DescribeTable	DeleteItem
ListTables	BatchGetItem
Query	BatchWriteItem
Scan	

CreateTable

PutItem

UpdateTable

GetItem

DeleteTable

UpdateItem

DescribeTable

DeleteItem

ListTables

BatchGetItem

Query

BatchWriteItem

Scan

CreateTable

PutItem

UpdateTable

GetItem

DeleteTable

UpdateItem

DescribeTable

DeleteItem

ListTables

BatchGetItem

Query

BatchWriteItem

Scan

Conditional updates

PutItem, UpdateItem, DeleteItem can take optional conditions for operation.

UpdateItem performs atomic increments.

One API call, multiple items

BatchGet returns multiple items by key.

BatchWrite performs up to 25 put or delete operations.

Throughput is measured by IO, not API calls.

CreateTable

PutItem

UpdateTable

GetItem

DeleteTable

UpdateItem

DescribeTable

DeleteItem

ListTables

BatchGetItem

Query

BatchWriteItem

Scan

Query vs Scan

Query for Composite Key queries.

Scan for full table scans, exports.

Both support pages and limits.

Maximum response is 1Mb in size.

Query patterns

Retrieve all items by **hash key**

Range key conditions: ==, <, >, >=, <=, begins with, between

Counts, Top and bottom n values, Paged responses

EXAMPLE 1:

Relationships & Secondary Indexes

Social Network

hash + range schemas

Social Network

- Store info about users
- Query for a user's friends

Social Network

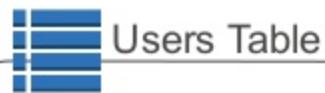


Friends Table



Users Table

Social Network



Users Table

User	Nicknames
Bob	[Rob, Bobby]
Alice	[Allie]
Carol	[Caroline]
Dan	[Daniel, Danny]

Social Network

Item →

Users Table

User	Nicknames
Bob	[Rob, Bobby]
Alice	[Allie]
Carol	[Caroline]
Dan	[Daniel, Danny]

Social Network

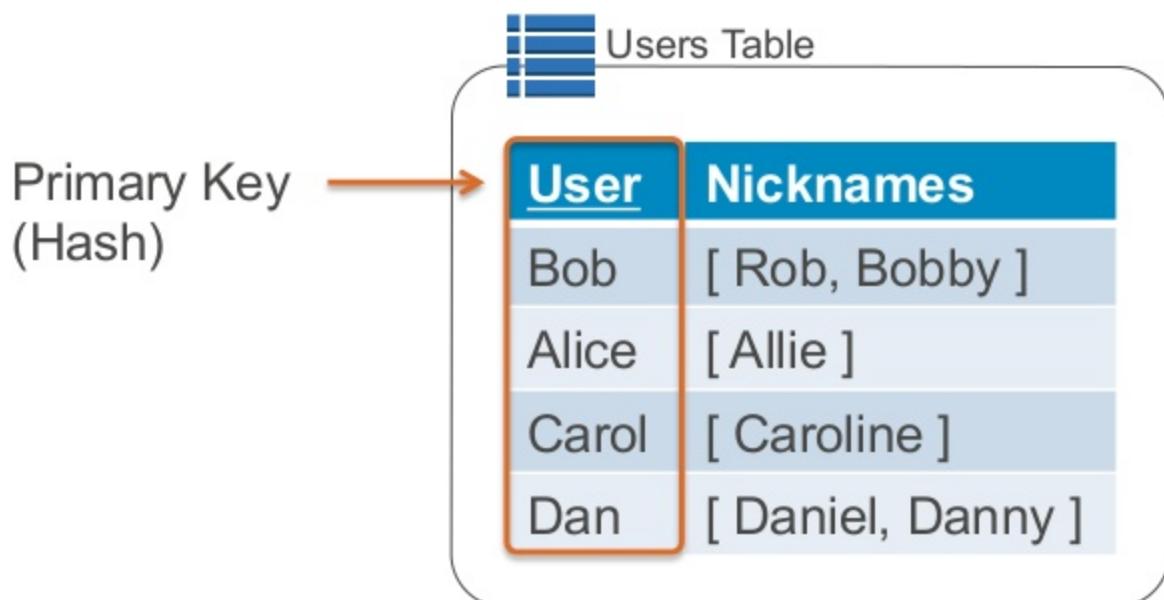
Attribute
(string, number, binary, set)

Users Table



User	Nicknames
Bob	[Rob, Bobby]
Alice	[Allie]
Carol	[Caroline]
Dan	[Daniel, Danny]

Social Network



Social Network

 Friends Table

User	Friend
Bob	Alice
Alice	Bob
Alice	Carol
Alice	Dan

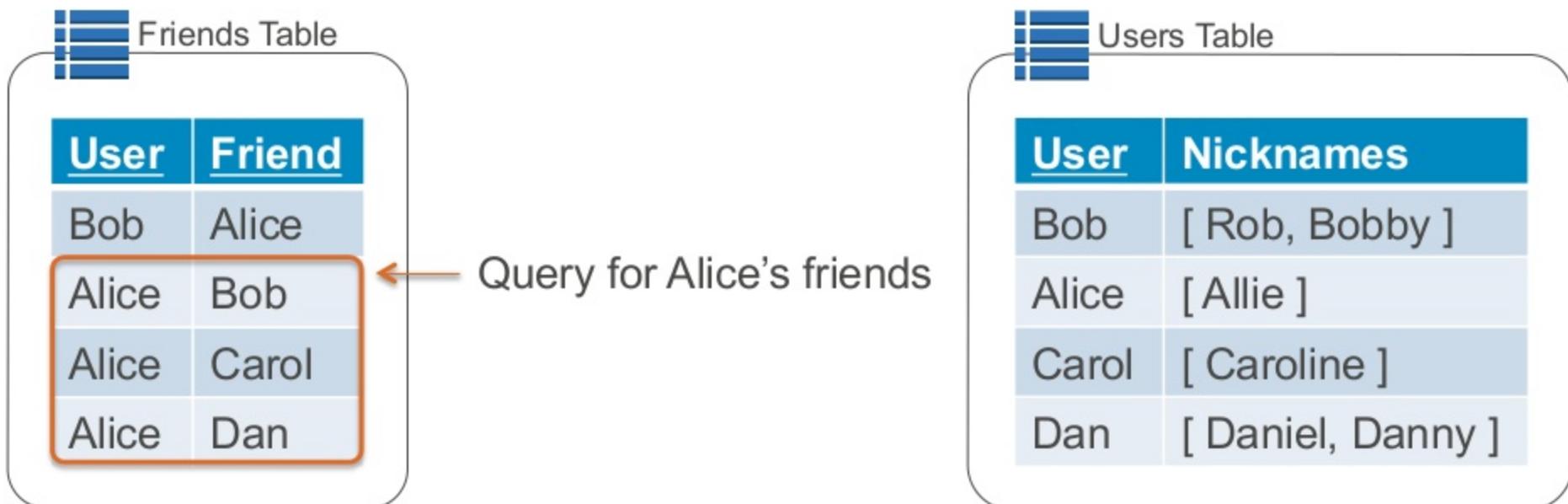
 Users Table

User	Nicknames
Bob	[Rob, Bobby]
Alice	[Allie]
Carol	[Caroline]
Dan	[Daniel, Danny]

Social Network



Social Network



Recap: Social Network

- Hash schema for key-value lookups
- Hash and Range schema for Query

Image Tagging

secondary indexes

Image Tagging

- Query a user's images
- Query a user's images by date
- Tag other users in images
- Query images a user is tagged in

Image Tagging



Images Table

Image Tagging



Images Table

User	Image	Date	Link
Bob	aed4c	2013-10-01	s3://...
Bob	cf2e2	2013-09-05	s3://...
Bob	f93bae	2013-10-08	s3://...
Alice	ca61a	2013-09-12	s3://...

Image Tagging

Hash and Range
Primary Key Schema



Images Table

A diagram showing a red arrow pointing from the 'Image' column header in the table to the 'Image' column header in the caption.

User	Image	Date	Link
Bob	aed4c	2013-10-01	s3://...
Bob	cf2e2	2013-09-05	s3://...
Bob	f93bae	2013-10-08	s3://...
Alice	ca61a	2013-09-12	s3://...

Image Tagging



Bob



Images Table

User	Image	Date	Link
Bob	aed4c	2013-10-01	s3://...
Bob	cf2e2	2013-09-05	s3://...
Bob	f93bae	2013-10-08	s3://...
Alice	ca61a	2013-09-12	s3://...

Query for Bob's Images

Image Tagging

- Query a user's images
- **Query a user's images by date**
- Tag other users in images
- Query images a user is tagged in

Local Secondary Indexes

- Alternate Range Key for your table
- More flexible Query patterns
- Local to the Hash Key

Image Tagging



Images Table

User	Image	Date	Link
Bob	aed4c	2013-10-01	s3://...
Bob	cf2e2	2013-09-05	s3://...
Bob	f93bae	2013-10-08	s3://...
Alice	ca61a	2013-09-12	s3://...

Table

Local to a Hash Key value

Image Tagging

Local Secondary Index on Date

Images Table

User	Image	Date	Link
Bob	aed4c	2013-10-01	s3://...
Bob	cf2e2	2013-09-05	s3://...
Bob	f93bae	2013-10-08	s3://...
Alice	ca61a	2013-09-12	s3://...

Table

User	Date	Image
Bob	2013-09-05	cf2e2
Bob	2013-10-01	aed4c
Bob	2013-10-08	f93bae
Alice	2013-09-12	ca61a

ByDate Local Secondary Index

Image Tagging

Query for Bob's two most recent images



Images Table

User	Image	Date	Link
Bob	aed4c	2013-10-01	s3://...
Bob	cf2e2	2013-09-05	s3://...
Bob	f93bae	2013-10-08	s3://...
Alice	ca61a	2013-09-12	s3://...

Table

User	Date	Image
Bob	2013-09-05	cf2e2
Bob	2013-10-01	aed4c
Bob	2013-10-08	f93bae
Alice	2013-09-12	ca61a

ByDate Local Secondary Index

Image Tagging

- Query a user's images
- Query a user's images by date
- **Tag other users in images**
- Query images a user is tagged in

Image Tagging



ImageTags
Table



Images
Table

Image Tagging

 ImageTags Table

Image	User
aed4c	Alice
aed4c	Bob
f93bae	Bob

Image Tagging

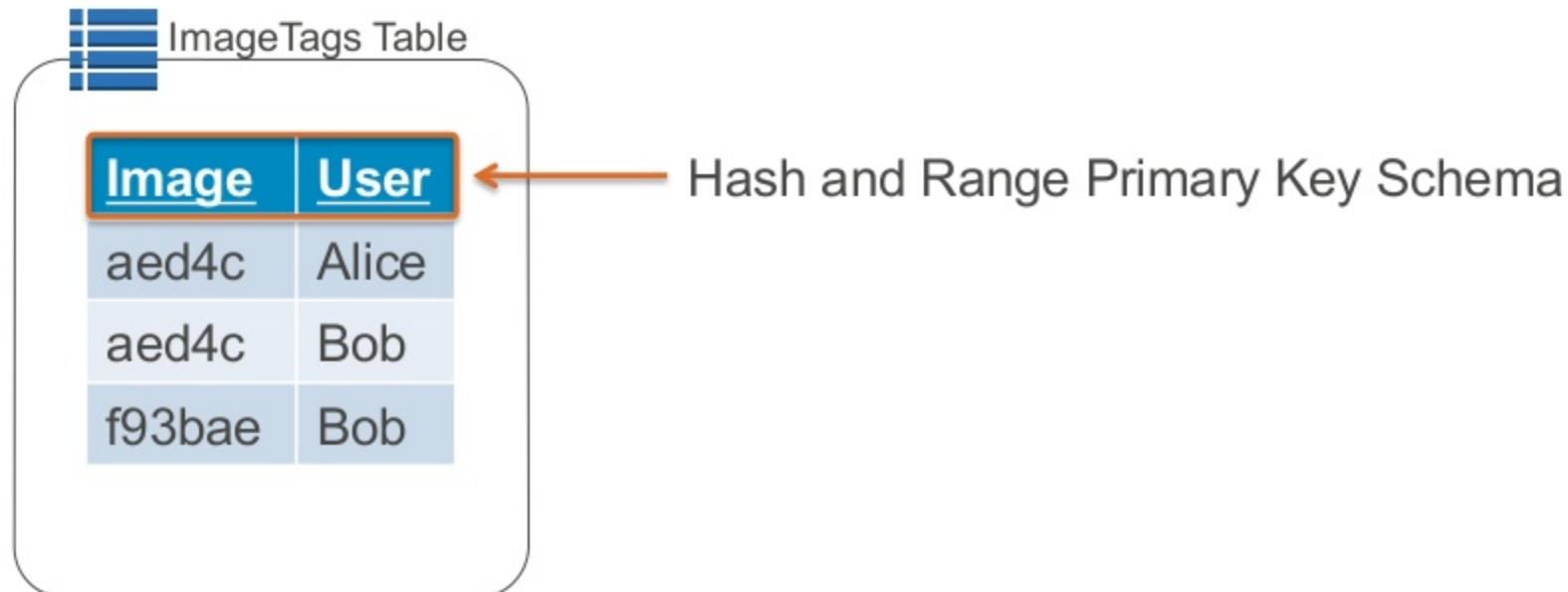


Image Tagging



Bob

ImageTags Table

Image	User
aed4c	Alice
aed4c	Bob
f93bae	Bob

Query Users
tagged in Image aed4c

Image Tagging



Bob

ImageTags Table

Image	User
aed4c	Alice
aed4c	Bob
f93bae	Alice
f93bae	Bob

Tag Alice in Image f93bae



Recap: Image Tagging

- Query a user's images
- Query a user's images by date
- Tag other users in images
- **Query images a user is tagged in**

Global Secondary Indexes

- Alternate Hash and/or Range Key for your table
- Even more flexible Query patterns

Image Tagging

 ImageTags Table

<u>Image</u>	<u>User</u>
aed4c	Alice
aed4c	Bob
f93bae	Alice
f93bae	Bob

Table

Image Tagging

Global Secondary Index on User, Image

ImageTags Table

<u>Image</u>	<u>User</u>
aed4c	Alice
aed4c	Bob
f93bae	Alice
f93bae	Bob

Table

ByUser Global Secondary Index

<u>User</u>	<u>Image</u>
Bob	aed4c
Bob	f93bae
Alice	aed4c
Alice	f93bae

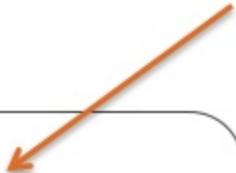


Image Tagging

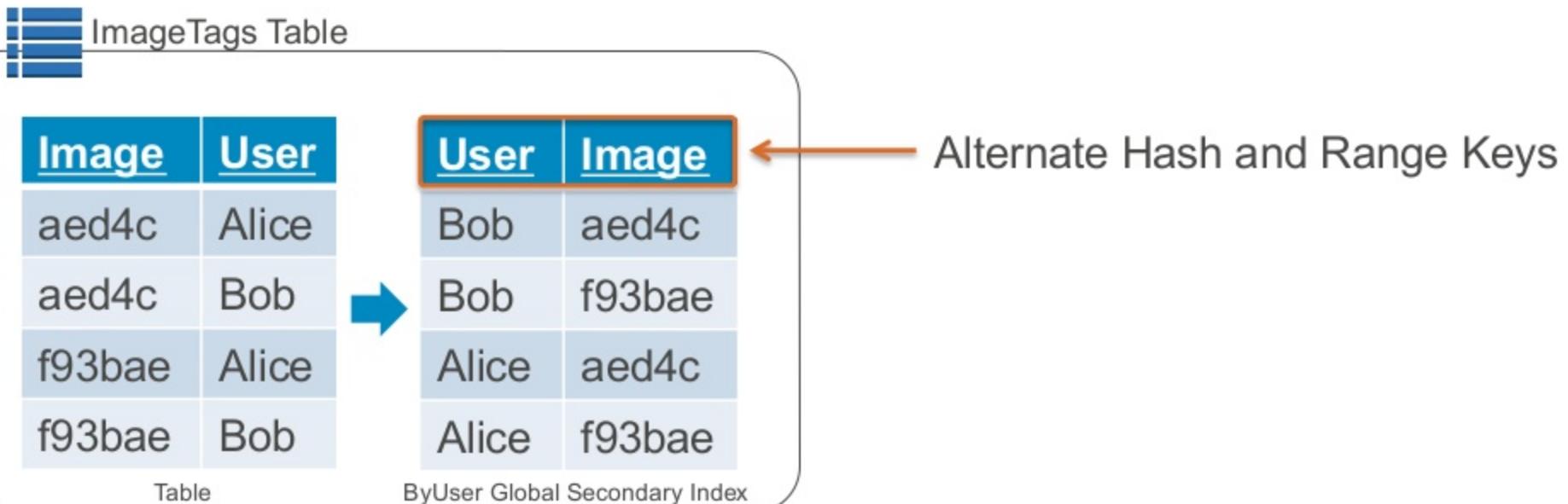


Image Tagging



Alice

Query for images tagged Alice

ImageTags Table

Image	User
aed4c	Alice
aed4c	Bob
f93bae	Alice
f93bae	Bob

User	Image
Bob	aed4c
Bob	f93bae
Alice	aed4c
Alice	f93bae

Table

ByUser Global Secondary Index



Recap: Image Tagging

- Local Secondary Indexes support flexible queries
- Global Secondary Indexes unlock even more flexible queries

EXAMPLE 3:

Storing large items

Unlimited storage

Unlimited attributes per item.

Unlimited items per table.

Maximum of 64k per item.

Split across items.

message_id = 1	part = 1	message = <first 64k>
message_id = 1	part = 2	message = <second 64k>
message_id = 1	part = 3	joined = <third 64k>

Store a pointer to S3.

message_id = 1	message = http://s3.amazonaws.com...
message_id = 2	message = http://s3.amazonaws.com...
message_id = 3	message = http://s3.amazonaws.com...

EXAMPLE 4:

Time Series Data

April

Hot and cold tables.

event_id = 1000	timestamp = 2013-04-16-09-59-01	key = value
event_id = 1001	timestamp = 2013-04-16-09-59-02	key = value
event_id = 1002	timestamp = 2013-04-16-09-59-02	key = value

March

event_id = 1000	timestamp = 2013-03-01-09-59-01	key = value
event_id = 1001	timestamp = 2013-03-01-09-59-02	key = value
event_id = 1002	timestamp = 2013-03-01-09-59-02	key = value



December

January

February

March

April

Archive data

Move old data to S3: lower cost.

Still available for analytics.

Run queries across hot and cold data
with Elastic MapReduce.

Partitioning

Uniform workload

Data stored across multiple partitions.

Data is primarily distributed by primary key.

Provisioned throughput is divided evenly across partitions.

To achieve and maintain full provisioned throughput, spread workload evenly across hash keys

Non-Uniform workloads

Might be throttled, even at high levels of throughput

BEST PRACTICE 1:

Distinct values for hash keys

Hash key elements should have a
high number of distinct values

Lots of users with unique user_id.
Workload well distributed across hash key.

user_id = mza	first_name = Matt	last_name = Wood
user_id = jeffbarr	first_name = Jeff	last_name = Barr
user_id = werner	first_name = Werner	last_name = Vogels
user_id = simone	first_name = Simone	last_name = Brunozzi
...

BEST PRACTICE 2:

Avoid limited hash key values

Hash key elements should have a
high number of distinct values.

Small number of status codes.
Unevenly, non-uniform workload.

status = 200	date = 2012-04-01-00-00-01
status = 404	date = 2012-04-01-00-00-01
status 404	date = 2012-04-01-00-00-01
status = 404	date = 2012-04-01-00-00-01

BEST PRACTICE 3:

Model for even distribution

Access by hash key value should be evenly distributed across the dataset.

Large number of devices.

Small number which are much more popular than others.

Workload unevenly distributed.

mobile_id = 100	access_date = 2012-04-01-00-00-01
mobile_id = 100	access_date = 2012-04-01-00-00-02
mobile_id = 100	access_date = 2012-04-01-00-00-03
mobile_id = 100	access_date = 2012-04-01-00-00-04
...	...

Sample access pattern.

Workload randomized by hash key.

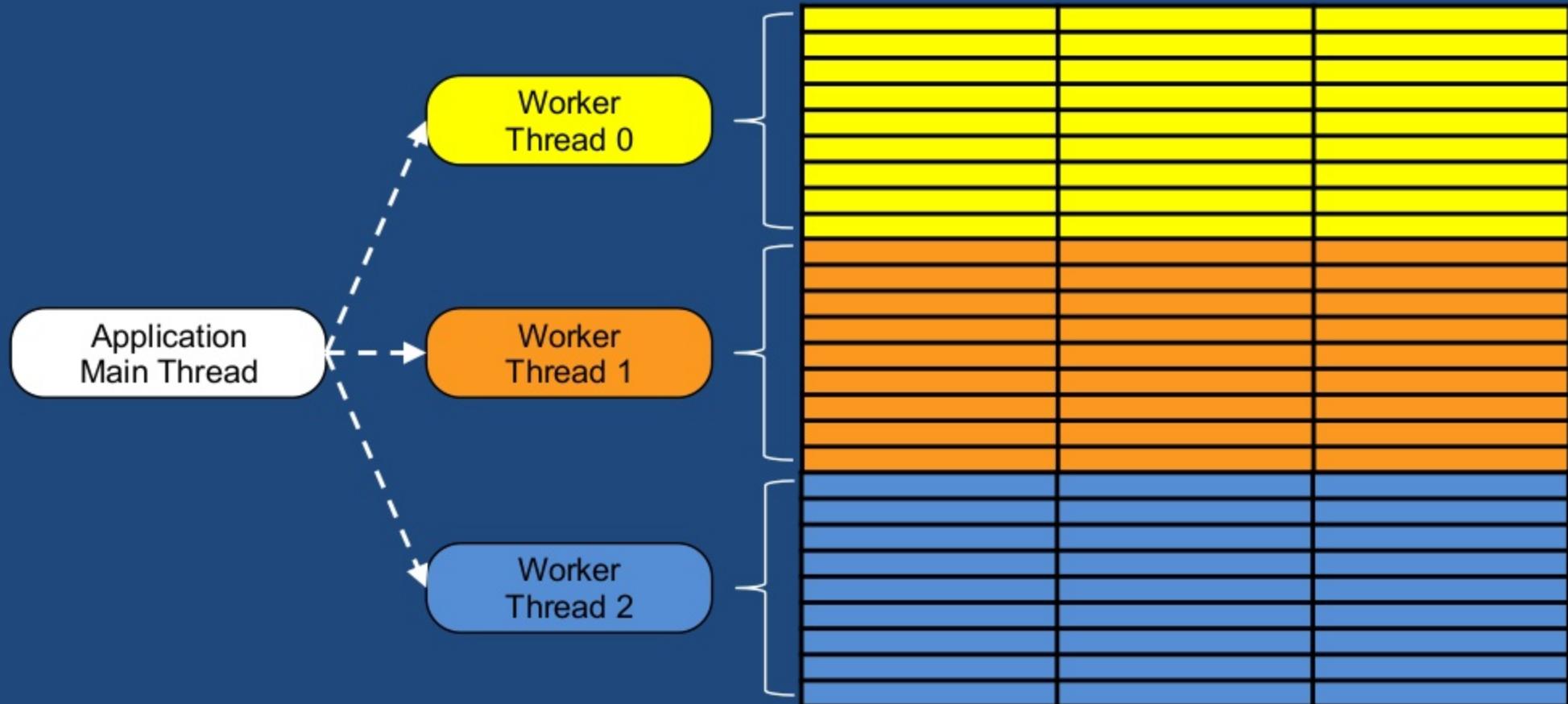
mobile_id = 100.1	access_date = 2012-04-01-00-00-01
mobile_id = 100.2	access_date = 2012-04-01-00-00-02
mobile_id = 100.3	access_date = 2012-04-01-00-00-03
mobile_id = 100.4	access_date = 2012-04-01-00-00-04
...	...

BEST PRACTICE 4:

Distribute scans across partitions

Improve retrieval times by scanning partitions concurrently using the Parallel Scan feature.

Parallel Scan: separate thread for each table segment



Working with the AWS CLI & SDKs

Working with the AWS CLI

AWS CLI for DynamoDB

Support for ad hoc operations, such as creating or deleting tables.

You can also use the CLI to embed DynamoDB operations within utility scripts

Working with the AWS CLI

```
aws dynamodb create-table  
  --table-name MusicCollection  
  --attribute-definitions  
    AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S  
  --key-schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE  
  --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1
```

This example creates a DynamoDB table named MusicCollection with two string attributes per item and these attributes set as a hash key and a range key. Provisioned throughput is set at 1 read capacity unit & 1 write capacity unit.

For readability, the command is broken into several lines.

AWS CLI actions for DynamoDB

batch-get-item	delete-table	put-item	update-table
batch-write-item	describe-table	query	
create-table	get-item	scan	
delete-item	list-tables	update-item	

As usual, you can get more details via the AWS CLI

```
$ aws dynamodb get-item help
```

Working with the AWS SDKs

Android



Browser



iOS



Java



.NET



Node.js



Python



PHP



Ruby



Simplify using Amazon DynamoDB in your applications with an API tailored to your programming language or platform
Comprehensive documentation for all these SDKs is available here: aws.amazon.com/dynamodb/developer-resources/

Amazon DynamoDB Object Mapper

Map your client-side classes to Amazon DynamoDB tables without having to write the code to transform objects into tables and vice versa



Working with DynamoDB Local

A small client-side database and server that mimics the DynamoDB service.

Enables you to write applications that use the DynamoDB API, without actually manipulating any tables or data in DynamoDB.

All API actions are rerouted to DynamoDB Local. When your application creates a table or modifies data, those changes are written to a local database. This lets you save on provisioned throughput, data storage, and data transfer fees whilst developing your applications.

3rd Party Tools, Libraries, Mappers & Mocks for DynamoDB

Autoscaling: Dynamic DynamoDB ([AWS Blog Post](#))

Testing: 3rd party emulators/mocks for development and testing

Backup/Archiving: DynamoDB Table Archiver

3rd Party Language Libraries: including – Coldfusion, Django, Erlang, Go, Groovy/Grails, Java, JavaScript, .NET, Node.js, Perl, Python & Ruby

Reporting & Analysis

Seamless scale

Scalable methods for data processing.
Scalable methods for backup/restore.

Amazon Elastic MapReduce

Managed Hadoop service for data-intensive workflows

aws.amazon.com/emr

Integrating data from DynamoDB with Amazon EMR + Hive

```
create external table items_db
(id string, votes bigint, views bigint) stored by
'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
tblproperties
("dynamodb.table.name" = "items",
"dynamodb.column.mapping" =
"id:id,votes:votes,views:views");
```

This is the same process as mapping an external table from data stored in HDFS, but it uses the AWS provided storage handler named dynamodb.DynamoDBStorageHandler

Integrating data from DynamoDB with Amazon EMR + Hive

```
select id, likes, views  
from items_db  
order by views desc;
```

This is the same process as mapping an external table from data stored in HDFS, but it uses the AWS provided storage handler named dynamodb.DynamoDBStorageHandler

Some other things...

Who is using DynamoDB?

“

We spend more on snacks
than we do on Amazon
DynamoDB.

Valentino Volonghi
CTO, Adroll



”

AdRoll, an online advertising platform, serves 50 billion impressions a day worldwide with its global retargeting platforms.

Adroll Uses AWS to Grow by More Than 15,000% in a Year

- Needed high-performance, flexible platform to swiftly sync data for worldwide audience
 - Processes 50 TB of data a day
 - Serves 50 billion impressions a day
 - Stores 1.5 PB of data
- Worldwide deployment minimizes latency

Find out more about this and other AWS case studies here: aws.amazon.com/solutions/case-studies/

Resources to learn more

Getting Started with Amazon DynamoDB (AWS Documentation Center):

docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStartedDynamoDB.html

Auto Scale DynamoDB with Dynamic DynamoDB (Jeff Barr's Blog):

aws.amazon.com/blogs/aws/auto-scale-dynamodb-with-dynamic-dynamodb/

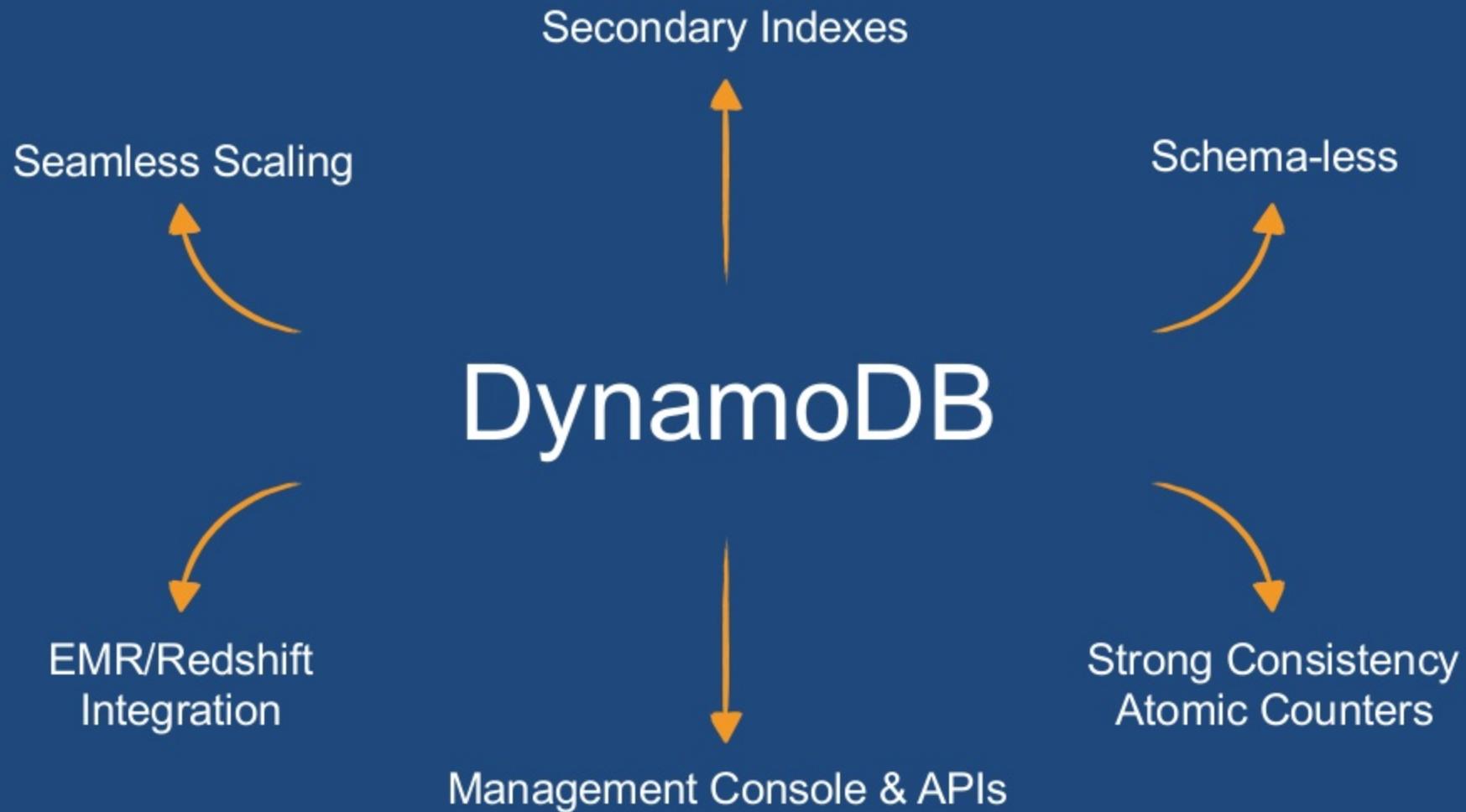
SmugMug: From MySQL to Amazon DynamoDB (DAT204) from AWS re:Invent 2013 (video)

<https://www.youtube.com/watch?v=g1Yk0E6sslq>

Amazon DynamoDB Design Patterns for Ultra-High Performance Apps (DAT304) from AWS re:Invent 2013 (video)

<https://www.youtube.com/watch?v=Dh8kp1AcRg0>

Summary



Find out more:

aws.amazon.com/dynamodb

AWS Training & Certification

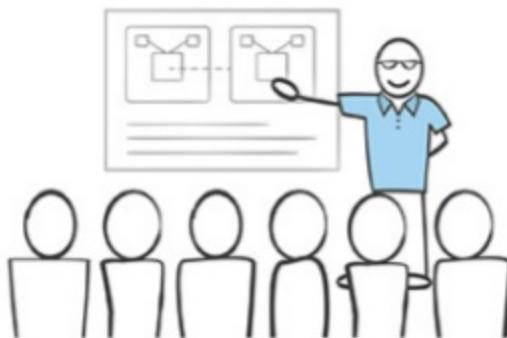
Self-Paced Labs



Try products, gain new skills, and get hands-on practice working with AWS technologies

[aws.amazon.com/training/
self-paced-labs](https://aws.amazon.com/training/self-paced-labs)

Training



Skill up and gain confidence to design, develop, deploy and manage your applications on AWS

aws.amazon.com/training

Certification



Demonstrate your skills, knowledge, and expertise with the AWS platform

aws.amazon.com/certification

Follow us for more
events & webinars



Ian Massingham – Technical Evangelist

 @IanMmmm



@AWS_UKI for local AWS events & news



@AWScloud for Global AWS News and Announcements

We typically see customers start by trying our services

All Products

Compute & Networking

Storage

Database

Application Services

Development & Management

AWS Marketplace Software

FAQ »

Find answers to common questions about the AWS Free Tier.



Amazon EC2 »

Web service that provides resizable compute capacity in the cloud.



Amazon S3 »

Highly-scalable, reliable, and low-latency data storage.



Amazon RDS »

Managed MySQL, Oracle and SQL Server databases.



Amazon CloudWatch »

Monitoring for AWS cloud resources and applications.



AWS Data Pipeline »

Orchestration for data-driven workflows.



Amazon DynamoDB »

Fully managed NoSQL database service with seamless scalability.



Amazon EBS »

Highly available, highly reliable, predictable storage volumes.



Amazon ELB »

Web service that provides scalability and high availability.



Amazon ElastiCache »

Managed scale-out caching.



Amazon SNS »

Web service to set up, operate, and send notifications from the cloud.



Amazon Elastic Transcoder »

Convert your media files easily, at low cost and at scale.



Amazon SWF »

Workflow service for building scalable, resilient applications.



AWS Marketplace »

Partner software pre-configured to run on AWS.



Amazon SQS »

Scalable queue for storing messages as they travel between computers.

Get started now at : aws.amazon.com/getting-started



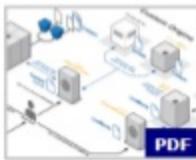
Design your application for the AWS Cloud

AWS Reference Architectures

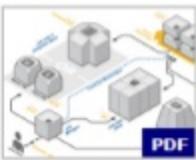
The flexibility of AWS allows you to design your application architectures the way you like. AWS Reference Architecture Datasheets provide you with the architectural guidance you need in order to build an application that takes full advantage of the AWS cloud. Each datasheet includes a visual representation of the architecture and basic description of how each service is used.



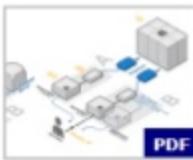
Web Application Hosting
Build highly-scalable and reliable web or mobile-web applications ([PDF](#))



Content and Media Serving
Build highly reliable systems that serve massive amounts of content and media ([PDF](#))



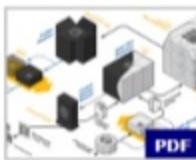
Batch Processing
Build auto-scalable batch processing systems like video processing pipelines ([PDF](#))



Fault tolerance and High Availability
Build systems that quickly failover to new instances in an event of failure ([PDF](#))



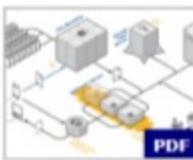
Large Scale Processing and Huge Data sets
Build high-performance computing systems that involve Big Data ([PDF](#))



Ad Serving
Build highly-scalable online ad serving solutions ([PDF](#))



Disaster Recovery for Local Applications
Build cost-effective Disaster Recovery solutions for on-premises applications ([PDF](#))



File Synchronization
Build simple file synchronization service ([PDF](#))

More details on the AWS Architecture Center at : aws.amazon.com/architecture

