Amazon Web Services for Java Developers

Using the Cloud to Make Enterprise Java Easier

Abstract

Amazon Web Services (AWS) is an ideal platform to develop on and to use for hosting enterprise Java applications. The zero up-front costs and virtually infinite scalability of resources enable Java EE developers to start small and be confident that their infrastructure will grow with their application. In addition, the nature of AWS and the services available help solve some of the problems Java developers often face in more-traditional environments. In this session, you will be introduced to AWS concepts, gain an understanding of how existing Java EE applications can be migrated to the AWS environment, what advantages there are in doing that, and how to architect a new Java EE application from the ground up to leverage the AWS environment for maximum benefit.

Speaker Bio

Craig S. Dickson is a software engineering professional with over 15 years of experience. He has proven leadership experience in both domestic and multi-national start-up and Fortune 500 corporations in the United States, Australia and Europe.

Craig specializes in enterprise Java development and cloud architecture and holds multiple certifications including Sun Certified Architect for JavaEE and Certified ScrumMaster. Craig brings specific expertise in enterprise software architecture and design, refining development processes and building development teams around Agile software engineering principles.

Educated in Australia, Craig holds a BSc(Hons) in Computer Science. He is based in Huntington Beach, CA, and Brisbane, Australia.

Presentation Outline

- AWS Basics
- Core AWS Services for Java Developers
- DevOps with AWS
- Advanced AWS Services for Java Developers
- Making the argument for AWS
- Questions

Topic

AWS Basics

What is AWS?

- Cloud-based infrastructure as a platform (IaaS)
- More than just hardware virtualization
- Hardware *and* Services
- Web-based Management Console
- REST API

Regions, Availability Zones (AZ), Edge Locations

What Do You Pay For?

- Pay only for what you use, no contracts
- Some services are completely FREE!
- Most services have a FREE usage tier
- Some services charge by the hour
- Other services charge for performance
- Other services charge by the amount of data transferred
- Some charge for all 3

Topic

Core AWS Services for Java Developers

Basic Elements of a JavaEE Application

Architectural Element	Related AWS Service
Application Server	Elastic Compute Cloud (EC2) and Elastic Block Store (EBS)
Web Server	Elastic Compute Cloud (EC2) and Elastic Block Store (EBS)
Relational Database	Relational Database Service (RDS)
NOSQL Database	SimpleDB or DynamoDB
Monitoring	CloudWatch
Load Balancing	Elastic Load Balancing (ELB)
Scaling	AutoScaling
File Based Storage	Simple Storage Service (S3) or Elastic Block Store (EBS)

Application & Web Servers

- Use the Elastic Compute Cloud (EC2) service
- Pick a server size to meet your immediate needs
- Run whatever you want, however you want
- Use an Amazon Machine Image (AMI) to save time
- Use Elastic Block Store (EBS)
- Manage servers from browser, or from SDK

NGINX™

APACHE HTTP SERVER

JBoss™

Relational Database

- Relational Database Service (RDS)
- Specialized RDBMS hosting
- Supports MySQL, Oracle, SQL Server
- Automated backups
- Live replication for fail-over support
- No difference for JDBC based apps
- Provision DBs from your browser or SDK

MySQL®

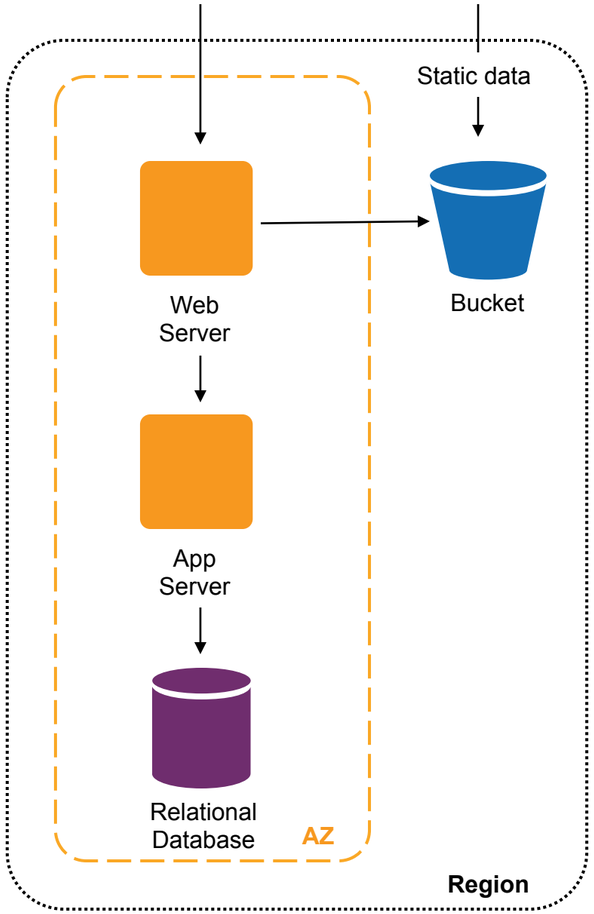
ORACLE®

Microsoft® SQL Server®

File Based Storage

- Simple Storage Service (S3)
  - Cloud based file storage
  - Simple API to CRUD files
  - Unlimited capacity
  - Excellent for static web content
  - Move BLOBS from RDBMS to S3
- Elastic Block Store (EBS)
  - Cloud-based hard drive

Basic N-Tier Architecture



NOSQL Database

- SimpleDB
  - excellent for smaller amounts of structured data
  - 10GB limit per table
  - use in combination with S3
- DynamoDB
  - hosted on SSD
  - no size or request limitations
  - pay for performance

Monitoring - Hardware AND Software

- CloudWatch
- Scalable unified resource monitoring
- Monitor EC2, RDS and other AWS resources
- Also monitor your own application metrics
- Create your own alarm conditions
- Flexible notification system via Simple Notification Service (SNS)

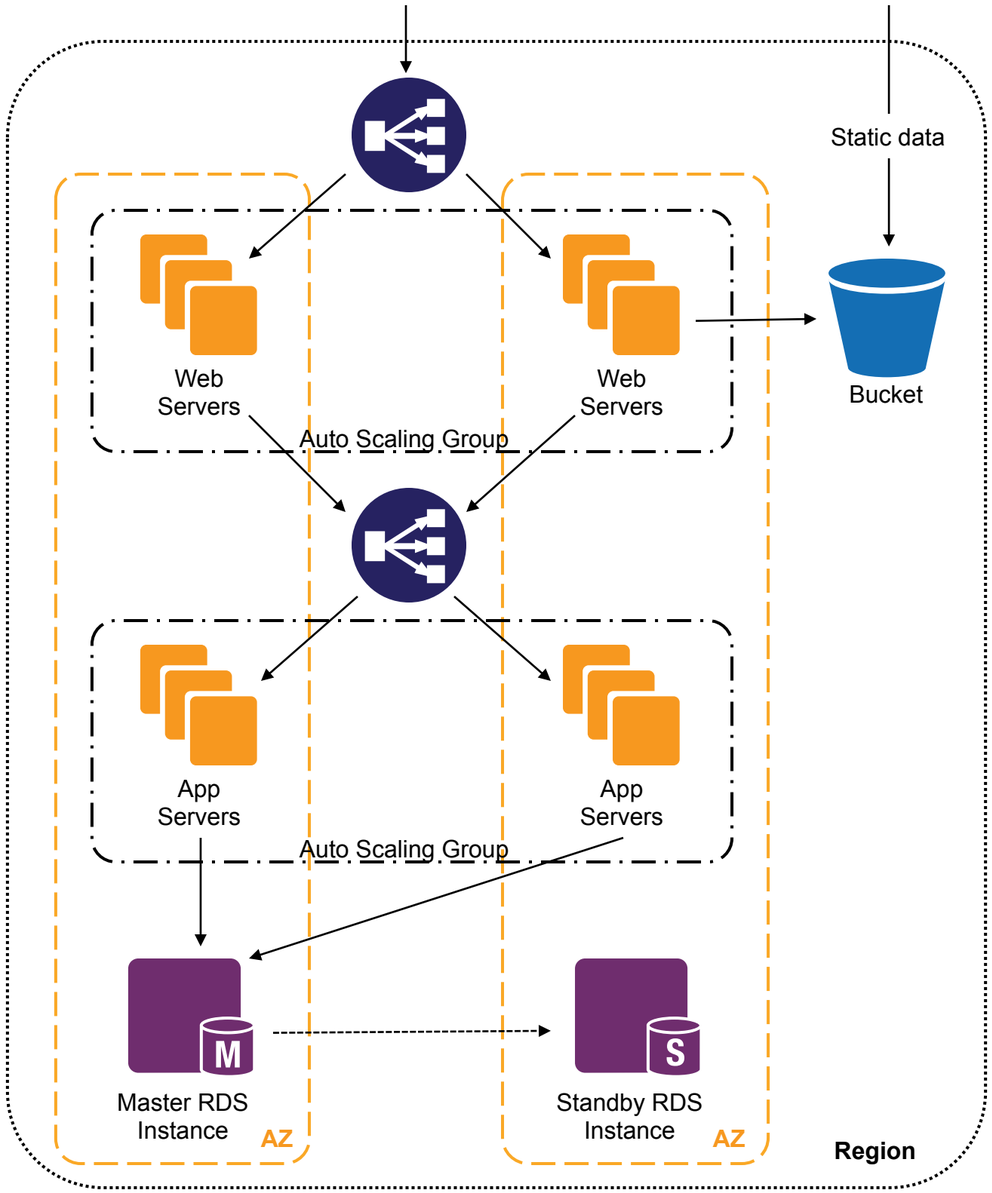
Load Balancing

- Elastic Load Balancing (ELB)
- Scaled fault-tolerant load balancing
- Works with Auto Scaling and CloudWatch
- Load balances requests over a set of EC2 instances
- Can monitor health of EC2 instances and shutdown non-performant instances
- Can trigger scale-up and scale-down events

Auto Scaling

- Works with CloudWatch and ELB
- Automatically provision additional EC2 instances when load increases
- Shutdown instances when load decreases
- Pre-emptive scaling
- Monitor health of EC2 instances

Multi-AZ N-Tier Architecture



Topics

DevOps with AWS

Java and AWS

- Command Line Clients
- AWS Java SDK
  - provides full API to manage AWS services from Java
  - also to use services from within your Java application
- Eclipse integration
- Maven integration

+

AWS at Development Time

- Self-serve hardware
- Guarantee everyone’s environment is the same
- Avoid the expense of redundant hardware
- Work with more than one development setup
- No more sacred demo environments
- Continuous Integration opportunities

AWS at Testing Time

- Self-serve hardware for testers
- Guarantee test environments are identical
- Make it easy to start each test run with a clean baseline
- Avoid redundant resources allocated to the test team
- Can run performance tests without hurting others
- Make QA environment available globally

AWS at Production Time

- Easily migrate from staging to production
- Easily handle unexpectedly high traffic
- Don’t waste hardware for unexpectedly low traffic
- Monitoring built in, no need to purchase and deploy additional monitoring solutions
- DOA servers with faulty hardware, or bad config. files can easily be thrown away and replaced

Topic

Advanced AWS Services for Java Developers

Platform As A Service (PaaS) w/ Elastic Beanstalk

- Bundles up services provided by EC2, S3, CloudWatch, Auto Scaling and ELB
- Browser based provisioning of production ready environments and Java Web Apps
- Often requires no modification to existing code
- Supports live hot-swapping for app upgrades
- Can use other AWS services from apps
- Can be extended through low-level system access

Messaging w/ SNS and SQS

- Simple Notification Service (SNS)
  - publish-subscribe model, comparable to JMS Topics
  - notifications via HTTP, SMTP, SMS
- Simple Queue Service (SQS)
  - queue model (FIFO), comparable to JMS Queues
  - publishers and consumers can be inside or outside of AWS
- Scalable fault tolerant messaging
- Using SNS and SQS as part of a Java application can realize many desirable application characteristics

Transactional Email w/ Simple Email Service

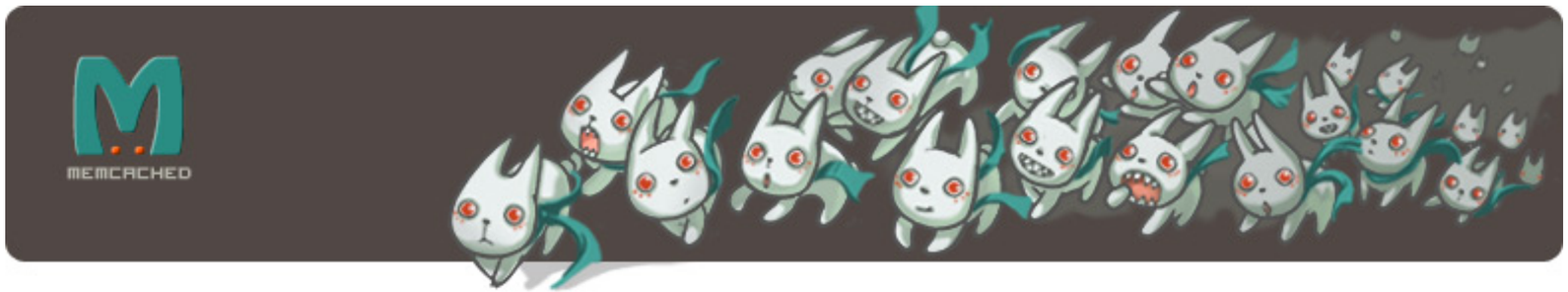
- Send email through API or SMTP
- SES analyzes messages for *deliverability*
- Ability to send increases with reputation
- Sandbox environment for testing
- Not for scatter-gun marketing (ie. SPAM)
- Feedback mechanism for bounces and complaints

Business Process Management w/ SWF

- Simple Workflow Service (SWF)
- Coordinates synchronous and asynchronous work in distributed applications
- Comparable to tools like JBPM
- Supports automated and manual (i.e. human) tasks
- API allows for definition and execution of workflows
- Like SNS and SQS, using SWF as part of a JavaEE application can help building *web scale* applications

Caching w/ ElastiCache

- Fast in-memory data caching
- Compatible with Memcached
- Simply pick a node size to meet your needs
- Health monitoring for nodes in cache cluster
- Easily scale cache size up or down using Console or SDK



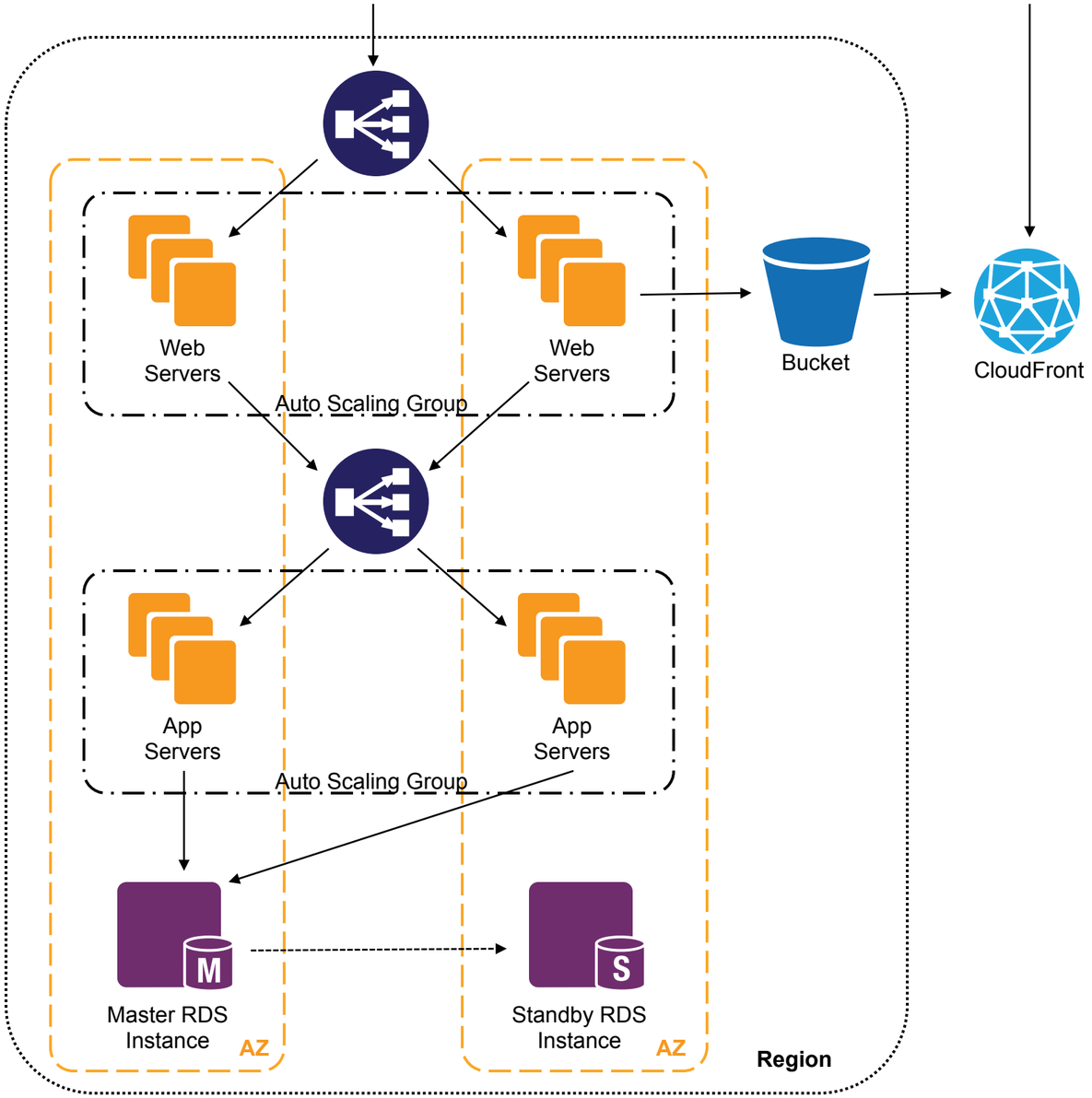
Big Data w/ Elastic Map Reduce

- “Big Data”
- MapReduce framework
- Hosted Apache Hadoop environment
- Uses EC2, S3 and DynamoDB services
- Specify the number and type of EC2 instances used
- Full control via Java API

Content Delivery Network (CDN) w/ CloudFront

- Serve static content from Edge locations
- Clients automatically routed to closest Edge location
- Origin server can be on AWS, or outside AWS
- Support for streaming content

N-Tier Architecture w/ CDN



Other AWS Services

- Route 53
- CloudSearch
- Glacier
- Virtual Private Cloud (VPC)
- Direct Connect
- Flexible Payment Service (FPS)
- DevPay
- Import / Export
- Storage Gateway
- Mechanical Turk
- Alexa Integration

Topic

Making the Argument for AWS

Traditional JavaEE Apps

- Large amounts of resources just to develop
- dev, CI, QA, UAT, performance testing, staging
- web servers, app. servers, databases …
- Not to mention Production
- scalability, reliability, security, resilience to failures, disaster recovery – the ‘ilities
- Many JavaEE apps. fail to meet non-functional goals

How can AWS help?

- Reduce upfront expenses
- Reduce ongoing expenses
- Reduce ongoing 3rd-party software expenses
- JIT infrastructure if things go well (or not)
- Efficiently test multiple scenarios
- Leverage repeatable “web scale” patterns
  - get some ‘ilities for free

Summary

- AWS is cloud based hardware *and* services
- Has excellent support for hosting enterprise Java applications
- Provides opportunities to expand and improve existing applications by integrating to AWS services from within Java applications
- Can help with DevOps issues, makes SysAdmins happy
- AWS can bring significant cost savings to both startups and established companies

Next Steps

- Sign up for an AWS account and experiment
  - http://aws.amazon.com/
- Check out the Java Developer Center
  - http://aws.amazon.com/java/
- Check out AWS related resources on my blog
  - http://craigsdickson.com/tag/aws/

Questions?

Craig S. Dickson

Email - craig@craigsdickson.com

Blog - http://craigsdickson.com

LinkedIn – http://bit.ly/csd-li

Twitter – http://bit.ly/csd-tw