Demystifying cloud-based data stores with Linux and Java

SESSION INTRO AND WHAT TO EXPECT

Harold Wong
Dev Experience – Azure Cloud

Bruno Terkaly
Dev Experience – Azure Cloud

Session Intro

Session Flow

- A complete introduction to what Azure offers
- A specific emphasis on Linux-based workloads
- A deep introduction into automated provisioning of cloud resources in Azure
- Deploying Java Apps to the cloud
- Implementing Model-View-Controlller Architectures in a web site with Azure Storage
- Working with Eclipse, Git, FileZilla, Maven, Azure SDKs

Coverage of RDBMS systems

- SQL Server
- Azure SQL DB
- MySQL
- PostGres

Coverage of NoSQL Data Stores

- Apache Cassadra
- MongoDB
- Redis
- DocumentDB

Session Intro

Activity	Start	Stop	Length
Session Definition	1:30 PM	1:40 PM	10
What is Azure, Getting Ready, Installation	1:40 PM	1:50 PM	10
MySQL - Discuss	1:50 PM	2:02 PM	12
MySQL - Hands-On	2:02 PM	2:37 PM	35
PostGres - Discuss	2:37 PM	2:49 PM	12
PostGres - Hands-On	2:49 PM	3:24 PM	35
MongoDB - Discuss	3:24 PM	3:36 PM	12
MongoDB - Hands-On	3:36 PM	4:11 PM	35
Cassandra - Discuss & Wrap-up	4:11 PM	4:23 PM	12
Cassandra - Hands-On	4:23 PM	5:00 PM	37

Session Intro

Getting An Azure Subscription And Configuring Your Environment	**01m 09s**
Provisioning An Ubuntu Linux VM At The Azure Portal	08m 37s
Installing The Azure Cross Platform Tooling	03m 29s
Provisioning Resources With The Azure Resource Manager - Part 1	06m 52s
Provisioning Resources With The Azure Resource Manager - Part 2	04m 57s
Provisioning Resources With The Azure Resource Manager - Part 3	14m 04s
Using The Azure Cross Platform CLI To Provision Infrastructure	07m 39s
Downloading And Installing Java, Eclipse, FileZilla And Git	11m 31s
Hello World With A Java Application And A Java Dynamic Web Application	12m 41s
Azure Storage - Introduction And Azure Blobs Overview - The Storage Landscape In Azure	06m 32s
Provisioning A Storage Account With The Portal And ARM	14m 53s

And much more online

What do you want to cover?

Rank May 2016	Rank Apr 2016	Rank May 2015	DBMS	Database Model	Score May 2016	Score Apr 2016	Score May 2015
1.	1.	1.	Oracle	Relational DBMS	1462.02	-5.51	+19.93
2.	2.	2.	MySQL	Relational DBMS	1371.83	+1.72	+77.56
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1142.82	+7.77	+11.79
4.	4.	4.	MongoDB	Document store	320.22	+7.78	+42.90
5.	5.	5.	PostgreSQL	Relational DBMS	307.61	+3.89	+34.09
6.	6.	6.	DB2	Relational DBMS	185.96	+1.87	-15.09
7.	↑ 8.	↑ 8.	Cassandra	Wide column store	134.50	+4.83	+27.95
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS	131.58	-0.39	-14.00
9.	9.	↑ 10.	Redis	Key-value store	108.24	-3.00	+13.51
10.	10.	↓ 9.	SQLite	Relational DBMS	107.26	-0.70	+2.10

Tools that we will download and install

Java	• An SDK and Runtime to support Java Programming
Eclipse	• Developer tooling we will use to write, debug, and deploy Java apps
FileZilla	• FTP client supported by Windows, MacOS, and Linux. • Used for deployment.
Git	• Used for source code control and for deployment
Azure SDKs	• Various SDKs will be used for specific tasks in Azure
Jersey – Java API	• Java libraries useful for REST-based programming
Tomcat	• Apache web server used to run our web applications. Also hosted in Azure.

Compiling Maven Code

3 main commands with Eclipse to compile project:

- Right mouse click on the project and select:
  - Maven / Update Project
  - Run As / Maven Clean
  - Run As / Maven Install

The 3 crucial commands in Eclipse when working with Maven

Getting to the downloads

HowToCreateAVM.mp4

http://bit.ly/azuresessiondownloads

http://bit.ly/osconcode

Download Notes

You will need two versions of Java.

- The latest 8.0 version for running Eclipse.
- You will need 1.7.0_51 for our Java Web Apps because that is the version hosted in Tomcat.

64-bit

You will need to make sure you download the 64-bit versions of everything if you are running a 64-bit Operating system.

Java

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html

http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jre-7u51-oth-JPR

Eclipse

http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/marsr

FileZilla

https://filezilla-project.org/download.php

Git

http://git-scm.com/download

Azure SDKs

Download as needed throughout the course

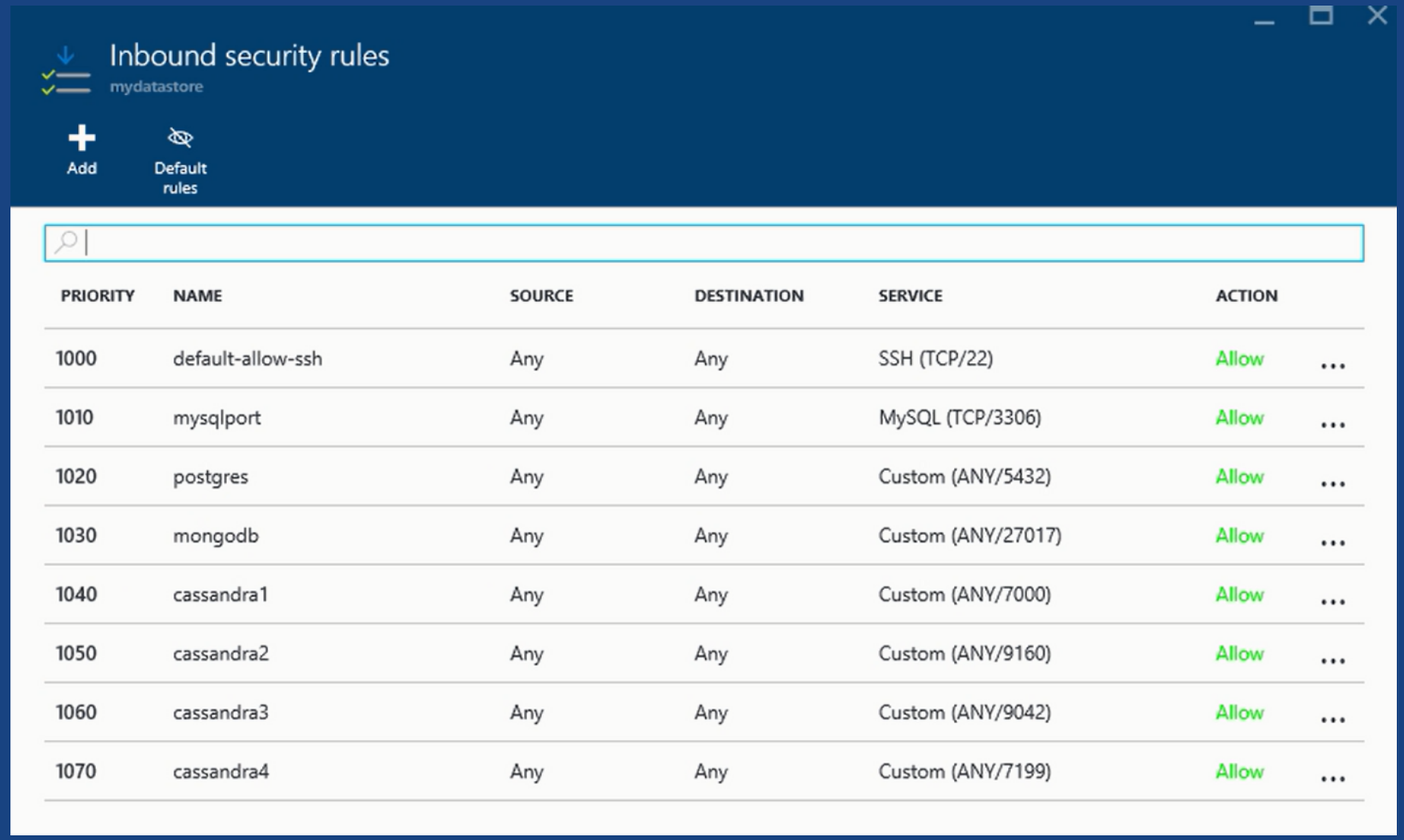
Jersey – Java API

https://jax-rs-spec.java.net/

Tomcat

http://tomcat.apache.org/download-70.cgi

Opening up ports on your Azure VM



PRIORITY	NAME	SOURCE	DESTINATION	SERVICE	ACTION
1000	default-allow-ssh	Any	Any	SSH (TCP/22)	Allow
1010	mysqlport	Any	Any	MySQL (TCP/3306)	Allow
1020	postgres	Any	Any	Custom (ANY/5432)	Allow
1030	mongodb	Any	Any	Custom (ANY/27017)	Allow
1040	cassandra1	Any	Any	Custom (ANY/7000)	Allow
1050	cassandra2	Any	Any	Custom (ANY/9160)	Allow
1060	cassandra3	Any	Any	Custom (ANY/9042)	Allow
1070	cassandra4	Any	Any	Custom (ANY/7199)	Allow

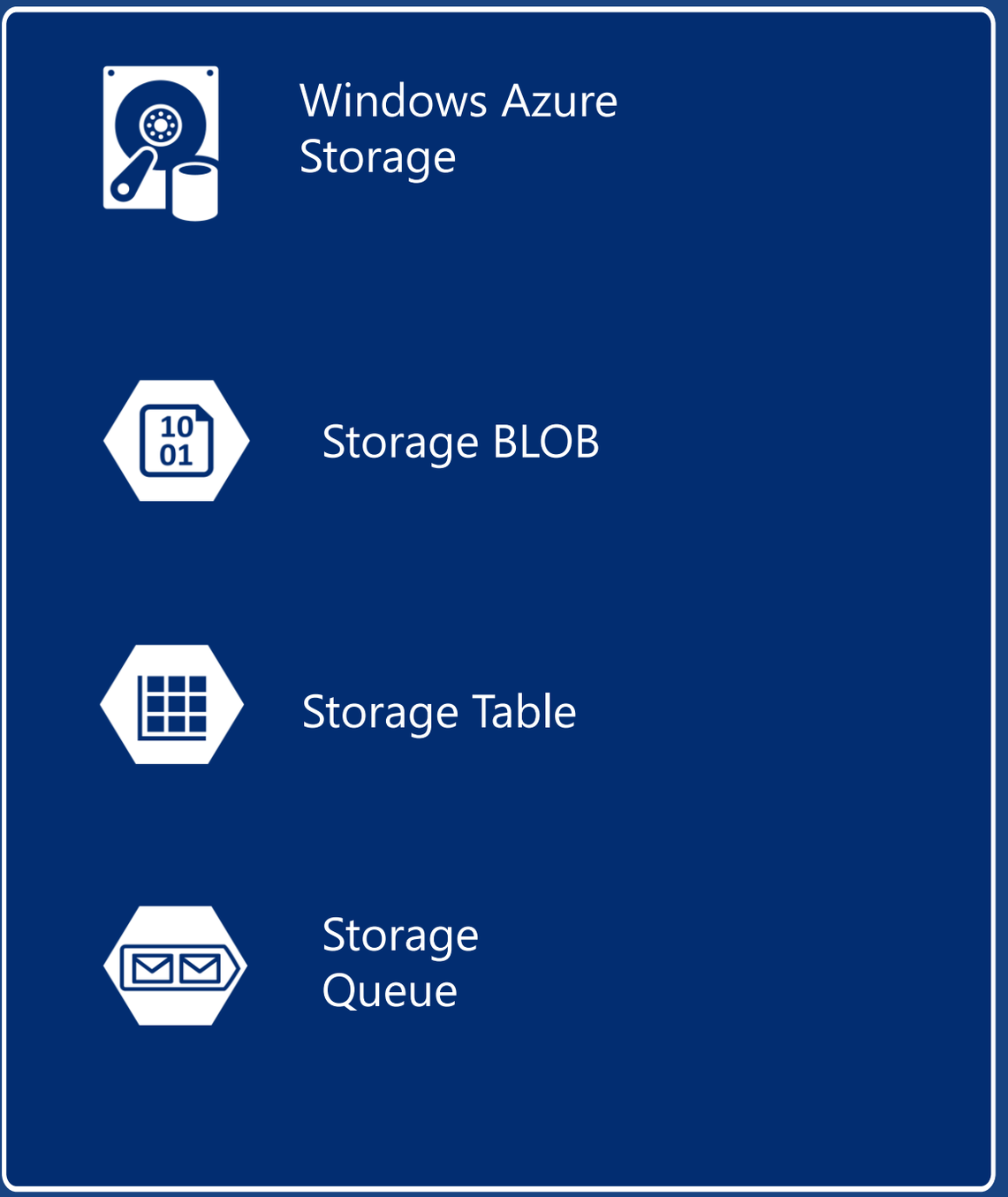
Session Intro

What this session is focused on

- This session is focused on the world's most popular data stores using the most popular programming language

This session provides high level plus running Java code

- This session is all about concrete examples, both from the perspective of installation and configuration, as well as creating code that directly manipulates data



Session Intro

Session Philosophy

- No copy and paste
- Everything starts with File / New Project
- Code is typed in one character at a time
- Fastest, most productive way to learn
- Full Source code provided

Learning Strategy

- Students follow along from beginning to end
- Encapsulates all 3 learning techniques
  - Hearing
  - Watching
  - Doing

Session Intro

Session Philosophy

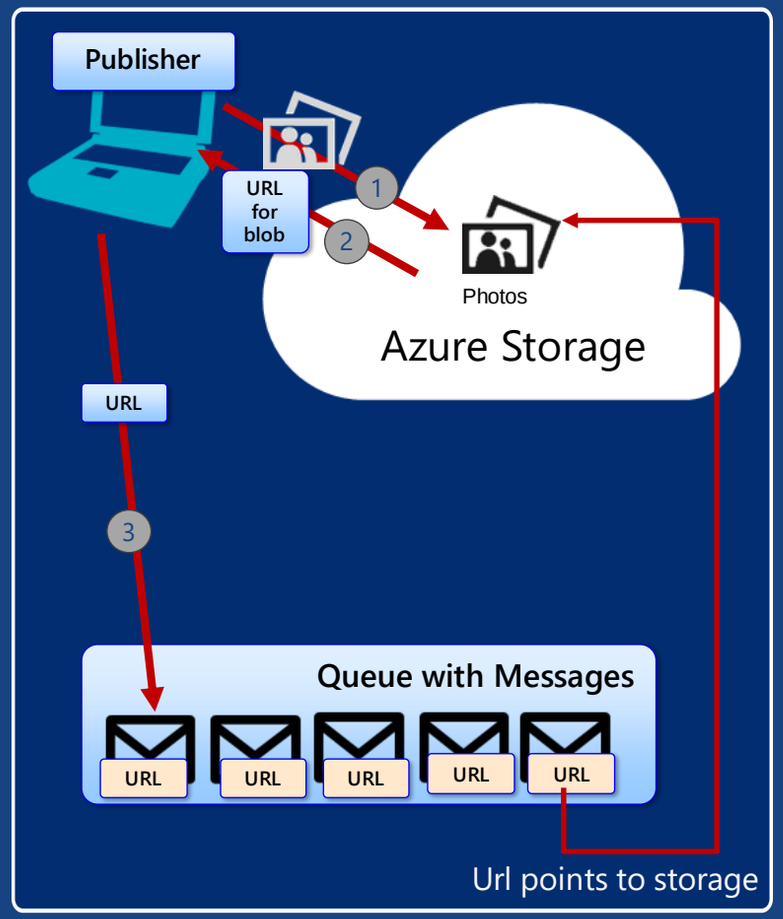
- No copy and paste
- Everything starts with File / New Project
- Code is typed in one character at a time
- Fastest, most productive way to learn
- Full Source code provided

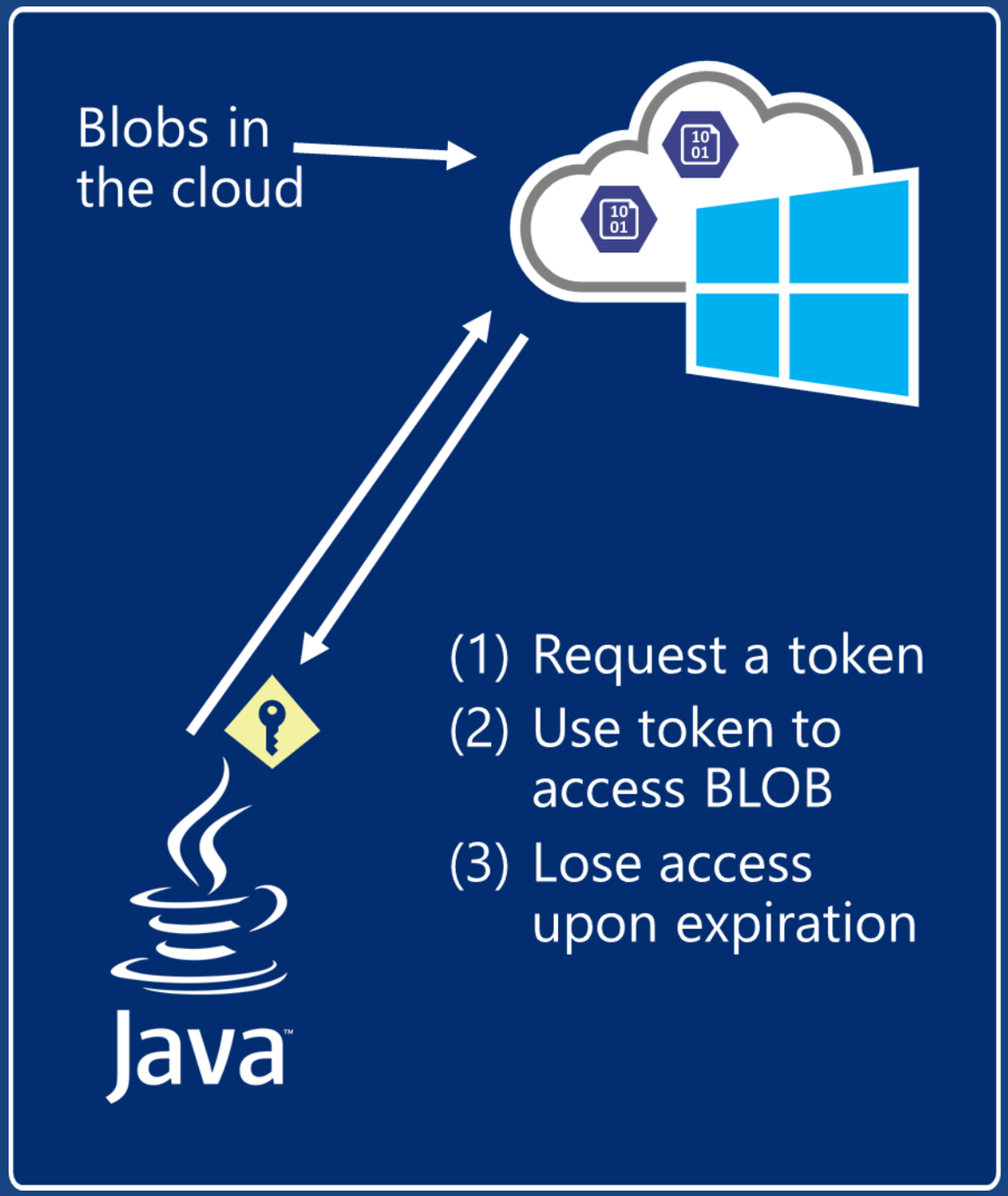
Learning Strategy

- Students follow along from beginning to end
- Encapsulates all 3 learning techniques
  - Hearing
  - Watching
  - Doing

Session Intro

For visual learners, numerous concise, clear diagrams are included





Session Intro

This session is hands-on

- Concrete examples for both installation and coding
- Demos on live, running Linux VMs
- All commands reviewed and explained
- Step by step Java samples using Eclipse, including debugging

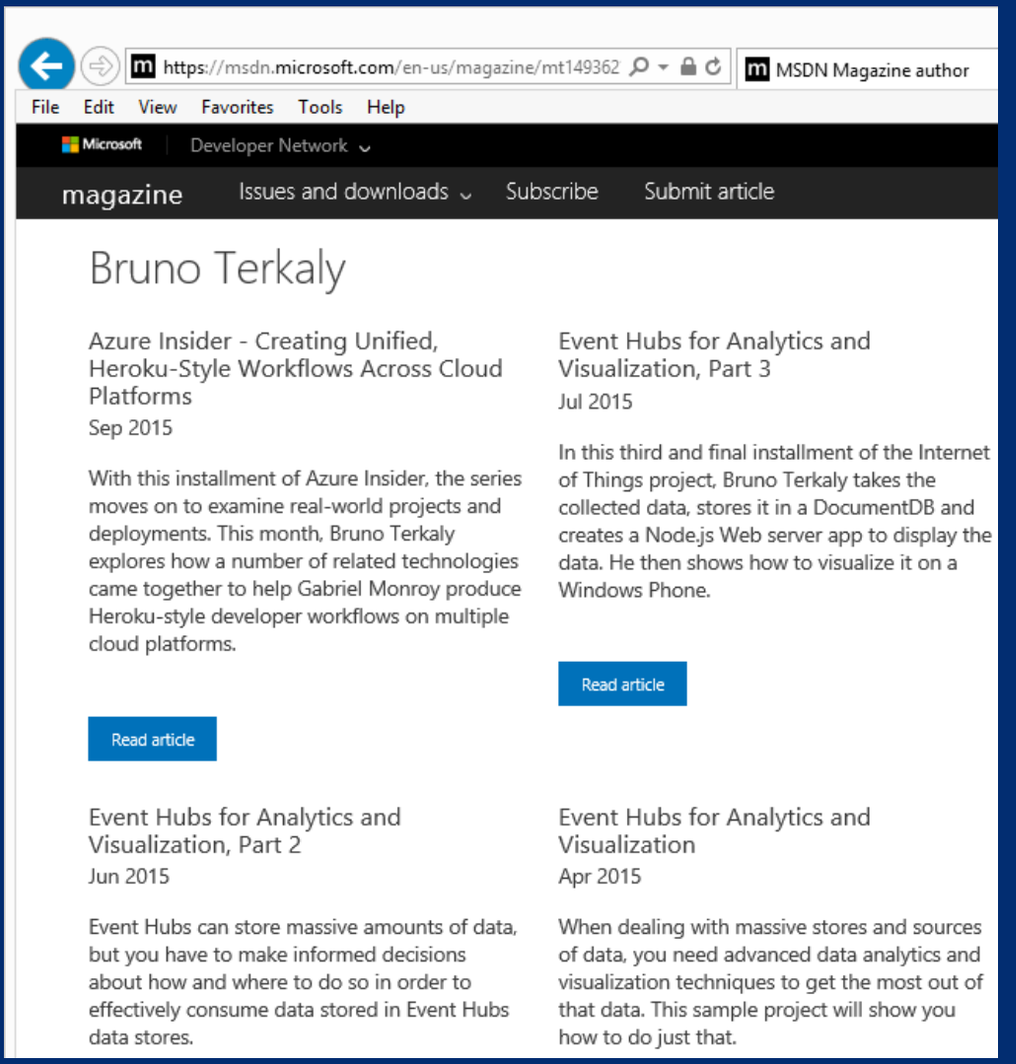
Cloud enabled examples

Hands-on construction

- Great for anybody getting started and cloud computing should consider this session
- The focus is to be practical and for students to start writing meaningful code immediately
- Great for anybody looking to land a new job in the cloud

Bruno Terkaly

- Principal Software Engineer at Microsoft with a 10 Year tenure
- A long career in education, writing, speech giving
- Popular and content-heavy blog at http://blogs.msdn.com/b/brunoterkaly/
- Developer and publisher of multiple phone apps in the Windows Store
- Finalist in InfoWorld's best 12 youtube.com instructional videos in 2013
  - http://www.infoworld.com/article/2606686/it-training/114051-From-IFTTT-to-Node.js-A-bakers-dozon-of-tech-tutorials.html



Mobile, Node.js, Big Data, Multi-Tenant Architectures, Azure, Windows Phone, Open Source, C/C++, C#, Node.js/Javascript, SQL, LINQ, and more

Bruno Terkaly

- Co-organizer of a 6,000+ member strong Bay Area Software Engineers on meetup.com
- Create huge live events and speaking to 1,000's of developers in the Silicon Valley Area
- Topics include Robotics, Machine Learning, Data Science, Node.js, Cloud Computing and much more (see http://www.meetup.com/software)
- Assist and directly engage with some of the largest software brands in the world to help them create products that run on Azure
  - Docker
  - Github
  - Mesosphere
  - DataStax
  - CoreOS
  - Deis

Top scoring presenter at the Microsoft Executive Briefing Center in Redmond, and Silicon Valley Code Camp

Work closely with Microsoft/Redmond-based engineering teams to accelerate customer adoption of Azure

Influenced roadmap for future versions of Azure

Organize and participated in weeklong engineering migration efforts for high-value customers.

AZURE FEATURE OVERVIEW – PART 1

BRUNO TERKALY – PRINCIPAL SOFTWARE ENGINEER – MICROSOFT - AZURE

Introduction to Azure

- A public cloud computing platform
- For building, deploying, and managing applications and services
- Supports extensive amount of languages, frameworks, tools

Public Cloud

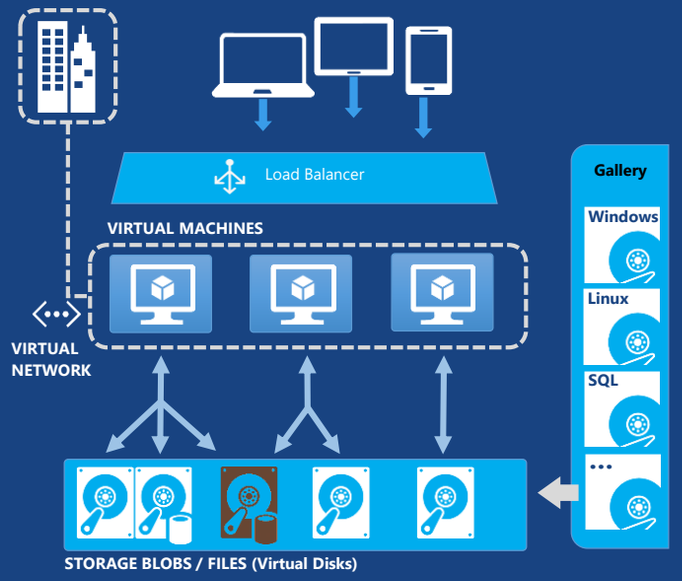
Global Set of Data Centers

Provides an extensive compute infrastructure

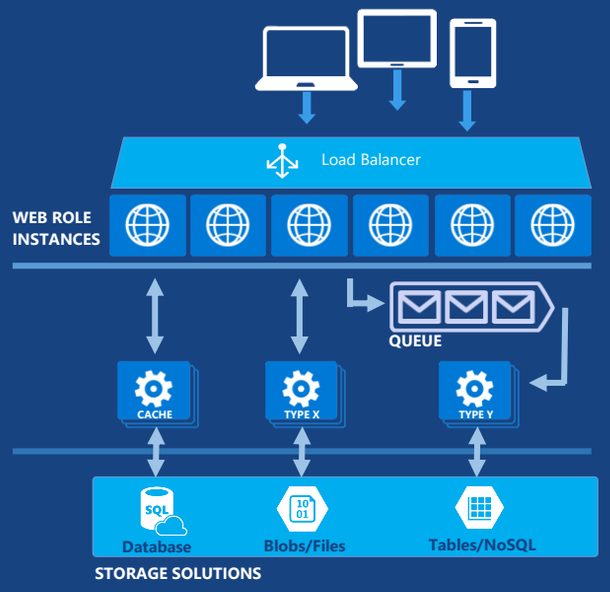
Web Sites/Services

Data

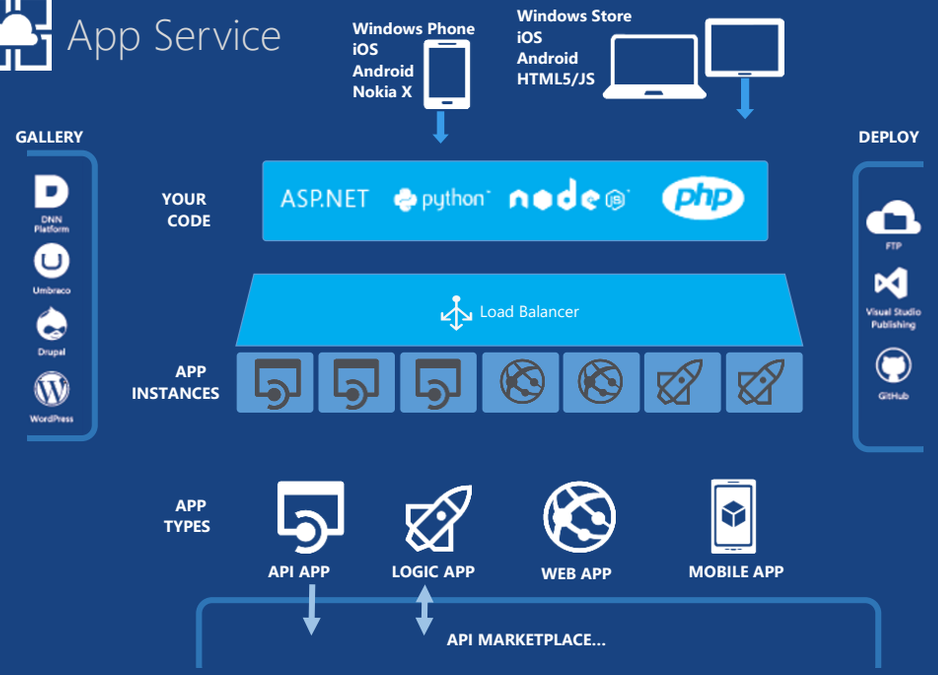
Virtual Machines



Cloud Services



App Service



COMPUTE

Virtual Machines
Get full control over a server in the cloud and maintain it as your business requires.

Cloud Services
Managed Virtual Machines with specific web and worker roles that are stateless

Batch
For running large scale parallel and high performance computing (HPC) applications

Scheduler
Create jobs that run reliably on simple or complex schedules to invoke any type of service.

Remote App
Access Windows apps that run within the Service on VM's from any device and any location.

NETWORKING

Virtual Network
Provision and manage VPNs in Azure and securely link to your on-premises IT infrastructure.

Express Route
Connect on-premises and cloud data centers directly through dedicated, non-internet lines.

Traffic Manager
Load-balance incoming global traffic across multiple services running in multiple data centers.

IDENTITY & ACCESS

Active Directory
Identity and access management for cloud applications and ability to link to on-premises Server AD.

Multi-Factor Authentication
Safeguard access to data and apps with additional physical layer of security control.

MEDIA & CDN

Media Services
Range of services that support video on-demand and live streaming workflows.

Content Delivery Network (CDN)
Cache content for your apps at 100's of edge locations to improve user experiences.

WEB & MOBILE

Web Apps
Managed web platform, get started for free and scale as you go using many tools/ languages.

Mobile Apps
Add backend capabilities to mobile apps, with native client support on most device platforms.

API Apps
Create and surface your app logic as APIs for other services and apps to consume.

Logic Apps
Build/execute business processes by linking your own custom API's with an API Gallery/Marketplace

API Management
Publish and Manage APIs to developers, partners and employees securely and at scale.

Notification Hubs
Deliver millions of cross platform push notifications from any application backend, anywhere.

ANALYTICS

HDInsight
Big Data (based on Apache Hadoop) analytics that integrate easily with Microsoft Office.

Machine Learning
Mine historical data with compute power to predict future trends or behavior.

Stream Analytics
Process data streams in real-time to discover and react to trends.

Data Factory
Ingest data from multiple sources to combine into a cloud based Data Warehouse.

Event Hubs
Ingest, persist, process millions of events per second from millions of devices.

Mobile Engagement
Ingest, persist, process millions of events per second from millions of devices.

STORAGE & BACKUP

Storage Blobs & Files
Store binary application data and web content – store for dedicated and shared virtual disks for VM's

Backup
Managed service that handles backup/restore of Windows Server machines/backup agent.

Import/Export
For massive data transfer – ship encrypted disks to move data in/out of blob storage.

Site Recovery
Coordinate replication and recovery of System Center private clouds

StorSimple
Automated, policy driven solution to extend on-premises primary storage for backup / DR.

DATA

SQL Database
Managed relational database service with high availability and selectable performance levels.

DocumentDB
Store/retrieve millions of JSON objects from a highly scalable NoSQL document database.

Redis Cache
Make applications scale and be more responsive under load by keeping data closer to app logic.

Search
Managed, scalable search service for your apps, create tunable search results and ranking models.

Tables
Massive scale for semi-structured key/value type data in this schema-less NoSQL store.

DEVELOPER SERVICES

Visual Studio Online
Store code, plan and track projects, build, deploy and test apps in the cloud collaboratively.

Application Insights
Analyze app usage, availability and performance to detect issues and solve problems proactively.

HYBRID INTEGRATION

Storage Queues
Simple message queue for application de-coupling architecture for scale out.

Biztalk Services
Build EDI and Enterprise App Integration (EAI) solutions in the cloud.

Hybrid Connections
Connect apps in Azure with on-premises resources without a VPN or dedicated line.

Service Bus
Messaging capabilities (pub/sub, queues) and on-premises to cloud connectivity solution.

MANAGEMENT

Automation
Run durable PowerShell scripts to automate frequent, long running, complex Azure tasks.

Portal
Web based experience to provision, control and monitor all Azure services.

Key Vault
Safeguard and control keys and secrets in cloud scale hardware security modules.

Operational Insights
Analyze and troubleshoot on-premises IT infrastructure without using instrumented code.

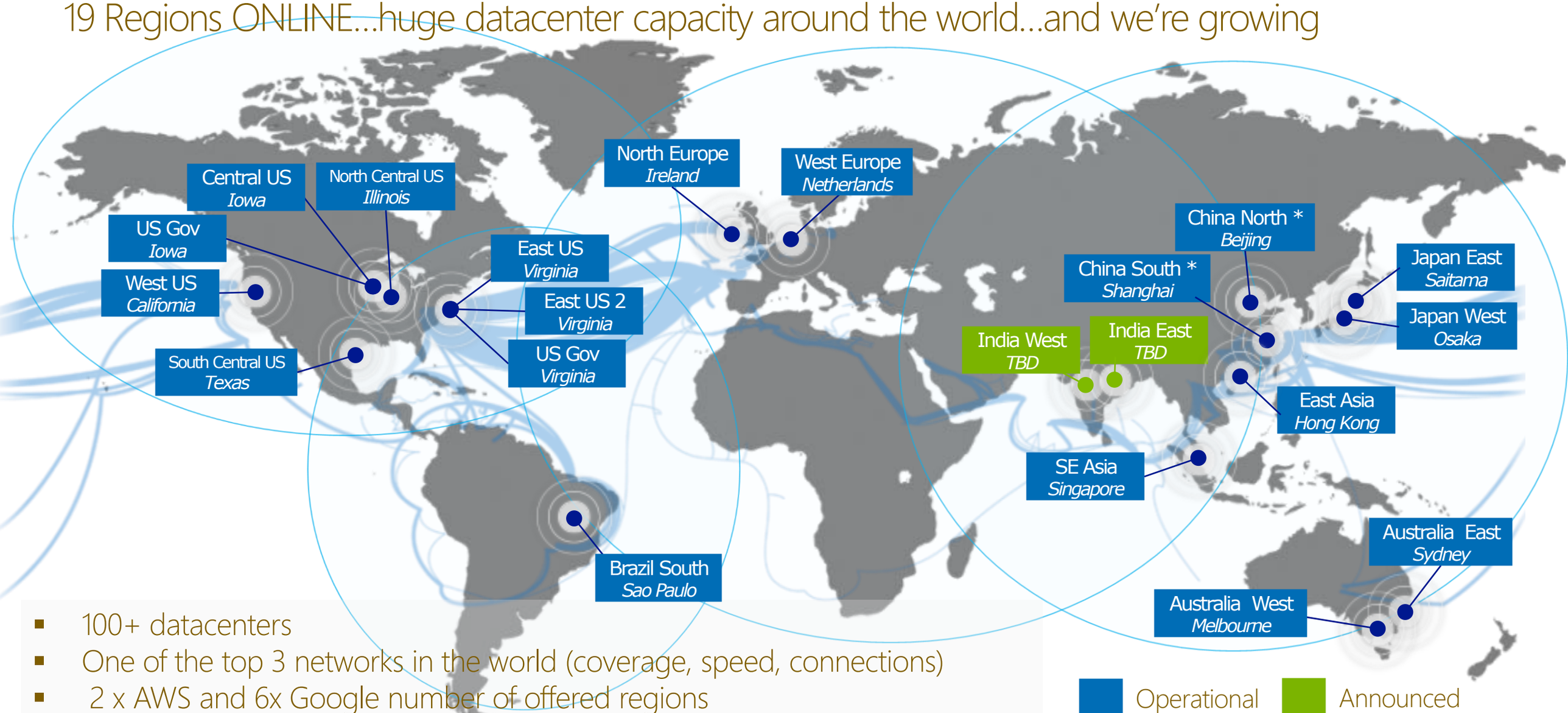
COMMERCE

Store / Marketplace
Find and manage other services provided by third parties.

VM Depot
Find free open source VM images that you can download and run in Azure Virtual Machines.

Huge infrastructure scale is the enabler

19 Regions ONLINE…huge datacenter capacity around the world…and we're growing



- 100+ datacenters
- One of the top 3 networks in the world (coverage, speed, connections)
- 2 x AWS and 6x Google number of offered regions
- G Series – Largest VM available in the market – 32 cores, 448GB Ram, SSD…

Operational Announced

* Operated by 21Vianet

Analytics





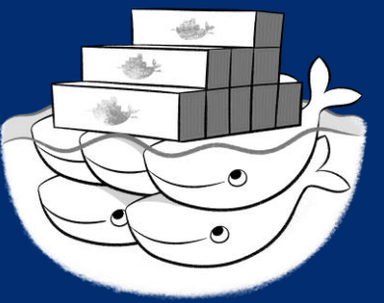

HDInsight	Big Data (based on Apache Hadoop) analytics that integrate easily with Microsoft Office.
Machine Learning	Mine historical data with compute power to predict future trends or behavior.
Stream Analytics	Process data streams in real-time to discover and react to trends.
Data Factory	Ingest data from multiple sources to combine into a cloud based Data Warehouse.
Event Hubs	Ingest, persist, process millions of events per second from millions of devices.
Mobile Engagement	About understanding users and their behavior with mobile devices. Provides analytics on how users interact with mobile applications.

Linux Focus

- Cross-Plat CLI scripting, not PowerShell
- Java, not C#
- MySQL and SQL Server
- Java SDKs
- Hadoop, Pig, Hive
- Services via REST

Open Standards

Hadoop
Pig
Hive



Docker

Getting Ready To Write Code

- We will:
  - Download software
  - Install Software
  - Configure Dev Environment

This is a demo heavy module

1. Java
2. Eclipse
3. FileZilla
4. Git
5. Azure SDKs
6. Tomcat
7. Jersey – Java API

Tools that we will download and install

Java	• An SDK and Runtime to support Java Programming
Eclipse	• Developer tooling we will use to write, debug, and deploy Java apps
FileZilla	• FTP client supported by Windows, MacOS, and Linux. • Used for deployment.
Git	• Used for source code control and for deployment
Azure SDKs	• Various SDKs will be used for specific tasks in Azure
Jersey – Java API	• Java libraries useful for REST-based programming
Tomcat	• Apache web server used to run our web applications. Also hosted in Azure.

The SQL Spectrum

- Control versus Cost
- Pay-as-you-go
- Upfront investment
- Bring your own license
- Multi-tenancy
- Scaling, Backups, Upgrades, DR
- Time to market
- SLA (VMs = 99.95%, Azure SQL DB = 99.99%)
- Compatibility

Physical	Raw Iron
Virtual	SQL Server Private Cloud with Virtual Machines
IaaS	SQL Server in an Azure VM
PaaS	Azure SQL Database

Overview of Relational Storage

Provisioning Options

- Oracle
- SQL Server
- MySQL
- Azure SQL Database

Oracle 12.1.0.1 Enterprise

Oracle Linux 6.4

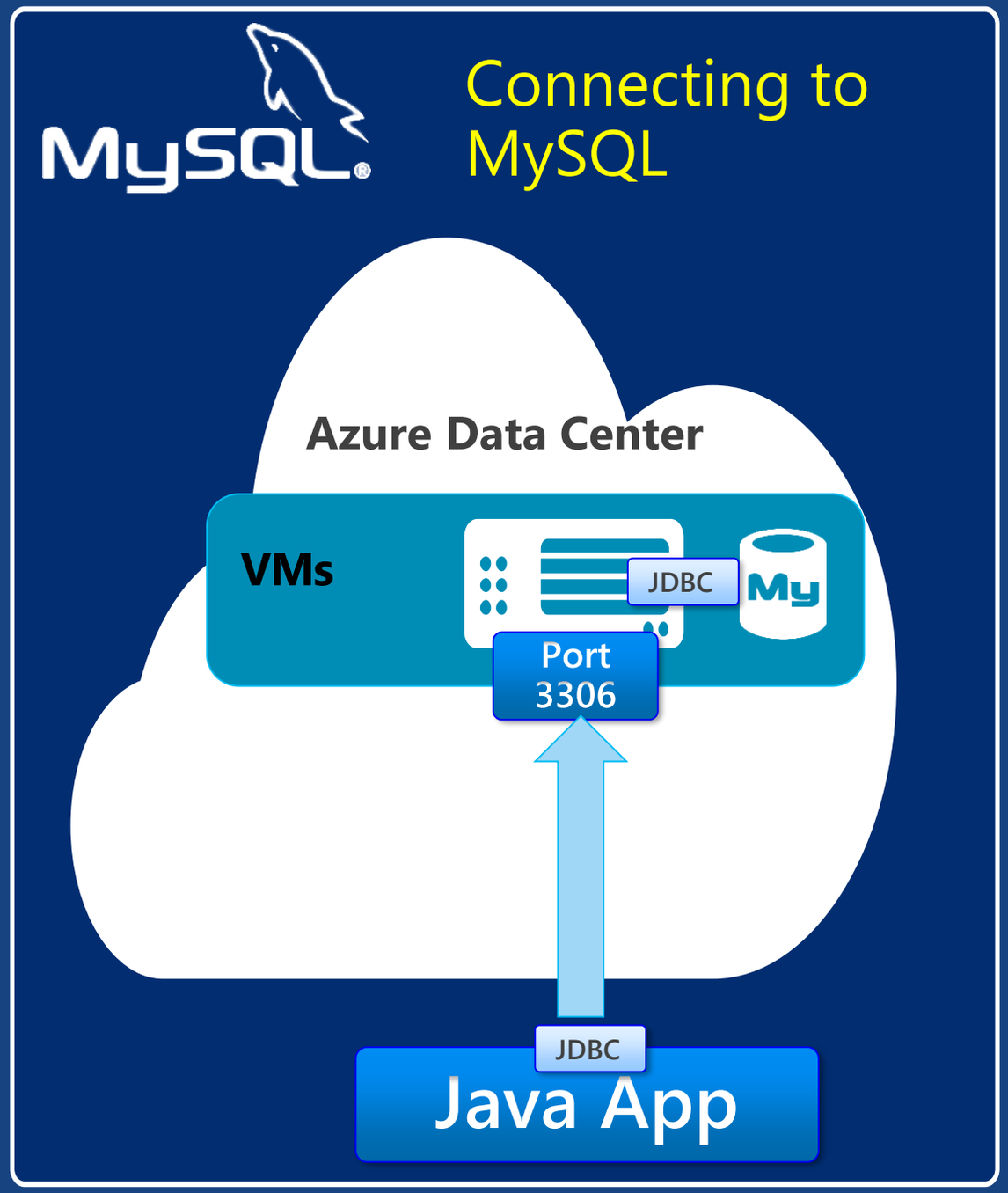
Oracle Linux 7.0

Oracle Web Logic

CONNECTING TO MYSQL FROM JAVA – PART 1

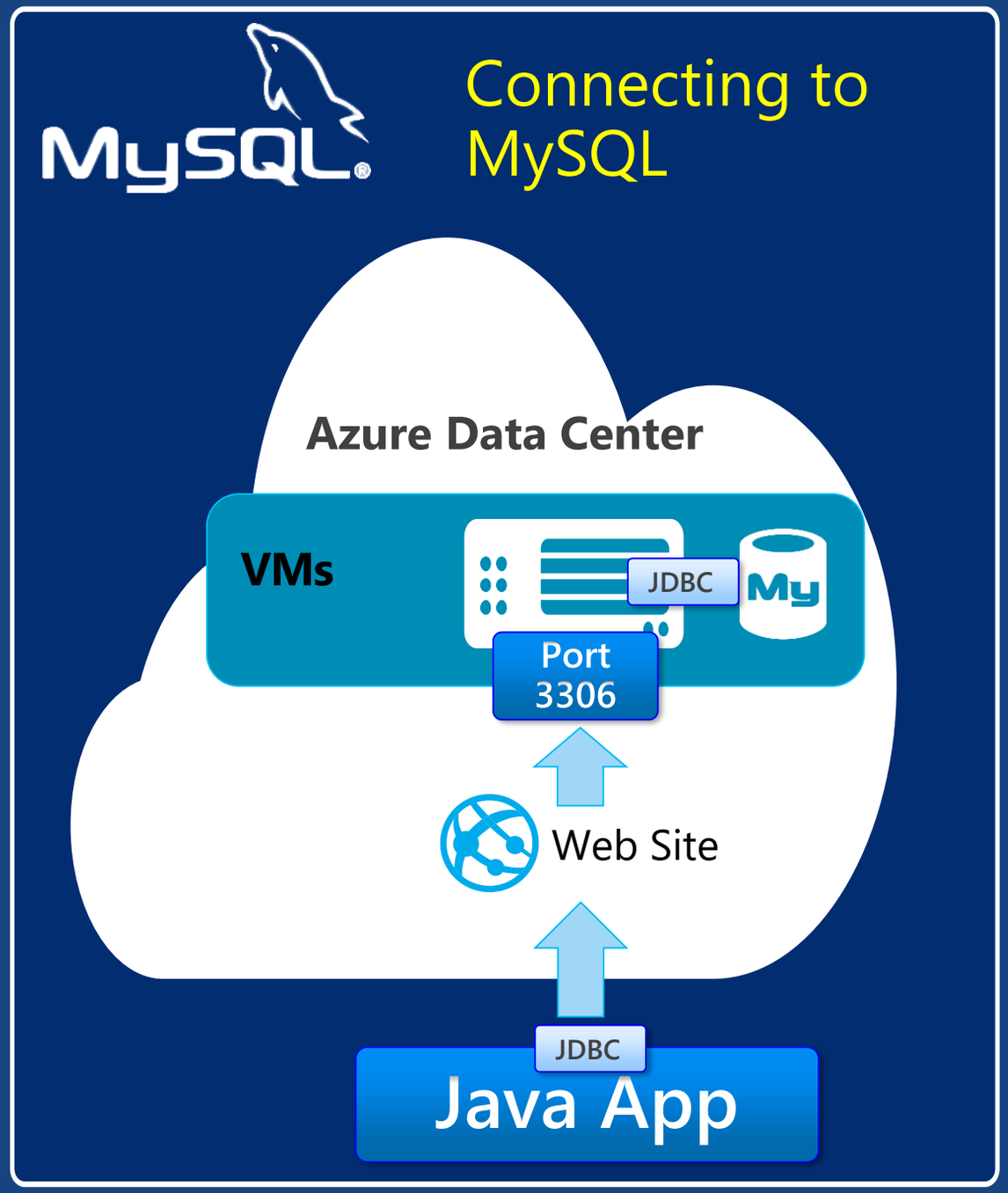
Connecting to MySQL from Java

- MySQL listens to port 3306
- It isn't open by default
- We will use the Azure Cross Platform Tooling to open



Connecting to MySQL from Java

- A better architecture
- Going through a web site

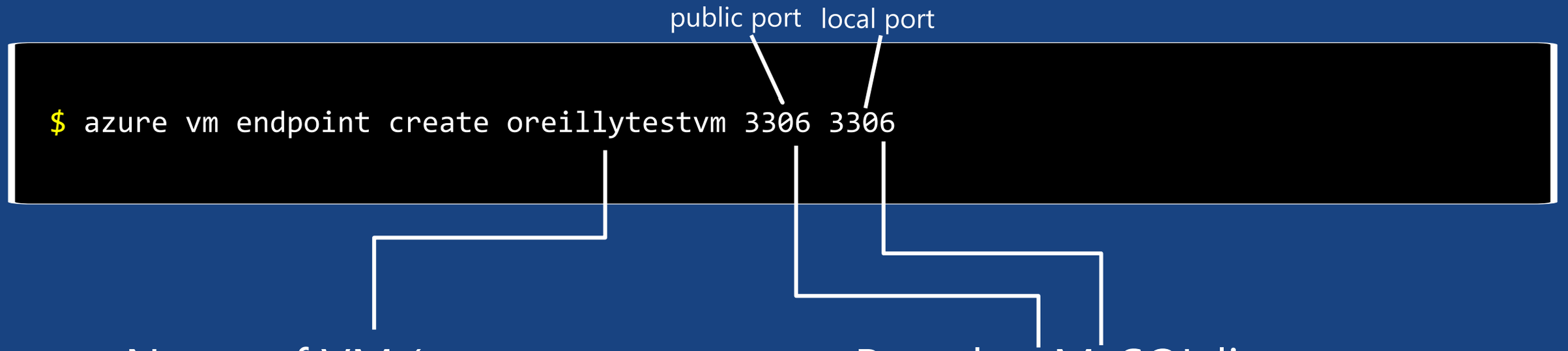


Setting Up MySQL/JDBC Driver on Ubuntu

```
$ sudo apt-get install libmysql-java
```

JDBC drivers for Ubuntu

Opening up port 3306



Opening a connection to the database

- A few modifications are needed
- Granting privileges

```
# Get to a mysql prompt
$
GRANT ALL PRIVILEGES ON *.* TO
'root'@'localhost' IDENTIFIED BY 'root' WITH
GRANT OPTION;
flush privileges;
```

password (replace with a better one)

- Modifying my.cnf, changing to 0.0.0.0

vim /etc/mysql/my.cnf

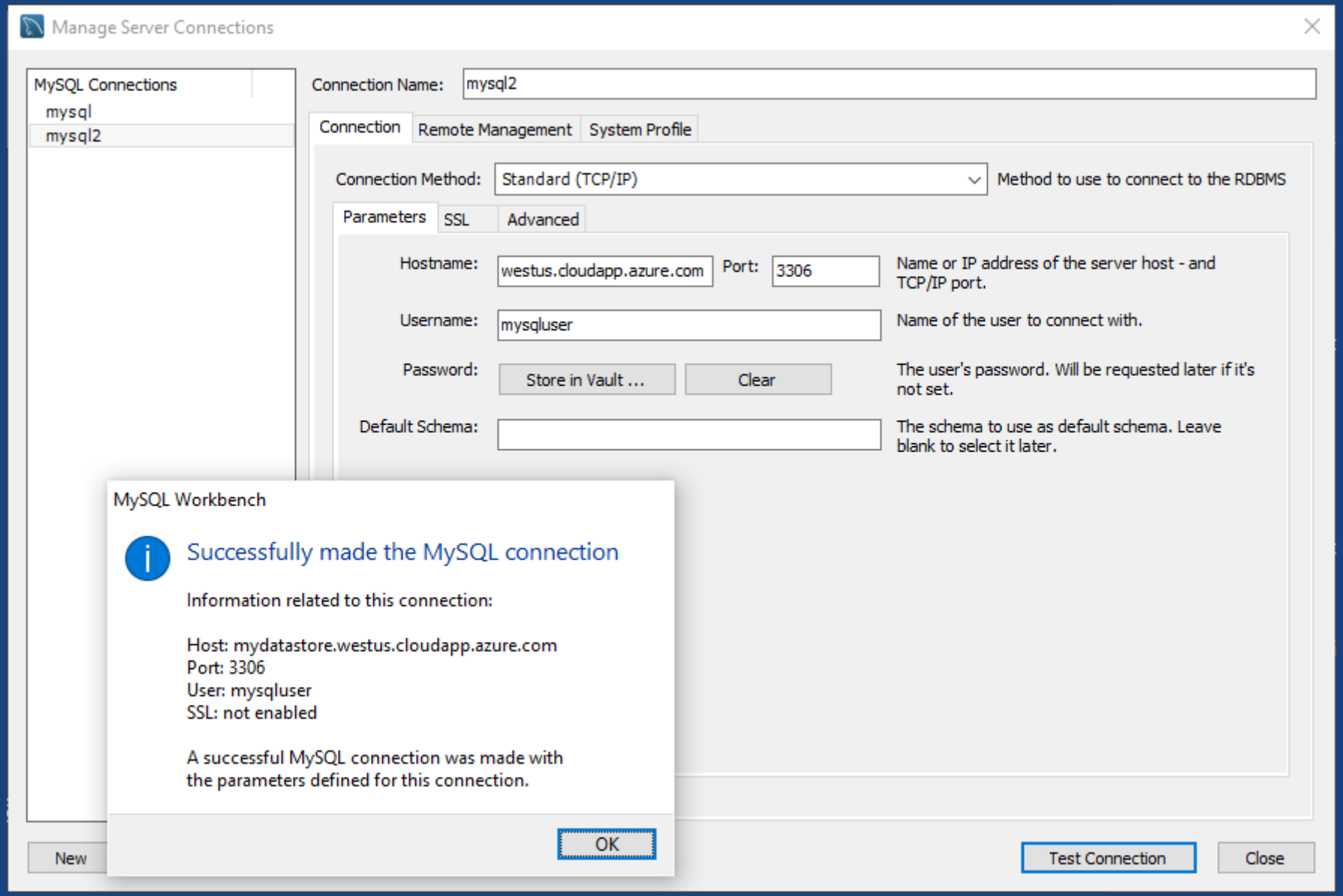
```
# Change the IP address
# bind-address to 0.0.0.0

# COMMENTED OUT
# bind-address = 127.0.0.1

bind-address            = 0.0.0.0
```

reboot after edit

Creating a user



Creating a user

```
mysql> create database azurecoure;
mysql> create user 'mysqluser'@'localhost' identified by 'password';
mysql> grant all on azurecoure.* to 'mysqluser'@'%' identified by 'password';
```

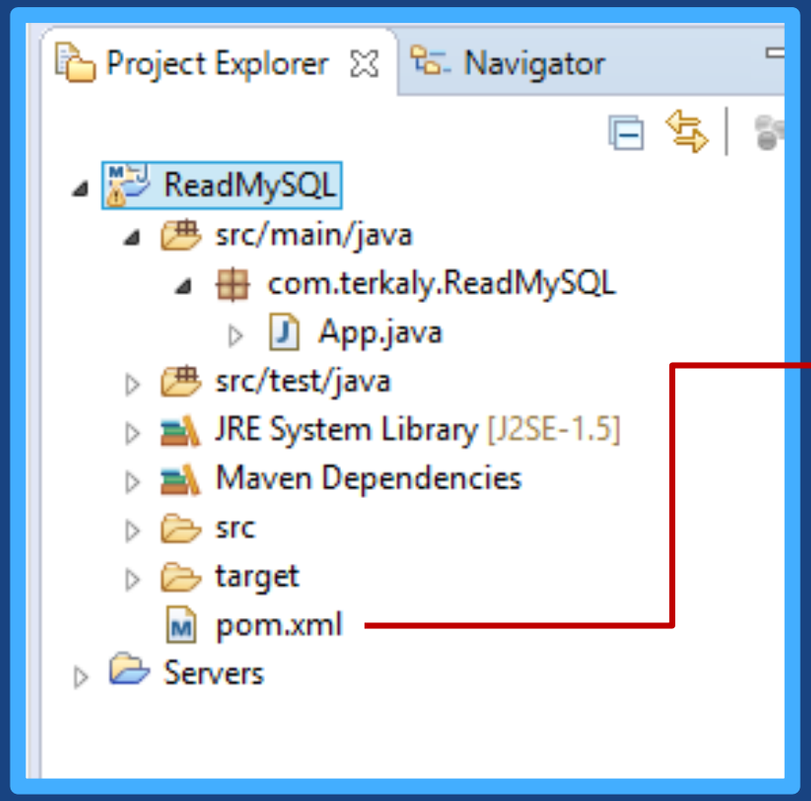
```
http://www.yougetsignal.com/tools/open-ports/
```

Listing Open Ports

```
$  azure vm endpoint list oreillytestvm
```

Name of VM (you can see it on the command line)

Setting Up MySQL/JDBC Driver on Java Client




```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.36</version>
</dependency>
```

Inserting Data with Java into MySQL

```
package com.terkaly.ReadMySQL;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public session App {
    static private Connection connect = null;
    static private Statement statement = null;
    static private PreparedStatement preparedStatement = null;
    static private ResultSet resultSet = null;
    public static void main(String[] args) throws
                InstantiationException,
                IllegalAccessException, ClassNotFoundException {

        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            // Setup the connection with the DB
            connect = DriverManager.getConnection(
                "jdbc:mysql://104.42.126.213:3306/azuresession?" +
                "user=root&password=root&"  +
                   "useUnicode=true&characterEncoding=UTF-8");
```

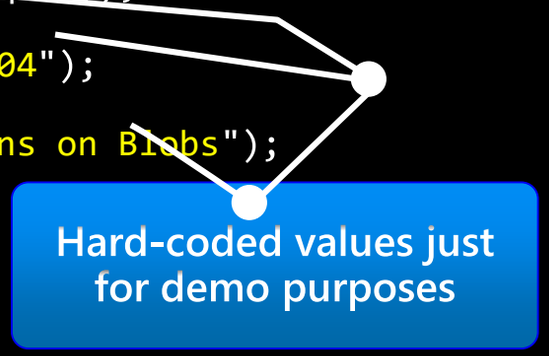
Continued…

IP Address or DNS Name of your MySQL VM

Database = azuresession

Inserting Data with Java into MySQL

```
            String query =
                "insert into sessions(id, sessionnumber, sessiontitle)
" +
                "values(?, ?, ?)";
            PreparedStatement preparedStmt =
                connect.prepareStatement(query);
            preparedStmt.setInt(1, 4);
            preparedStmt.setString(2, "0404");
            preparedStmt.setString(3,
                "Performing CRUD Operations on Blobs");
            preparedStmt.executeUpdate();
            connect.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



1. A prepared statement or parameterized statement is a feature used to execute the same or similar database statements repeatedly with high efficiency

2. It is used with SQL statements such as queries or updates

3. It takes the form of a template into which certain constant values are substituted during each execution

Selecting data using Java to query MySQL

```
public static void PerformSelectOfData() throws SQLException {
    // Statements allow to issue SQL queries to the database
    Statement statement;
    try {
        statement = connect.createStatement();
        resultSet =
          statement.executeQuery("select * from azuresession.sessions");
        WriteResults(resultSet);
    } catch (Exception e) {

        e.printStackTrace();
    }
    // Result set get the result of the SQL query
}
```

Displaying data using Java to query MySQL

```
public static void WriteResults(ResultSet resultSet) throws SQLException {
    System.out.println("The columns in the table are: ");

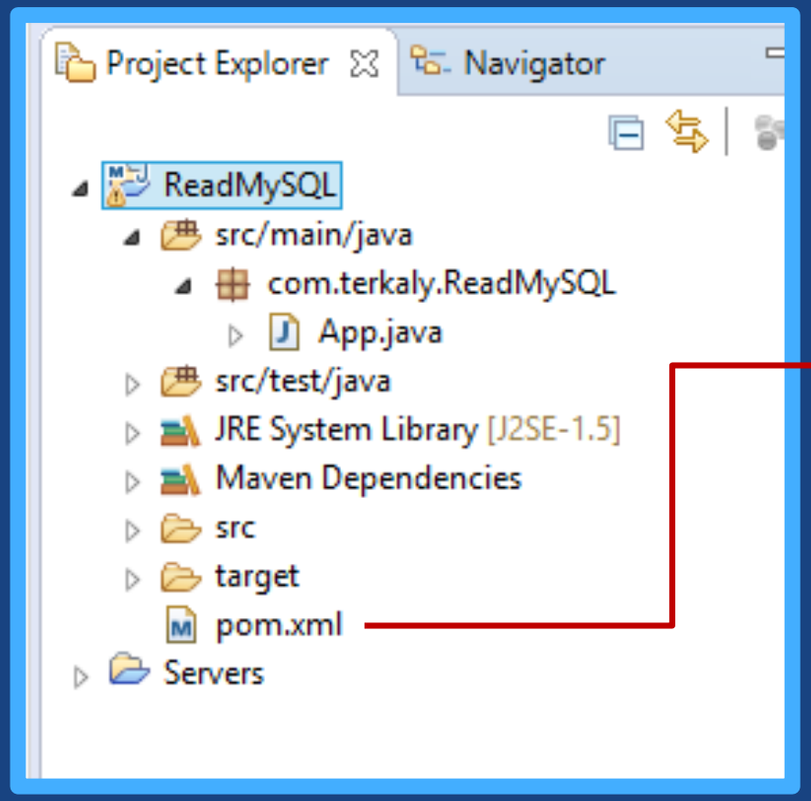
    try {
        System.out.println("Table: " + resultSet.getMetaData().getTableName(1));
        for (int i = 1; i <= resultSet.getMetaData().getColumnCount(); i++) {
            System.out.println("Column " + i + " " +
                                resultSet.getMetaData().getColumnName(i));
        }

        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String sessionnumber = resultSet.getString("sessionnumber");
            String sessiontitle = resultSet.getString("sessiontitle");
            System.out.println("ID: " + String.valueOf(id));
            System.out.println("Session Number: " + sessionnumber);
            System.out.println("Session Title: " + sessiontitle);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

CONNECTING TO MYSQL FROM JAVA – PART 2

CONNECTING TO MYSQL FROM JAVA – PART 2 (HANDS-ON)

Setting Up MySQL/JDBC Driver on Java Client




```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.36</version>
</dependency>
```

Inserting Data with Java into MySQL

```
package com.terkaly.ReadMySQL;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public session App {
    static private Connection connect = null;
    static private Statement statement = null;
    static private PreparedStatement preparedStatement = null;
    static private ResultSet resultSet = null;
    public static void main(String[] args) throws
                InstantiationException,
                IllegalAccessException, ClassNotFoundException {

        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            // Setup the connection with the DB
            connect = DriverManager.getConnection(
                "jdbc:mysql://104.42.126.213:3306/azuresession?" +
                "user=root&password=root&"  +
                   "useUnicode=true&characterEncoding=UTF-8");
```

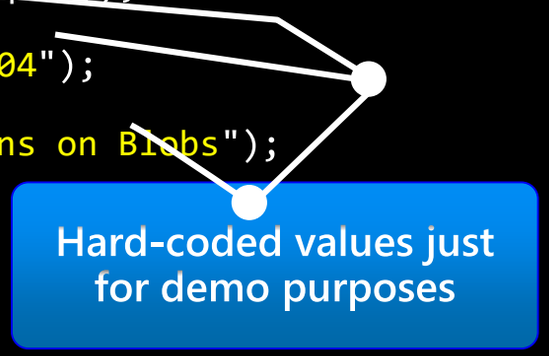
Continued…

IP Address or DNS Name of your MySQL VM

Database = azuresession

Inserting Data with Java into MySQL

```
            String query =
                "insert into sessions(id, sessionnumber, sessiontitle)
" +
                "values(?, ?, ?)";
            PreparedStatement preparedStmt =
                connect.prepareStatement(query);
            preparedStmt.setInt(1, 4);
            preparedStmt.setString(2, "0404");
            preparedStmt.setString(3,
                "Performing CRUD Operations on Blobs");
            preparedStmt.executeUpdate();
            connect.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



1. A prepared statement or parameterized statement is a feature used to execute the same or similar database statements repeatedly with high efficiency

2. It is used with SQL statements such as queries or updates

3. It takes the form of a template into which certain constant values are substituted during each execution

Selecting data using Java to query MySQL

```
public static void PerformSelectOfData() throws SQLException {
    // Statements allow to issue SQL queries to the database
    Statement statement;
    try {
        statement = connect.createStatement();
        resultSet =
          statement.executeQuery("select * from azuresession.sessions");
        WriteResults(resultSet);
    } catch (Exception e) {

        e.printStackTrace();
    }
    // Result set get the result of the SQL query
}
```

Displaying data using Java to query MySQL

```
public static void WriteResults(ResultSet resultSet) throws SQLException {
    System.out.println("The columns in the table are: ");

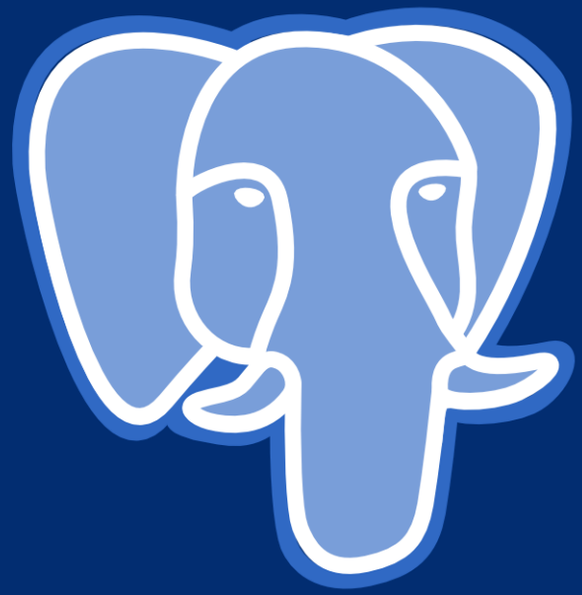
    try {
        System.out.println("Table: " + resultSet.getMetaData().getTableName(1));
        for (int i = 1; i <= resultSet.getMetaData().getColumnCount(); i++) {
            System.out.println("Column " + i + " " +
                                resultSet.getMetaData().getColumnName(i));
        }

        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String sessionnumber = resultSet.getString("sessionnumber");
            String sessiontitle = resultSet.getString("sessiontitle");
            System.out.println("ID: " + String.valueOf(id));
            System.out.println("Session Number: " + sessionnumber);
            System.out.println("Session Title: " + sessiontitle);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

INTRODUCING POSTGRES

Apache PostGres

- PostgreSQL an object-relational database management system (RDBMS)
- PostgreSQL is cross-platform and runs on many operating systems including:
  - Linux
  - FreeBSD
  - OS X
  - Solaris
  - Microsoft Windows



PostgreSQL

- Protects against dirty reads and full serializability
- Handles complex SQL queries

Apache PostGres

- Supports indexing methods that are not available in other databases
  - Updateable views
  - Materialized views
  - Triggers
  - Foreign keys

- A view is a virtual table representing the result of a database query
  - An updatable view is useful because you can provide a stable interface to a client and still change the structure of underlying tables

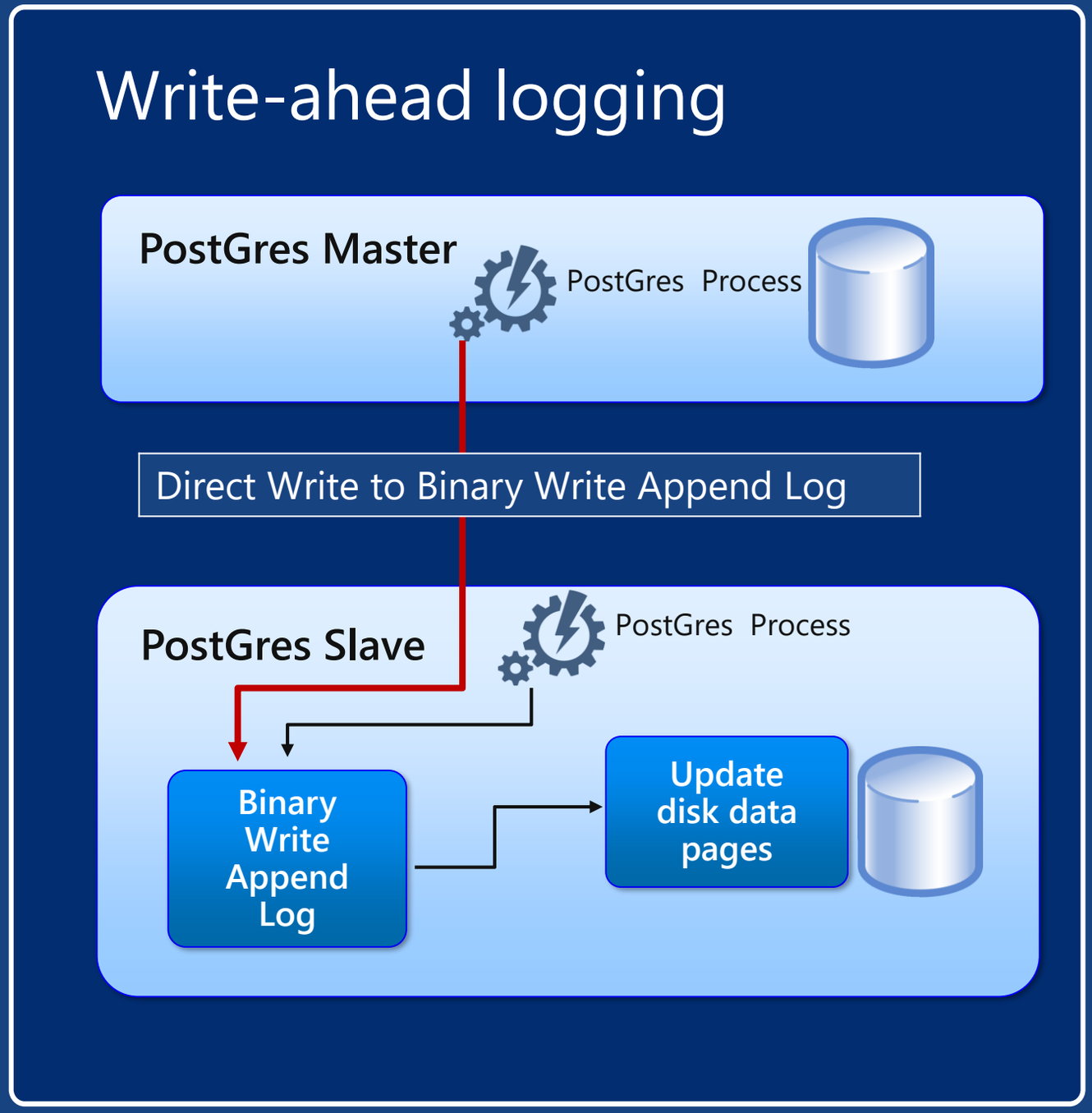
Materialized View

- A object that contains the results of a query
- Could be:
  - A local copy of data located remotely
  - A subset of the rows and/or columns of a table or join result
  - A summary using an aggregate function.

Apache PostGres

Built-in replication

- Supports a complete replica of your database
- Prevents all but a couple seconds of data loss
- Even under catastrophic circumstances
- Load balance between write master server and multiple read-only slave servers
- Run reporting or other long-running queries on a replica server
- Reduce load on main transaction-processing server



Apache PostGres

User Defined Types

- The ability to create types
- You specify an existing base
- You can create your own base types in C
- Useful when defining contraints on a type
- Example
  - CREATE DOMAIN custInitials AS text CHECK (value ~ '^[a-z][a-z]$');

You can create your own base types beyond scalar types, like “text,” “double” etc

Creating a new base type requires implementing functions to operate on the type in a low-level language, usually C.

Could also be enums

- CREATE TYPE shirtsize AS ENUM (‘large', ‘medium', ‘small');

- You must define the Input and output functions

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

Tunable Levels of Durability

- Transaction durability is tunable - can be specified per-database, per-user, per-session or even per-transaction
- Some workloads need to favor performance over data integrity
- There can be a mixture of synchronous and asynchronous standby servers

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = Shared Buffers

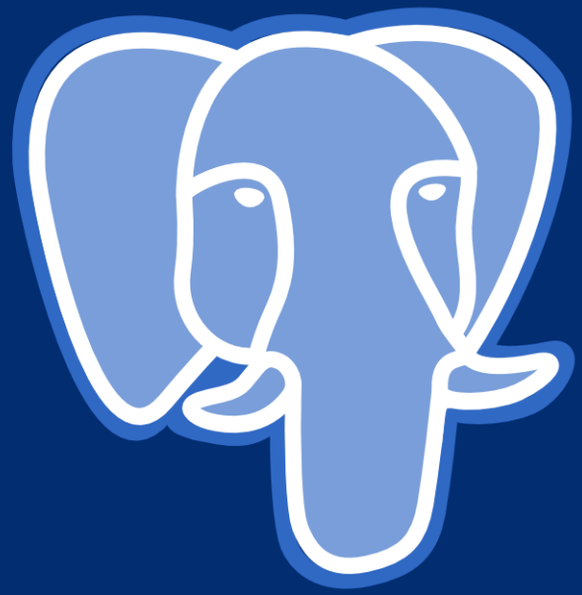
- How much memory is dedicated to caching data
- Avoid need to recompile Linux kernel for large chunks of shared memory
- PostGres starts with 25% as a target for percentage of memory dedicated to shared buffers
- Sometimes OS-level caching might be more performant

- Shared Buffers ⇦
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

INTRODUCING POSTGRES

Apache PostGres

- PostgreSQL an object-relational database management system (RDBMS)
- PostgreSQL is cross-platform and runs on many operating systems including:
  - Linux
  - FreeBSD
  - OS X
  - Solaris
  - Microsoft Windows



PostgreSQL

- Protects against dirty reads and full serializability
- Handles complex SQL queries

Apache PostGres

- Supports indexing methods that are not available in other databases
  - Updateable views
  - Materialized views
  - Triggers
  - Foreign keys

- A view is a virtual table representing the result of a database query
  - An updatable view is useful because you can provide a stable interface to a client and still change the structure of underlying tables

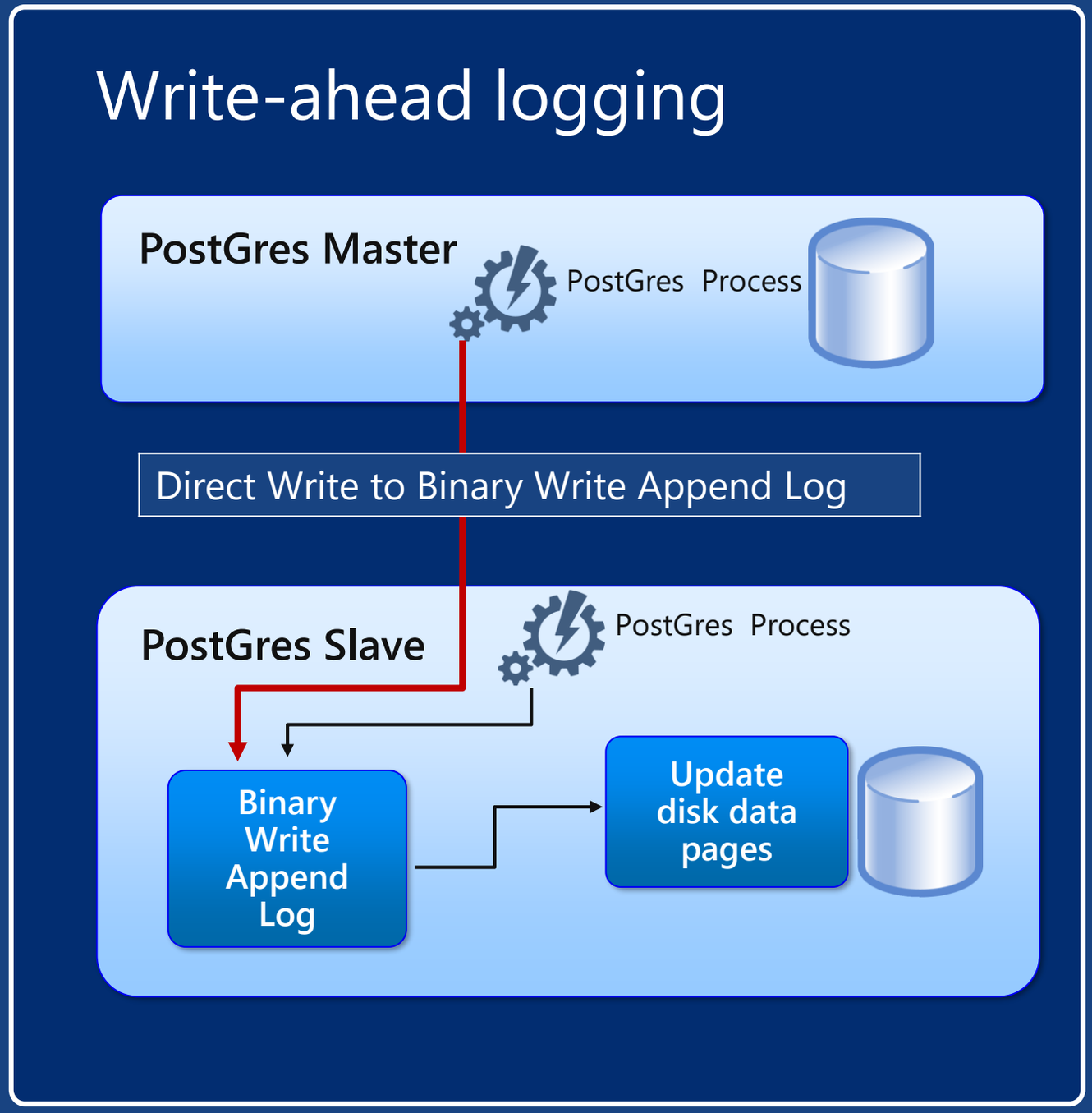
Materialized View

- A object that contains the results of a query
- Could be:
  - A local copy of data located remotely
  - A subset of the rows and/or columns of a table or join result
  - A summary using an aggregate function.

Apache PostGres

Built-in replication

- Supports a complete replica of your database
- Prevents all but a couple seconds of data loss
- Even under catastrophic circumstances
- Load balance between write master server and multiple read-only slave servers
- Run reporting or other long-running queries on a replica server
- Reduce load on main transaction-processing server



Apache PostGres

User Defined Types

- The ability to create types
- You specify an existing base
- You can create your own base types in C
- Useful when defining contraints on a type
- Example
  - CREATE DOMAIN custInitials AS text CHECK (value ~ '^[a-z][a-z]$');

You can create your own base types beyond scalar types, like “text,” “double” etc

Creating a new base type requires implementing functions to operate on the type in a low-level language, usually C.

Could also be enums

- CREATE TYPE shirtsize AS ENUM (‘large', ‘medium', ‘small');

- You must define the Input and output functions

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

Tunable Levels of Durability

- Transaction durability is tunable - can be specified per-database, per-user, per-session or even per-transaction
- Some workloads need to favor performance over data integrity
- There can be a mixture of synchronous and asynchronous standby servers

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = Shared Buffers

- How much memory is dedicated to caching data
- Avoid need to recompile Linux kernel for large chunks of shared memory
- PostGres starts with 25% as a target for percentage of memory dedicated to shared buffers
- Sometimes OS-level caching might be more performant

- Shared Buffers ⇦
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

INTRODUCING POSTGRES
PART 2

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = Max Connections

- Maximum number of client connections allowed
- On good hardware expect a few hundred connnections
- Consider connection pooling software to reduce the connection overhead

- Shared Buffers
- Max Connections ⇦
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = listen_addresses

- By default, PostgreSQL only responds to connections from the local host.
- To support TCP/IP networking and connectivity outside the VM, you need to change listen_addresses from its default
- Change to listen_addresses = '*'
- And then control who can and cannot connect via the pg_hba.conf file

- Shared Buffers
- Max Connections
- listen_addresses 
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = effective_cache_size

- Memory for disk caching by the operating system and within the database itself
- Takes into account the OS and other applications
- Used by the Query Planner
- A Low effective cache size may lead to faulty index usage
- 1/2 of total memory would be a normal conservative setting
- 3/4 of memory is a more aggressive but still reasonable amount

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = checkpoint_segments

- PostgreSQL writes new transactions to the database in files called WAL segments that are 16MB
- For normal systems set to 10
- For write heavy set to 32 (checkpoint every 512MB)
- For bulk loading can bump up to 64

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments ⇦
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = autovacuum

- Autovacuum process takes care of several maintenance chores inside your database that you really need
- You should set to true

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum ⇦
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

INTRODUCING POSTGRES
PART 3

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = logging

- pgFouine is a tool used to analyze postgresql logs
- PgCluu is an handy tool from the author of PgBadger, and is a PostgreSQL performances monitoring and auditing tool.
- log_min_error_statement to see errors
- log_min_duration_statement to find slow queries

and more

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging ⇦
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = wal_sync_method/wal_buffers

- After every transaction, PostgreSQL forces a commit to disk out to its write-ahead log
- Increasing wal_buffers from its tiny default of a small number of kilobytes is helpful for write-heavy systems
- Benchmarking generally suggests that just increasing to 1MB is enough for some large systems
- This data may become data as more powerful hardware is increasingly becoming available

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers ⟸
- max_prepared_transactions
- synchronous_commit
- random_page_cost

Apache PostGres

How to tune and tweek PostGres

Setting = max_prepared_transactions

- PREPARE TRANSACTION prepares the current transaction for two-phase commit
- Its purpose is to allow an external transaction manager to perform atomic global transactions across multiple databases
- PREPARE TRANSACTION is for distributed transactions across multiple servers, usually used by transaction monitors

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions ⇦
- synchronous_commit

random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = synchronous_commit

- Specifies whether transaction commit will wait for WAL records to be written to disk before the command returns a "success" indication to the client

- Set synchronous_commit to off when performance is more important than durability of a transaction.

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit ⇦
- random_page_cost

Apache PostGres

https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

How to tune and tweek PostGres

Setting = random_page_cost

- Meant to represent the relative cost of looking up one row (out of many) via sequential reads, vs. the cost of looking up a single row individually using random access (disk seeks)

- It influences the planner's decisions to use:
  - table scan
  - composite indexes
  - simple indexes

- Shared Buffers
- Max Connections
- listen_addresses
- effective_cache_size
- checkpoint_segments
- autovacuum
- logging
- wal_sync_method/wal_buffers
- max_prepared_transactions
- synchronous_commit
- random_page_cost ⟸

INSTALLING POSTGRES

Apache PostGres


```
$  apt-get install postgresql
$  apt-get install postgresql-contrib
```

- Using apt-get to perform the install

Apache PostGres


```
$  su postgres
$  createdb azuresession
$  psql -d azuresession

$  create table sessions(id int, sessionnumber varchar(128),
       sessiontitle varchar(512));
```

- Leverage postgres account
- Create database
- Get to sql prompt and create table inside azuresession database

Apache PostGres


```
$  sudo -u postgres psql postgres
$  \password postgres
```

- Set the password for the “postgres” account
- The “postgres” account is added during installation

We will need these credentials when we write our Java Client Code

Apache PostGres


```
$  iptables -A INPUT -s 40.78.31.35 -p tcp --destination-port 5432 -m state --state
NEW,ESTABLISHED -j ACCEPT

$  iptables -A OUTPUT -d  40.78.31.35 -p tcp --source-port 5432 -m state --state
ESTABLISHED -j ACCEPT
```

- Open up ports
- May or may not be needed depending on how your VM is setup
- 40.78.31.35 is the public IP address of our VM
- PostGres uses the 5432 port for client connections

Apache PostGres


```
$  vim /etc/postgresql/9.3/main/postgresql.conf
$  vim /etc/postgresql/9.3/main/pg_hba.conf
$  /etc/init.d/postgresql restart
```

```
55 #------------------------------------------------------------------------------
56
57 # - Connection Settings -
58
59 listen_addresses = '*'          # what IP address(es) to listen on;
```

postgresql.conf

```
15 # Your host entry needs to look like you see it below
16 host      all  all  all  trust
```

pg_hba.conf

```
$  azure network nsg rule create -g oreilly-rg -a oreillybigdata -n cassandra-rule1
-c Allow -p '*' -r Inbound -y 100 -f Internet -o '*' -e '*' -u '*'
```

- Adds a rule to network security

Resource Group	oreilly-rg
Network Security Group	oreillybigdata
Network security rule	Cassandra-rule1
What the rule does	Allows incoming/outgoing TCP/UDP traffic on VM

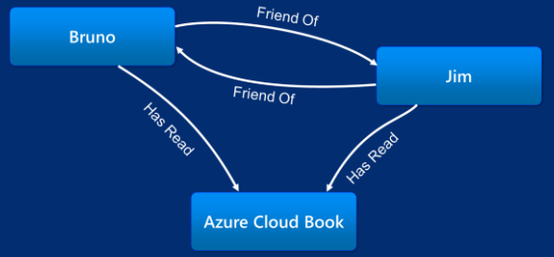
NoSQL Data Stores

- Data storage and retrieval that is non-tabular/relational
- Benefits include simplicity, easier scaling
- Higher availability
- Key-value, graph, document

Types of NoSQL

- **Key-value**
  Demonstrated Previously

- **Graph**



- **Document**
  See DocumentDB and MongoDB Hands-On

```
{
    "data": [
    {
        "id": "X999_Y999",
        "from": {
        "name": “Bruno Terkaly",
        "idsrc": "X12"
        },
    }
}
```

NoSQL Data Stores

- Not all data fits into 2 dimensions
- Volatility of the data model
- Likely to change and evolve?
- Schema rigidity
- Web-centric apps need flexibility

Why or why not NoSQL

- Dependency on dedicated DBAs
- Developers like object focused nature of JSON
- Vertical scaling is expensive
- Distributed file systems
- Reporting needs
- Real time analytics

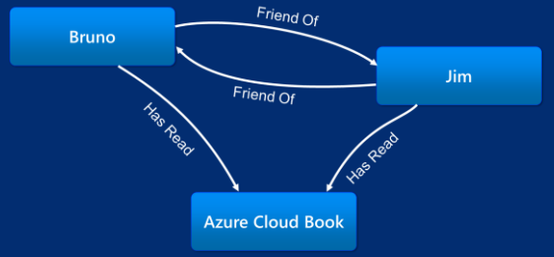
NoSQL Data Stores

- Data storage and retrieval that is non-tabular/relational
- Benefits include simplicity, easier scaling
- Higher availability
- Key-value, graph, document

Types of NoSQL

- **Key-value**
  Demonstrated Previously

- **Graph**



- **Document**
  See DocumentDB and MongoDB Hands-On

```
{
    "data": [
    {
        "id": "X999_Y999",
        "from": {
        "name": “Bruno Terkaly",
        "idsrc": "X12"
        },
    }
}
```

NoSQL Data Stores

- Not all data fits into 2 dimensions
- Volatility of the data model
- Likely to change and evolve?
- Schema rigidity
- Web-centric apps need flexibility

Why or why not NoSQL

- Dependency on dedicated DBAs
- Developers like object focused nature of JSON
- Vertical scaling is expensive
- Distributed file systems
- Reporting needs
- Real time analytics

Data Stores that run on Azure

- PaaS Offerings
  - DocumentDB (JSON)
  - Azure Tables (KeyValue)
- IaaS
  - Everything else – Azure is open
  - Cassandra, Mongo, etc

Why DocumentDB?

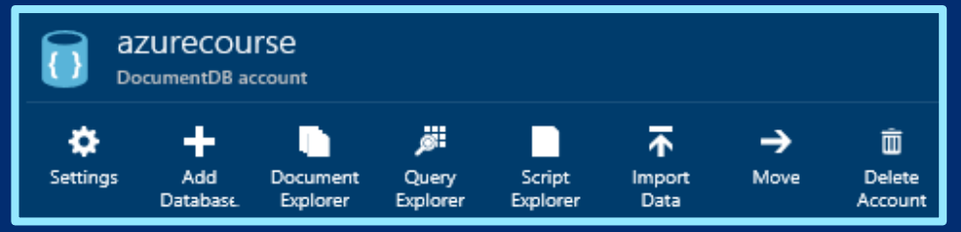
- Need for managed service
- No pre-defined schema, or indexes
- A variety of clients
- Velocity and variability not known
- Scalable, fast deployment
- Low cost
- Query Language
- Support for transactions
- JSON-based
- REST-based access

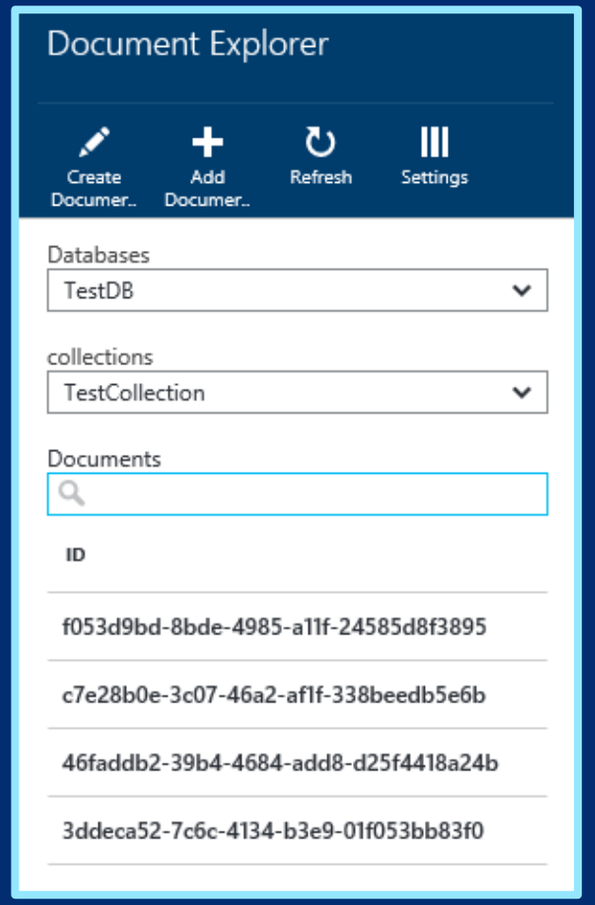
PROVISIONING DOCUMENTDB DATABASES

Using the portal to interact with your data

- Add database
- Explore documents
- Execute queries
- View scripts
- Import data

DocumentDB Portal





Databases

Collections

Documents

CONNECTING TO AZURE DOCUMENTDB WITH JAVA

DocumentDB Code

1. Instantiate a DocumentClient w/ your DocumetnDB Endpoint and AuthKey.
2. Instantiate a database object
3. Query for a database with the ID of TestDB
4. If does not exist, then create. Else get a reference to it.
5. Define a new collection using the id above.
6. Query for a collection called TestCollection
7. If collection not found, create
8. Create an object, serialize it in to JSON, and wrap it in to a document.
9. Insert document into document collection
10. Retrieve all inserted documents
11. If find at least one, retrieve the first one
12. Convert to a real object
13. Print out the first name

- DocumentClient documentClient
- Database myDatabase
- List<Database> databases
- DocumentCollection myCollection
- RequestOptions requestOptions
- SqlQuerySpec sqlQuerySpec
- List<DocumentCollection> collections
- Person person = new Person();
- Document myDocument
- List<Document> documentList

CONNECTING TO AZURE DOCUMENTDB WITH JAVA – PART 2

DocumentDB Code

1. Instantiate a DocumentClient w/ your DocumetnDB Endpoint and AuthKey.
2. Instantiate a database object
3. Query for a database with the ID of TestDB
4. If does not exist, then create. Else get a reference to it.
5. Define a new collection using the id above.
6. Query for a collection called TestCollection
7. If collection not found, create
8. Create an object, serialize it in to JSON, and wrap it in to a document.
9. Insert document into document collection
10. Retrieve all inserted documents
11. If find at least one, retrieve the first one
12. Convert to a real object
13. Print out the first name

- DocumentClient documentClient
- Database myDatabase
- List<Database> databases
- DocumentCollection myCollection
- RequestOptions requestOptions
- SqlQuerySpec sqlQuerySpec
- List<DocumentCollection> collections
- Session session, Rating rating;
- Document myDocument
- List<Document> documentList

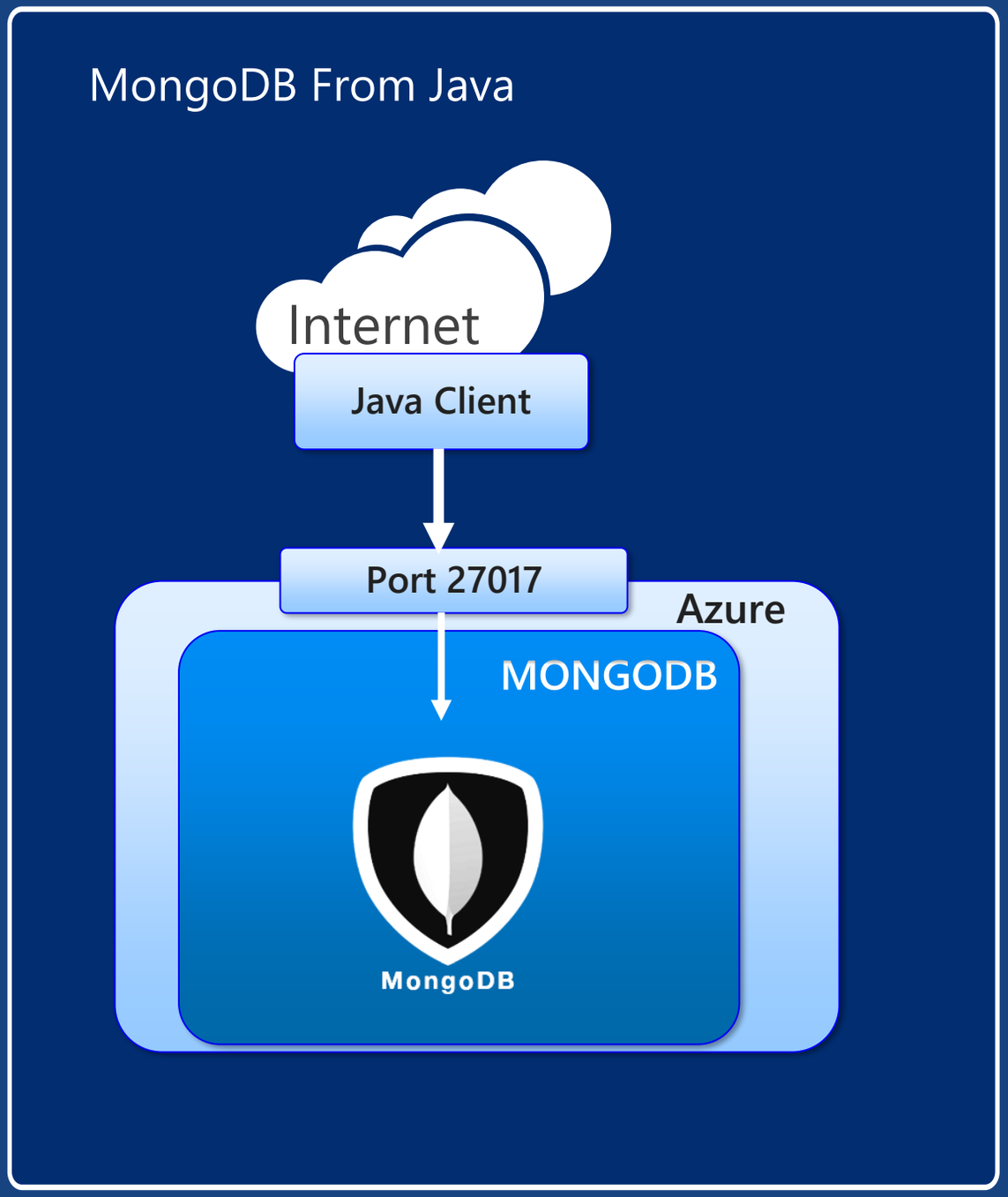
CONNECTING TO AZURE DOCUMENTDB WITH JAVA – PART 3

CONNECTING TO AZURE DOCUMENTDB WITH JAVA – PART 4

UNDERSTANDING MONGODB DATABASES

Understanding MongoDB

- MongoDB is a cross-platform document-oriented database
- Is a NoSQL database
- Data stored as JSON-like documents with dynamic schemas
- MongoDB calls the format BSON



Understanding MongoDB

- A popular combination is Node.js + Mongo + JavaScript
- Makes the integration of data in certain types of applications easier and faster
- Used by Craigslist, eBay, and Foursquare among others
- As of July 2015, MongoDB is the fourth most popular type of database management system

MongoDB and Node.js

- Node's is all about JavaScript
- This means that MongoDB documents get their most natural representation
  - JSON everywhere
    - the application layer
    - the database layer

Understanding MongoDB

- Document-oriented
  - Can store relational objects in one document
- Supprots Ad hoc queries
  - Supprots search by field, range queries, regular expression searches
- Indexing
  - Any field in a MongoDB document can be indexed
- Replication
  - MongoDB provides high availability with replica sets
- Load balancing
  - MongoDB scales horizontally using sharding using a shard key
- Aggregation
  - MapReduce can be used for batch processing of data and aggregation operations
  - Similar to the SQL GROUP BY clause
- Server-side JavaScript execution
  - JavaScript can be used in queries and aggregation functions
- Capped collections
  - Acts as a circular queue on overflow

PROVISIONING MONGODB DATABASES

Installing Mongo

```
$  # Import the public key used by the package management system.
$
$  # Create a list file for MongoDB.
$
$  # Reload local package database.
$
$  # Install the latest stable version of MongoDB.
$
$  # Start MongoDB
$
```

Installing Mongo

```
$ # Verify port where MongoDB is listening
$

$ # Open up ports
$

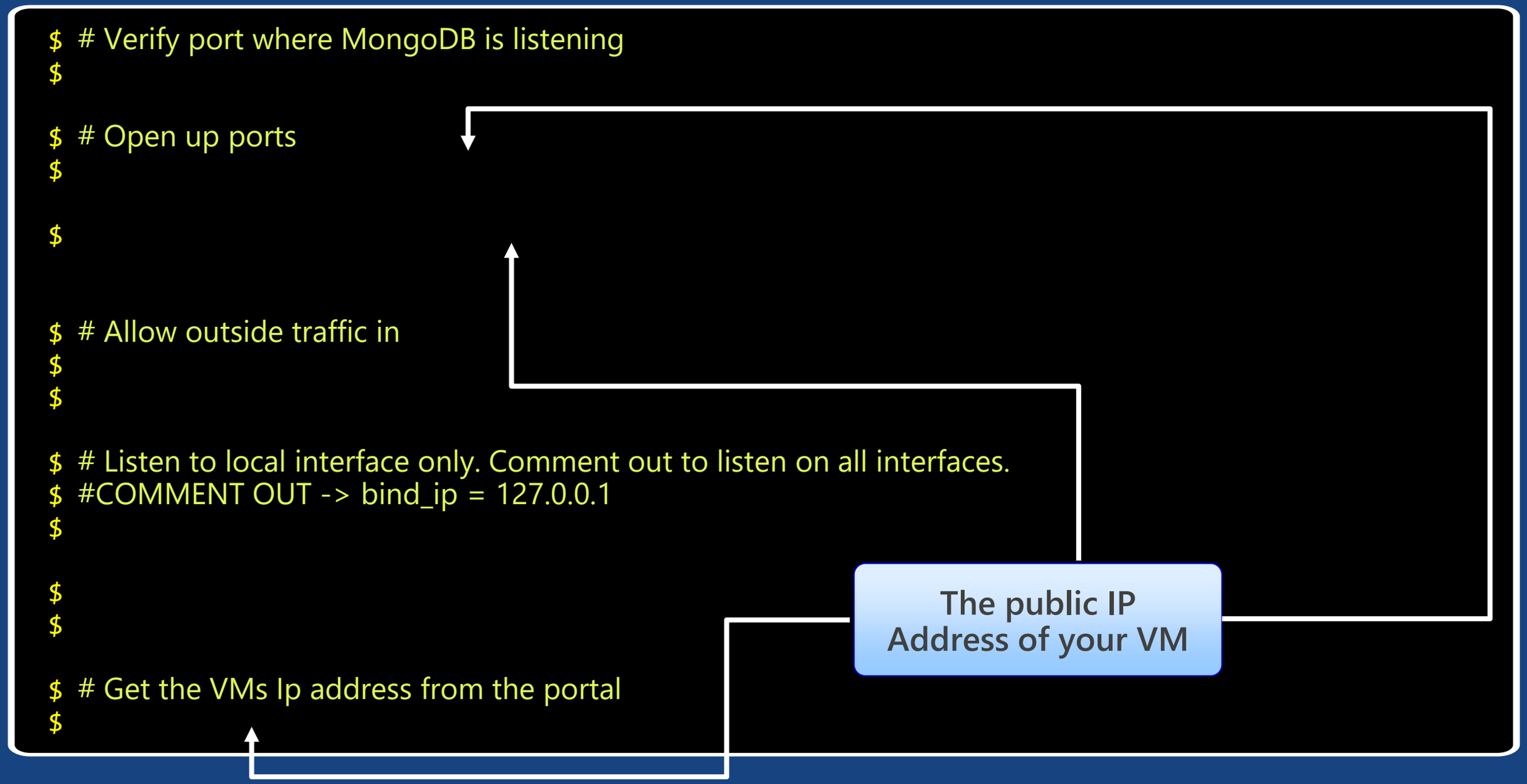
$


$ # Allow outside traffic in
$
$

$ # Listen to local interface only. Comment out to listen on all interfaces.
$ #COMMENT OUT -> bind_ip = 127.0.0.1
$

$
$

$ # Get the VMs Ip address from the portal
$
```



PROVISIONING MONGODB DATABASES PART 2

Installing Mongo

```
# BASH Provisioning Commands for MongoDB

````shell
# Import the public key used by the package management system.
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

# Create a list file for MongoDB.
echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb_release -sc)"/mongodb-org/3.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list

# Reload local package database.
sudo apt-get update

# Install the latest stable version of MongoDB.
sudo apt-get install -y mongodb-org

# Start MongoDB
sudo service mongod start

# Verify port where MongoDB is listening
vim /var/log/mongodb/mongod.log

# Open up ports
iptables -A INPUT -s 104.42.126.213 -p tcp --destination-port 27017 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -d  104.42.126.213 -p tcp --source-port 27017 -m state --state ESTABLISHED -j ACCEPT

// Allow outside traffic in
cd /etc
vim mongod.conf
# Listen to local interface only. Comment out to listen on all interfaces.
#bind_ip = 127.0.0.1
bind_ip = 0.0.0.0
rm /var/lib/mongodb/mongod.lock
reboot
#Should get your in
telnet  104.42.126.213 27017
```
```

INTRODUCING APACHE CASSANDRA

Apache Cassandra

- A free open source NoSQL database
- Designed to manage very large data sets
- Leverages distributed computing architectures using commodity hardware

Used by:

- CERN
- Comcast
- eBay
- GitHub
- GoDaddy
- Hulu
- Instagram
- Intuit
- Netflix
- Reddit
- The Weather Channe

Apache Cassandra

Popular because:

- Almost linear scalability and high availability
- Massive ingestion capabilities
- Built in support for replication across data centers
- Hybrid between a key-value and a column-oriented (or tabular) database.
- Performance of log structured updates
- Built in caching

- Open source
- Peer to peer architecture
- Schema-Free
- High Availability
- Fault tolerant
- Tunable Consistency
- Column Oriented
- Elastic Scale

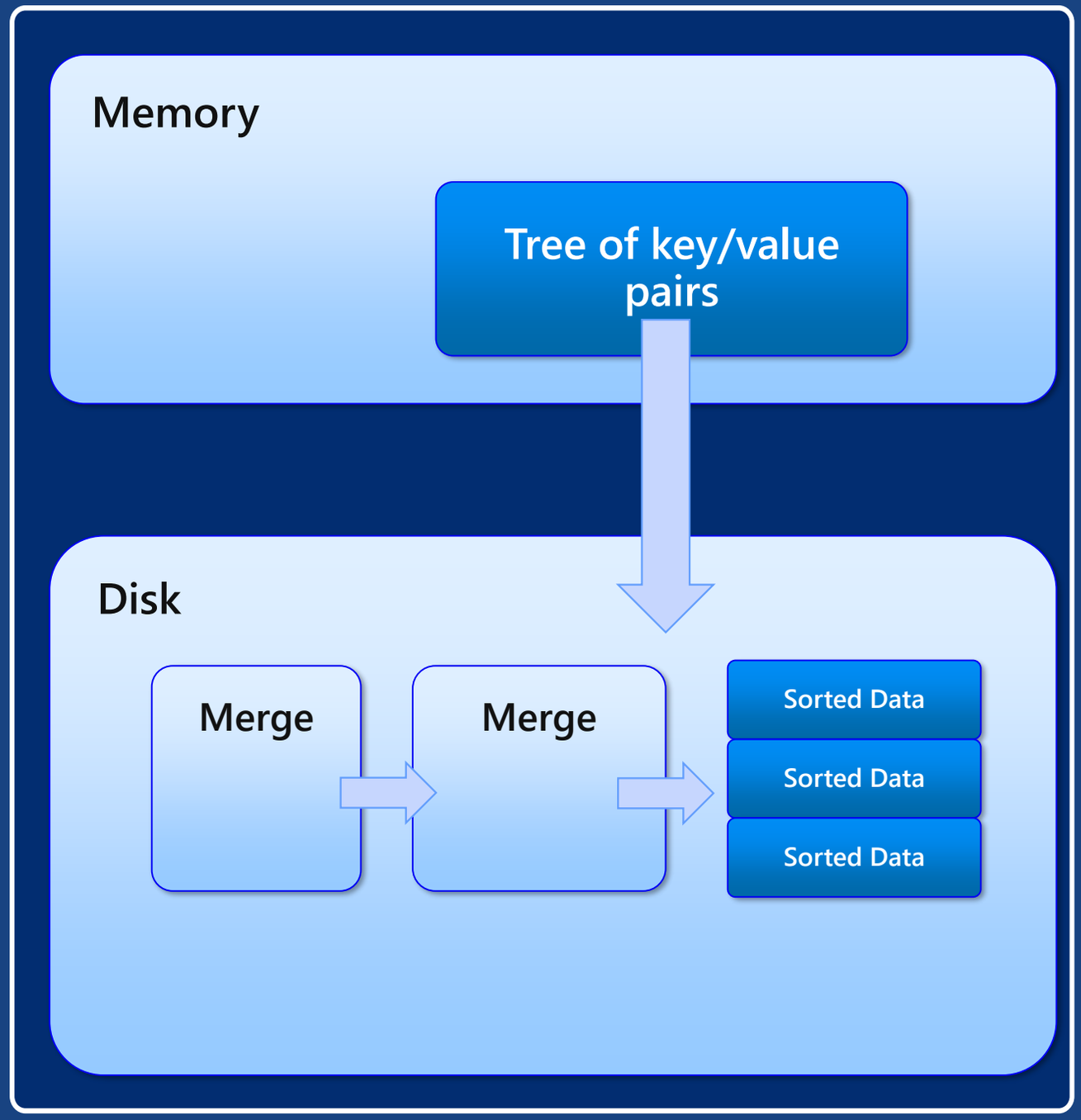
Apache Cassandra

- A column-oriented DBMS
  - Stores data as columns
- Faster data retrieval because columns are closer together
- This reduces data read from disk, processed by the cpu, and cached in memory
- Advantages for data warehouses, CRM systems, etc
- Sweet spot is for ad hoc inquiry systems

- Imagine that I wish to analyze product sales by store
- Only a few columns are going to be used (location, product id, sales)
- Better that this data is grouped together for efficiency purposes

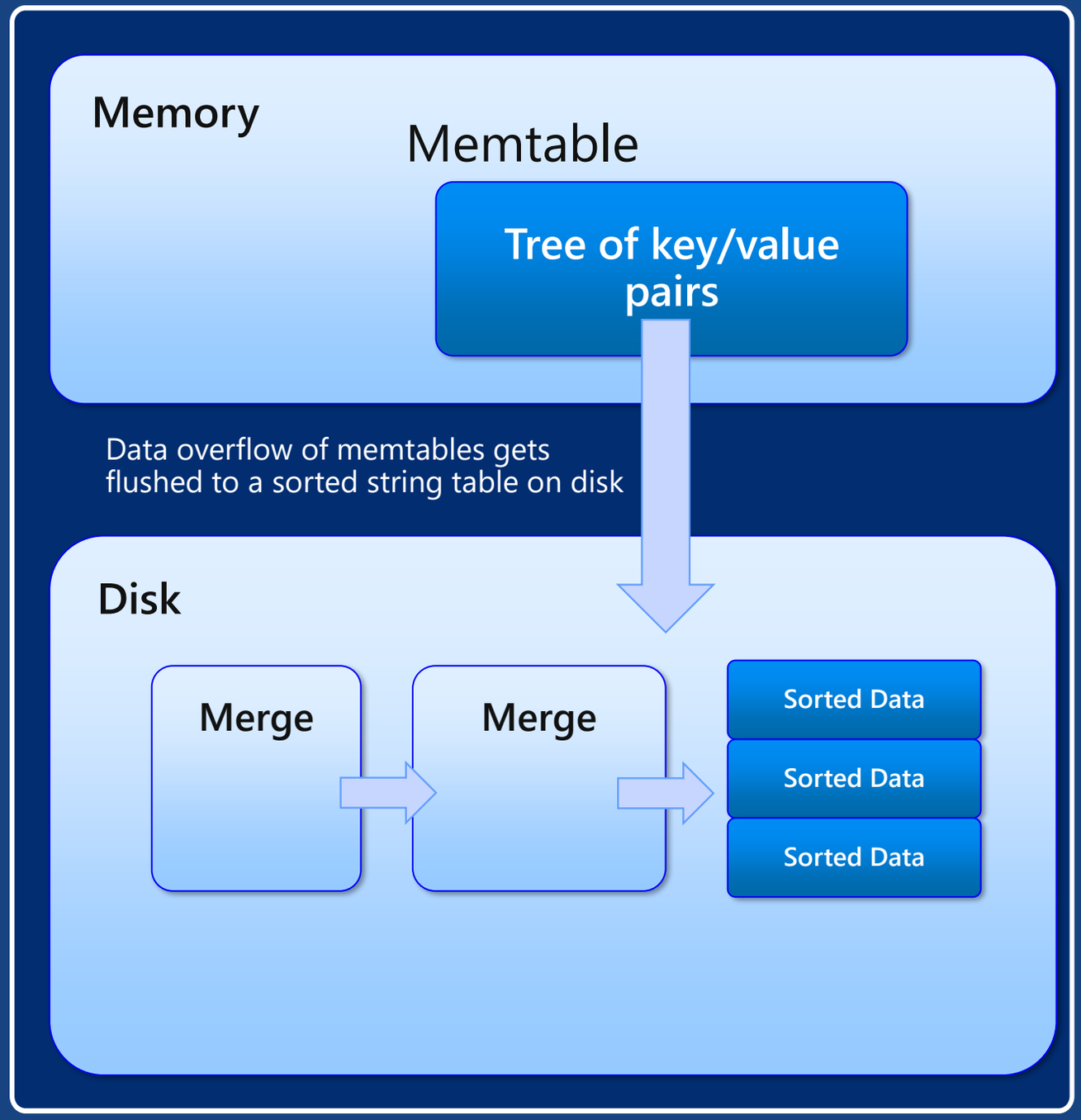
Apache Cassandra

- Near linear scale ability for read/write operations
- No master nodes - all "created equal" nodes in cluster
- Consists of a log-structured merge tree
  - Standard disk-based index structures could be slow
  - The B-tree will dramatically increase the I/O cost of the transaction to maintain an index such in real time
  - It is very important to support a real-time index at high efficiency



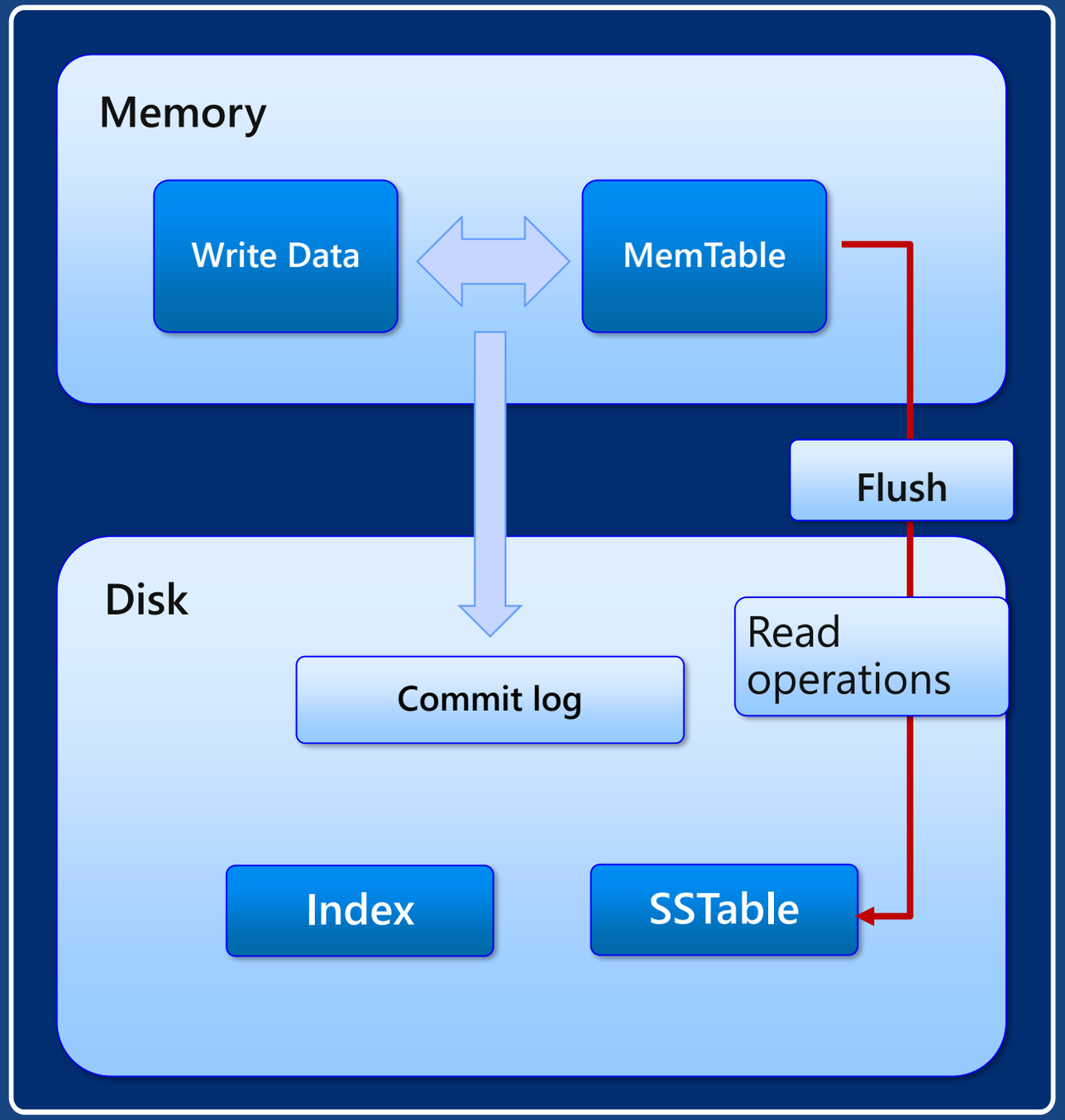
Apache Cassandra

- Consists of a log-structured merge tree
  - A disk based data structure that defers and batches index changes
  - Reduces disk arm movement
  - Favors inserts over reads
  - Good for history of events, log files



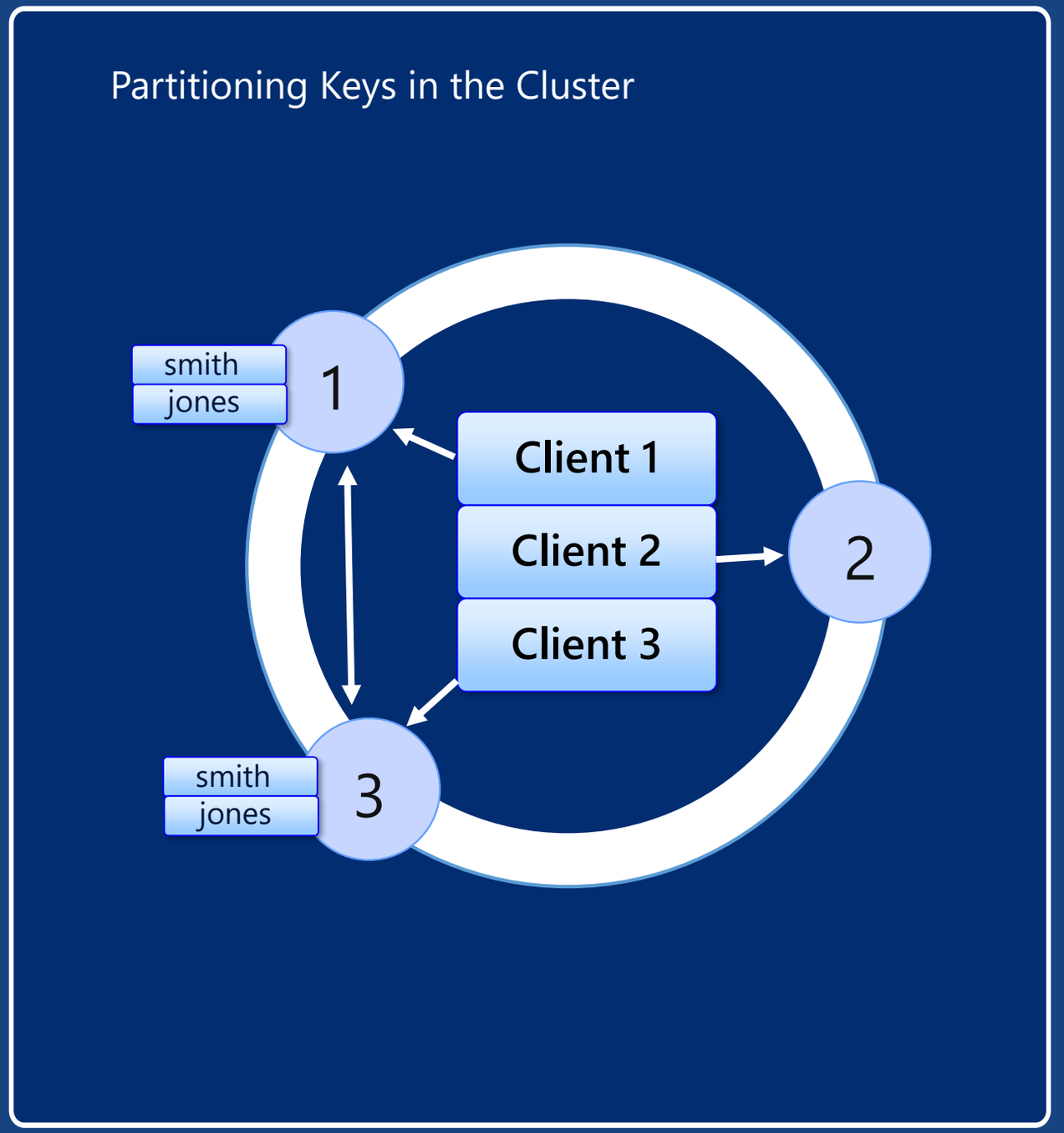
Apache Cassandra

- Write operations are sent to a persistent commit log
- Also writes to a write-back cache called a memtable
- Data overflow of memtables gets flushed to a sorted string table on disk
- The sorted string tables get merged and compacted at specified intervals



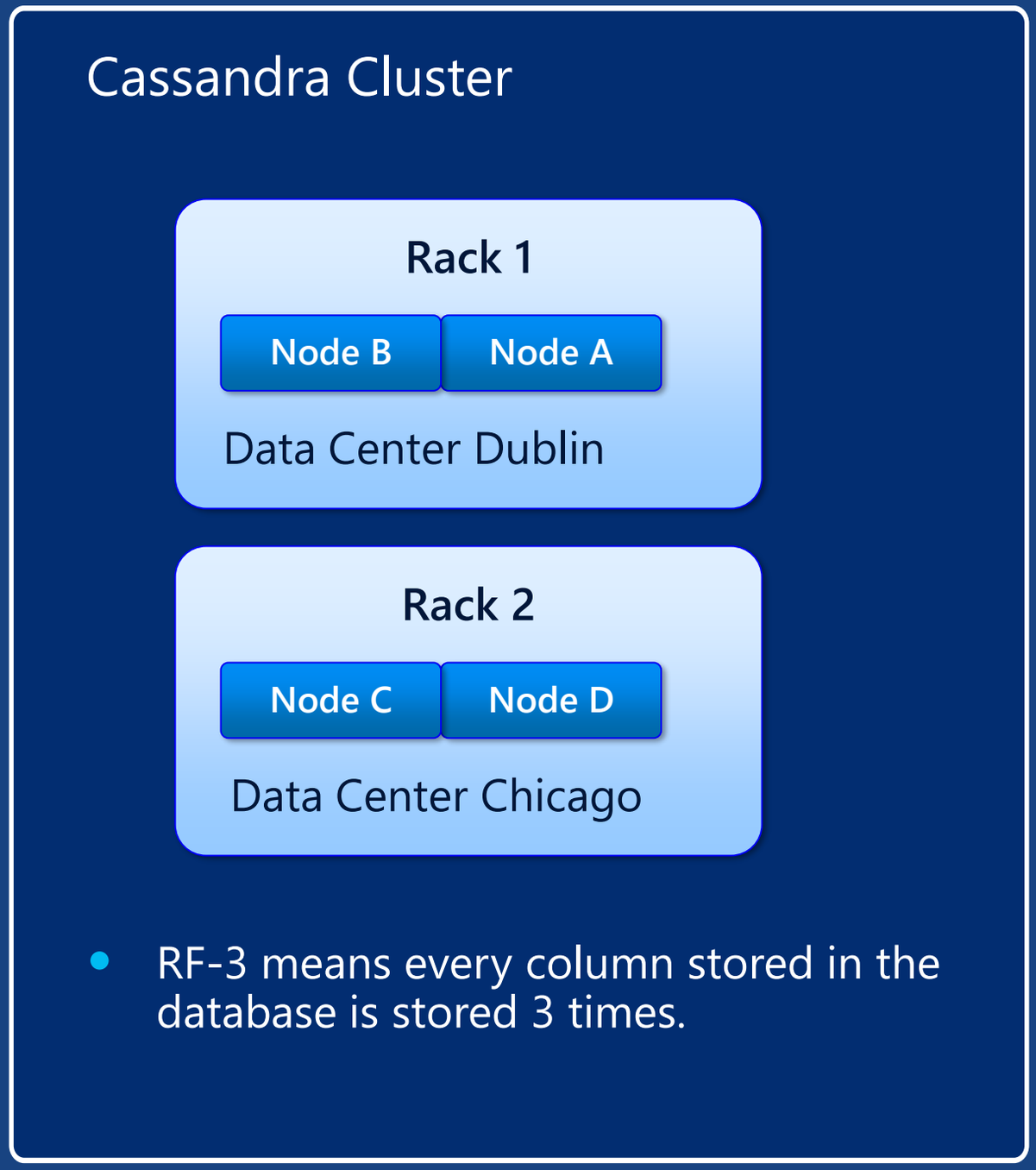
Apache Cassandra

- Append-only sequential writes for performance
- Data place randomly
- Periodically, the SSTable files are merged and compacted
- Ring network topology distributes data evenly
- Redundant writes to multiple nodes



Apache Cassandra

- Has rack awareness for intelligent replica placement
- Support for automatic rebalance to avoid overutilization
- Node rebalance efficient because of limited data movement
- Allows you to think of your data as a range of database rows
- Automatically tracks unreachable replica nodes



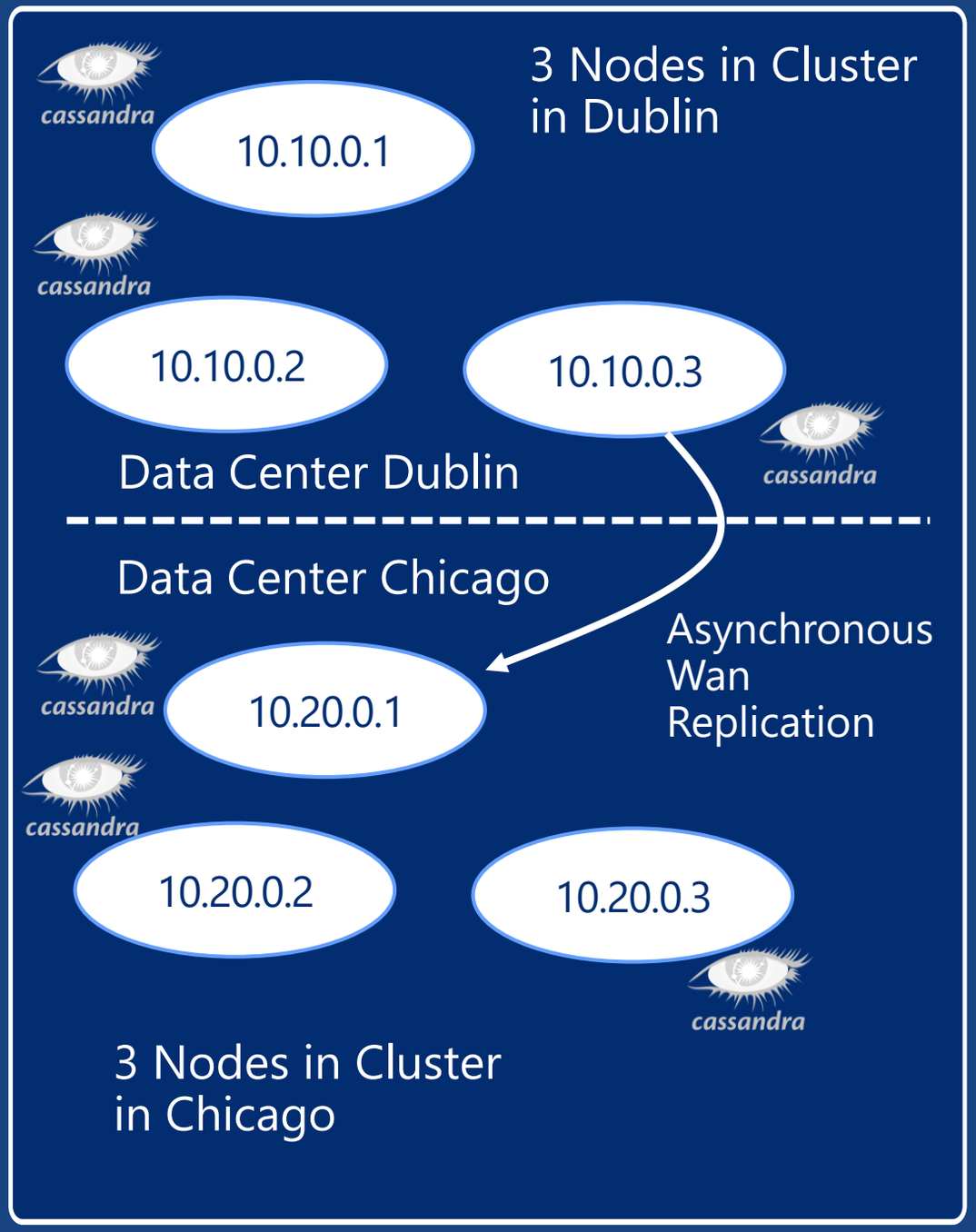
Apache Cassandra

Background Replication

- Issues background reads across replica nodes to ensure data consistency
- Will update replicas that are out of sync with each other

Supports eventually consistent data model

- Offers an eventually consistent data model, sacrificing deterministic behavior for performance critical multiclient scenarios
- This means replicas are not updated simultaneously in a transactional manner



Apache Cassandra

- Database engine **tunable** - level of consistency adjustable based on willingness to sacrafice read and write performance
- Confirmation of successful write operations can be delayed
- The most conservative setting is enforcing that all write operations to all replicas are complete before allowing clients to perform read operations
- A more balanced approach is to leverage the Quorum level of consistency whereby successful writes are acknowledgeable of a majority of replica nodes confirm write operations

11 Levels of consistency

- ALL
- EACH_QUORUM
- QUORUM
- LOCAL_QUORUM
- ONE
- TWO
- THREE
- LOCAL_ONE
- ANY
- SERIAL
- LOCAL_SERIAL

ALL	(1) A write must be written to the commit log and memtable on all replica nodes in the cluster for that partition. (2) Provides the highest consistency and the lowest availability of any other level.
ANY	(1) A write must be written to at least one node. Provides low latency and a guarantee that a write never fails. (2) Delivers the lowest consistency and highest availability.

Apache Cassandra

- Supports a powerful query language (CQL) that includes SELECT, INSERT, UPDATE, and DELETE statements
- But no support for join operation and the referential integrity of traditional DBMS systems
- Supports common data types like integers, floats and doubles, blobs, and more
- Allows you to assign heavier work loads to nodes that are running on more powerful hardware

Version 1.0 CQL Keywords

USE	DROP
SELECT	BATCH
UPDATE	CREATE KEYSPACE
DELETE	CREATE COLUMNFAMILY
TRUNCATE	CREATE INDEX

Connecting to Cassandra

Cassandra Ports

- 7000 for cluster communication
- 7001 if SSL is enabled
- 9160 for Thrift clients
- 9042 for native protocol clients
- 7199 for JMX

- The Thrift interface is a legacy API for older clients.
- JMX
  - Java Management Extensions
  - For managing and monitoring Java applications and services

INSTALLING CASSANDRA

Apache Cassandra


```
$  cd /tmp
$  wget http://apache.mesi.com.ar/cassandra/2.1.9/apache-cassandra-2.1.9-bin.tar.gz
$  mkdir -p ~/cassandra
$  cd ~/cassandra
$  tar -xvf /tmp/apache-cassandra-2.1.9-bin.tar.gz
```

- Download Cassandra with wget
- Untar contents

Includes (1) the core server (2) the nodetool administration command-line interface (3) a development shell (cqlsh and the old cassandra-cli)

Apache Cassandra


```
$  vim ~/cassandra/apache-cassandra-2.1.9/conf/cassandra.yaml
$  vim ~/.bashrc
```

```
446 rpc_address: 0.0.0.0
360 # TCP port, for commands and data
361 # For security reasons, you should not
362 # expose this port to the internet.
363 # Firewall it if needed.
364 # used to be 7000
365 storage_port: 8000
366 ssl_storage_port: 8001
```

cassandra.yaml

- Opening connectivity with rpc_address
- storage_port needed to be changed because of conflict

```
119 #
120 # fix up environment variables and path
121 #
122 export M2_HOME="/usr/bin/apache-maven-3.3.3"
123 export M2=$M2_HOME/bin
124
125 export CASSANDRA_HOME=~/cassandra/apache-cassandra-2.1.9
126 export PATH="$M2:$CASSANDRA_HOME/bin:$PATH"
```

.bashrc

- Setting CASSANDRA_HOME and the PATH

Apache Cassandra


```
$  azure network nsg rule create -g oreilly-rg -a oreillybigdata -n cassandra-rule1
-c Allow -p '*' -r Inbound -y 100 -f Internet -o '*' -e '*' -u '*'
```

- Adds a rule to network security

Resource Group	oreilly-rg
Network Security Group	oreillybigdata
Network security rule	Cassandra-rule1
What the rule does	Allows incoming/outgoing TCP/UDP traffic on VM

Apache Cassandra


```
<dependency>
        <groupId>com.datastax.cassandra</groupId>
        <artifactId>cassandra-driver-core</artifactId>
        <version>2.1.8</version>
</dependency>
```

- The POM.xml entry for Cassandra
- http://mavenrepository.com/artifact/com.datastax.cassandra/cassandra-driver-core/2.1.8

Apache Cassandra


```
// Adding the import statement

import com.datastax.driver.core.*;
public static void main( String[] args )
{
        try    {
               // Connect to Cassandra
                connect("40.78.31.35");
                // Create keyspace and table
               createSchema();
               // Load table with data
               loadData();
               // Query table using "where" clause
               querySchema();
               // cleanup and close
               close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
}
```

Apache Cassandra


```
// connect to connect to Cassandra running in Azure
public static void connect(String node) {
    cluster = Cluster.builder().addContactPoint(node).build();
    Metadata metadata = cluster.getMetadata();
    System.out.printf("Connected to cluster: %s\n",
    metadata.getClusterName());
    for ( Host host : metadata.getAllHosts() ) {
        System.out.printf("Datacenter: %s; Host: %s; Rack: %s\n",
        host.getDatacenter(), host.getAddress(), host.getRack());
    }
    session = cluster.connect();
}
```

Apache Cassandra


```
// Create the keyspace and tables
public static void createSchema() {
       session.execute("CREATE KEYSPACE IF NOT EXISTS simplex WITH replication " +
        "= {'session':'SimpleStrategy', 'replication_factor':3};");
       session.execute(
                     "CREATE TABLE IF NOT EXISTS simplex.songs (" +
                     "id uuid PRIMARY KEY," +
                     "title text," +
                     "album text," +
                     "artist text," +
                     "tags set<text>," +
                     "data blob" +
                     ");");
  …
}
```

Apache Cassandra


```
// load the data
public static void loadData() {

    session.execute(
                  "INSERT INTO simplex.songs (id, title, album, artist, tags)  +
                  "VALUES (" +
                  "756716f7-2e54-4715-9f00-91dcbea6cf50," +
                  "'La Petite Tonkinoise'," +
                  "'Bye Bye Blackbird'," +
                  "'Joséphine Baker'," +
                  "{'jazz', '2013'})" +
                  ";");
…
}
```

Apache Cassandra


```
// query the data
public static void querySchema() {
        ResultSet results = session.execute("SELECT * FROM simplex.playlists "
                                +
                                "WHERE id = 2cc9ccb7-6221-4ccb-8387-f22b6a1b354d;");
                                System.out.println(String.format("%-30s\t%-20s\t%-20s\n%s", "title"
                                "album", "artist",
                                "-------------------------------+-----------------------"
                                + "--------------------"));
        for (Row row : results) {
        System.out.println(String.format("%-30s\t%-20s\t%-20s",
        row.getString("title"),
        row.getString("album"), row.getString("artist")));
        }
        System.out.println();
        }
```