Running Java applications on the Amazon Elastic Compute Cloud

Chris Richardson
Author of POJOs in Action
Founder of Cloud Tools
www.chrisrichardson.net

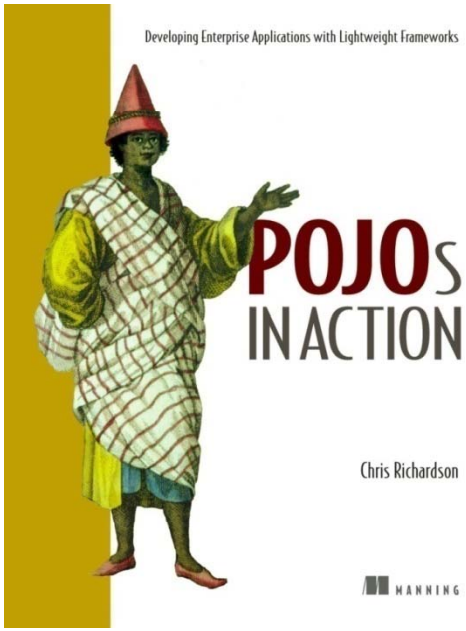
Overall presentation goal

Show how to use

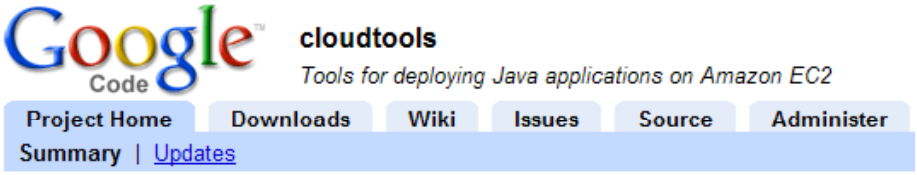
Amazon Elastic Compute Cloud

for developing and deploying Java applications

About Chris



- Grew up in England and live in Oakland, CA
- Over twenty years of software development experience
  - Building object-oriented software since 1986
  - Using Java since 1996
  - Using J2EE since 1999
- Author of POJOs in Action
- Speaker at JavaOne, SpringOne, NFJS, JavaPolis, Spring Experience, etc.
- Chair of the eBIG Java SIG in Oakland (www.ebig.org)
- Run a consulting and training company that helps organizations build better software faster and deploy it on Amazon EC2
- Founder of Cloud Tools, an open-source project for deploying Java applications on Amazon EC2: http://code.google.com/p/cloudtools



Agenda

- **Cloud computing with Amazon EC2**
- Using Amazon EC2
- Overview of Cloud Tools
- Developing on Amazon EC2
- Deploying on Amazon EC2

Computing has come a long way

Past

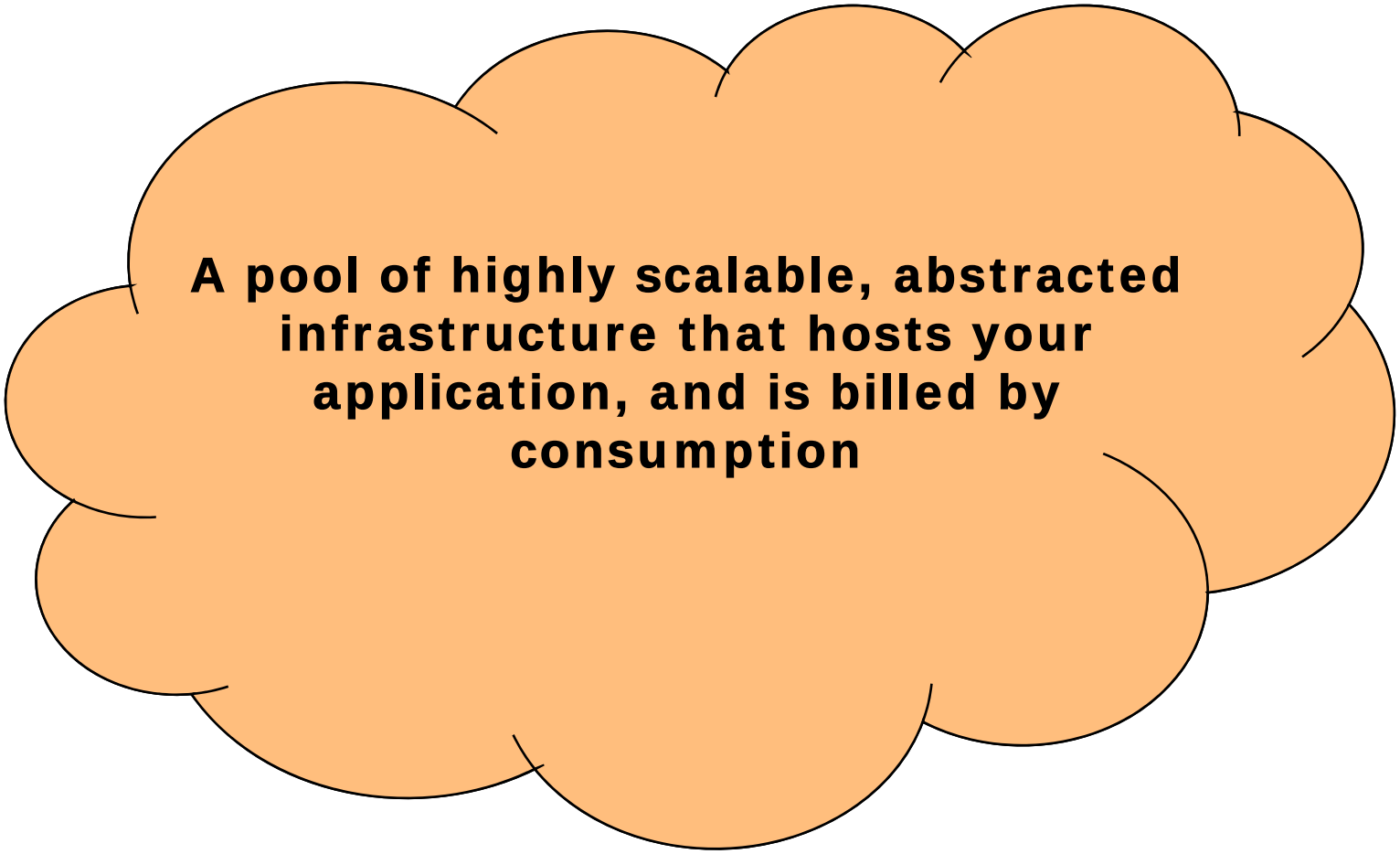
www.computermuseum.org.uk

Present

www.dell.com

Yet we rarely have enough

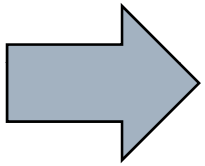
Cloud computing



By James Staten of Forrester Research

Power generation

Past



Present

Amazon-Style Cloud Computing

- Elastic Compute Cloud (EC2)
  - On-demand computing
- Elastic Block Storage (EBS)
  - "SAN on demand"
- Simple Storage Service (S3)
  - Stores blobs of data
- Simple Queue Service (SQS)
  - Hosted queue-based messaging system
- SimpleDB
  - Store data sets
  - Execute queries

Pay per use services managed by Amazon

What is Amazon EC2?

- Virtualized computing environment
- Server instances managed through a web service API
- IP addresses and host names assigned dynamically
- Pay by the hour ($0.10-0.80/hour) + external bandwidth ($0.10-0.18/Gbyte)

```
https://ec2.amazonaws.com/?Action=RunInstances
  &ImageId=ami-398438493
  &MaxCount=3
  &MinCount=3
```

```
<RunInstancesResponse>
...
</RunInstancesResponse>
```

```
cer@arrakis ~
$ ssh ... root@ec2-67-202-41-150.compute-1.amazonaws.com
Last login: Sun Dec 30 18:54:43 2007 from 71.131.29.181
[root@domU-12-31-36-00-38-23:~]
```

Instance types

	Virtual Cores	Compute Units /core*	32/ 64 Bit	Memory	Storage	$/hr
Small	1	1	32 bit	1.7G	160G	0.10
High-CPU Medium	2	2.5	32 bit	1.7G	350G	0.20
Large	2	2	64 bit	7.5G	850G	0.40
Extra Large	4	2	64 bit	15G	1690G	0.80
High-CPU XL	8	2.5	64 bit	7G	1690G	0.80

* EC2 Compute Unit = 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor

Operating systems

- Use Amazon provided Machine Image (AMI)
  - 32-bit Fedora Core 4
  - 64-bit Fedora Core 6
- Many 3rd parties have public AMIs
  - Various Linux distributions
  - E.g. Redhat, RightScale
- Sun provides OpenSolaris
- Windows is private beta
- Build your own Linux:
  - Install applications starting with someone else's AMI and save it
  - Create an AMI from scratch

One minor thing…

Terminate your instance

$\Rightarrow$

your **local** data is lost.

Either very good or very bad

Elastic Block Storage

- Mountable storage volumes
  - "On-demand SAN"
  - Size: 1 GB to 1 TB
  - Mount on a single instance
- Create snapshots
  - Stored in S3
  - Create new volumes from the snapshot
- Cost:
  - $0.10/GByte/month
  - $0.10 per 1 million I/O requests

Elastic IP addresses

- Instance IP addresses are dynamically allocated on start-up
  - Does not work well for publicly accessible services, e.g. a website
- Elastic IP addresses:
  - Statically allocated addresses
  - Associated with your account (max. 5)
  - Attached to an instance (e.g. public facing web server)
  - You configure DNS to resolve to the elastic IP address
- Pricing:
  - Non-attached Elastic IP address - $0.01/hour
  - $0.10 per remap (if > 100 in a month)

Regions and availability zones

- By default, your database master and slave could run on the same physical host!
- Regions:
  - Geographically dispersed locations
  - Currently only one
- Availability zone:
  - Part of a region
  - Engineered to be insulated from failure in other zones
- Specify availability zone when launching instances:
  - Same zone as other instances for free data transfer
  - Different zone for higher-availability

What is the Amazon Simple Storage Service (S3)?

- Flat storage model consisting of buckets and objects
  - Bucket – has a name and contains objects
  - Objects – has a key, stores 1 byte - 5G
  - Object key can look like a path ☺
- Cost:
  - $0.15/GB-Month
  - $0.10-0.18/GB of data transferred
  - $0.00001-$0.000001/Web Service call
  - Data transfers between EC2 and S3 are free of bandwidth charges
- Buckets and objects can be:
  - Public – accessible by anyone
  - Private – accessible to owner, acl member

S3 REST API

```
PUT / HTTP/1.1
Host: <BucketName>.s3.amazonaws.com
Authorization: AWS AWSAccessKeyId:Signature
...
```

Create a bucket

```
PUT /<ObjectName> HTTP/1.1
Host: <BucketName>.s3.amazonaws.com
Authorization: AWS AWSAccessKeyId:Signature
...
...Bytes...
```

Create an item in a bucket

```
GET /<ObjectName> HTTP/1.1
Host: <BucketName>.s3.amazonaws.com
Authorization: AWS AWSAccessKeyId:Signature
...
```

Download an item

```
DELETE /<ObjectName> HTTP/1.1
Host: <BucketName>.s3.amazonaws.com
Authorization: AWS AWSAccessKeyId:Signature
...
```

Delete an item

Using EC2 and S3 together

- AMIs are stored in S3
- EC2 instances use S3:
  - Use REST API
  - Store database snapshots in S3
  - Use 3rd party Linux file system that stores data in S3
  - Store EBS volume snapshots in S3

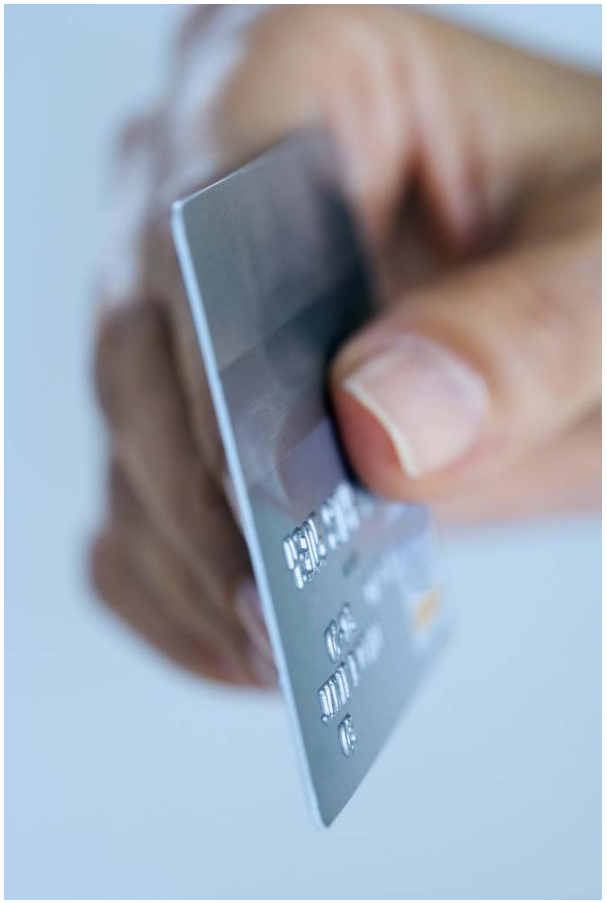
So what does this mean?

- For developers
  - Immediate access to many servers
  - Simplified setup
  - Great for testing
- For deployment
  - Eliminates capital expenses
  - Reduces risk of success catastrophe

Agenda

- Cloud computing with Amazon EC2
- **Using Amazon EC2**
- Overview of Cloud Tools
- Developing on Amazon EC2
- Deploying on Amazon EC2

Signing up for Amazon Web Services



- AWS access identifiers:
  - Account Id
  - Access Id
  - Secret key
  - Private key and certificate
- Only takes a few minutes

EC2 API and Tools

- SOAP and Query APIs
  - Launch and manage instances etc
- Amazon provided CLI tools
  - CLI equivalents of APIs
  - AMI creation tools
- AWS CLI tools from Tim Kay
  - CLI for S3 and EC2
  - Alternatives to Amazon CLI tools
- ElasticFox
  - Awesome Firefox plugin
  - Launch and manage instances

Using the Query API

- https://ec2.amazonaws.com/?queryparameters...
- Mandatory parameters:
  - Action – what to do
  - AWSAccessKeyId – your access id
  - Version – API version
  - Timestamp – when request was made
  - Expires – when it expires
  - Signature – digest of parameters and secret key
  - SignatureVersion – set to 1
- Other parameters depend on Action
- Returns an XML document

Example EC2 requests

Action	Parameters
RunInstances	MinCount, MaxCount, ImageId, InstanceType, ...
TerminateInstances	InstanceId.n
DescribeInstances	InstanceId.n
CreateSecurityGroup	GroupName, GroupDescription
AuthorizeSecurityGroupIngress	GroupName, SourceSecurityGroupName, IpProtocol
DeauthorizeSecurityGroupIngress	...
...	

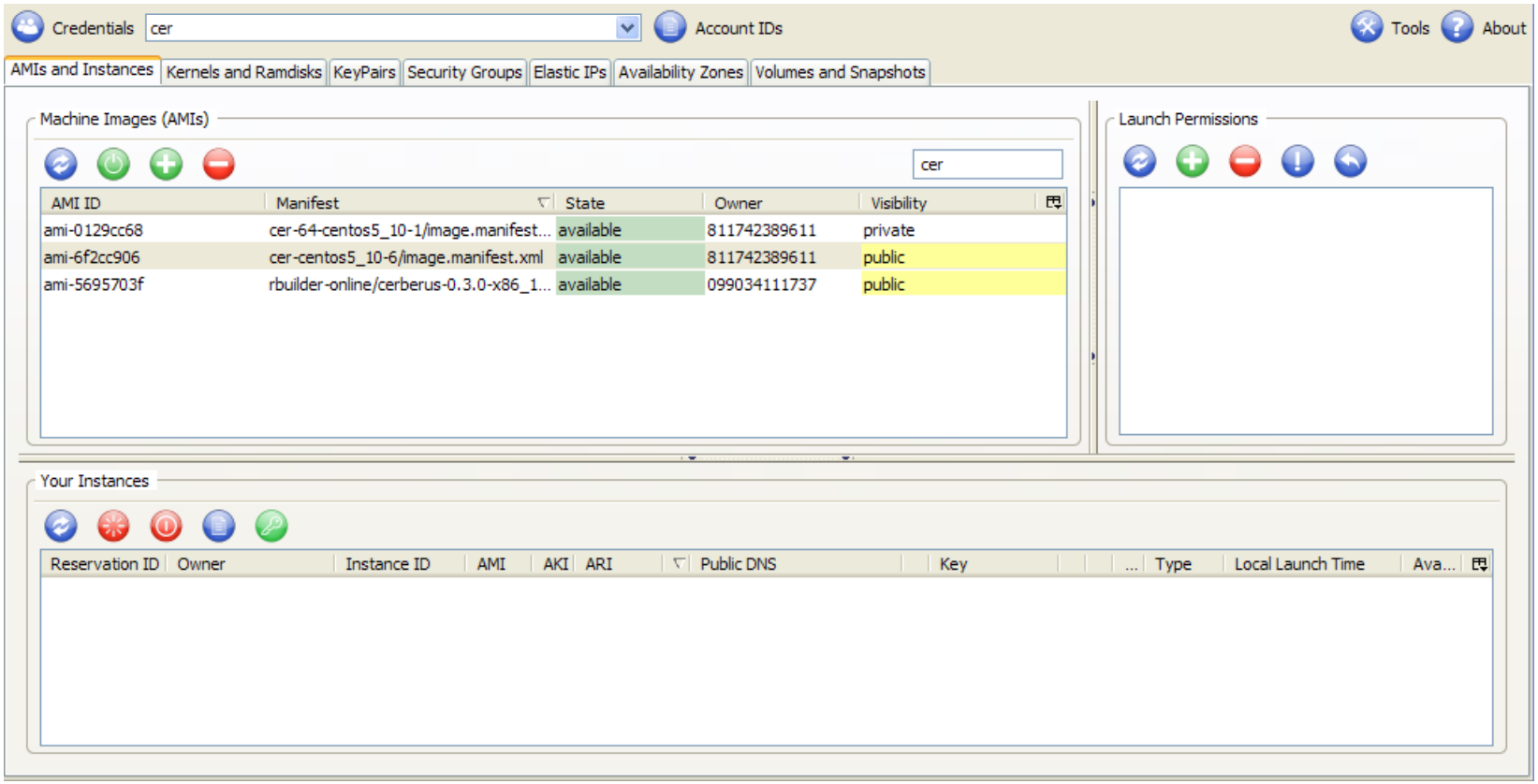
aws - simple access to EC2 and S3

- http://timkay.com/aws/
- Easy to use CLI for EC2 and S3
- Implemented in Perl
- Authenticates using access id and secret key stored in ~/.awssecret

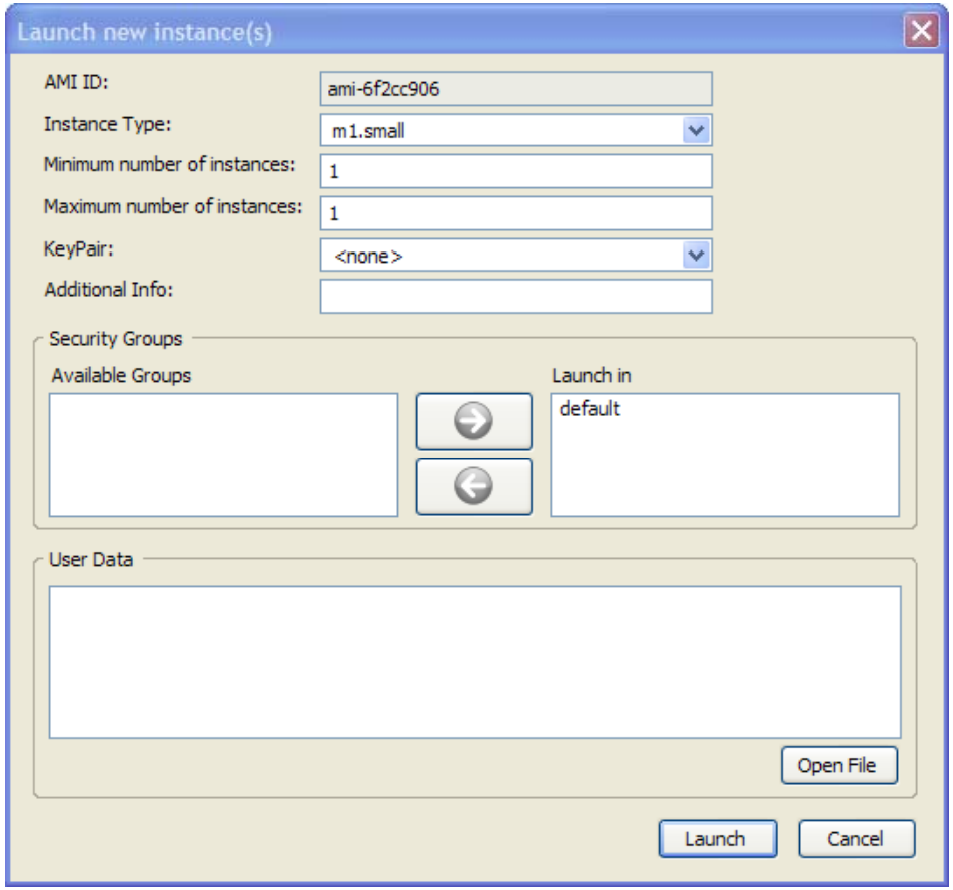
```
$ aws describe-instances
+------------+--------------+------------------------+--
--------------------+
| instanceId |   imageId    |      instanceState      |
   launchTime        |
+------------+--------------+------------------------+--
--------------------+
| i-82728eeb | ami-6f2cc906 | code=48 name=terminated | U
-02-14T01:42:42.000Z |
| i-85728eec | ami-6f2cc906 | code=48 name=terminated | U
```

```
$ aws terminate-instances i-42b24c2b
+------------+-------------------------------+----------------------+
| instanceId |       shutdownState           |    previousState     |
+------------+-------------------------------+----------------------+
| i-42b24c2b | code=32 name=shutting-down | code=16 name=running |
+------------+-------------------------------+----------------------+

cer@arrakis ~
$
```

ElasticFox– Firefox plugin



Launching instances



TIP: launch with a key pair or else you won't have access

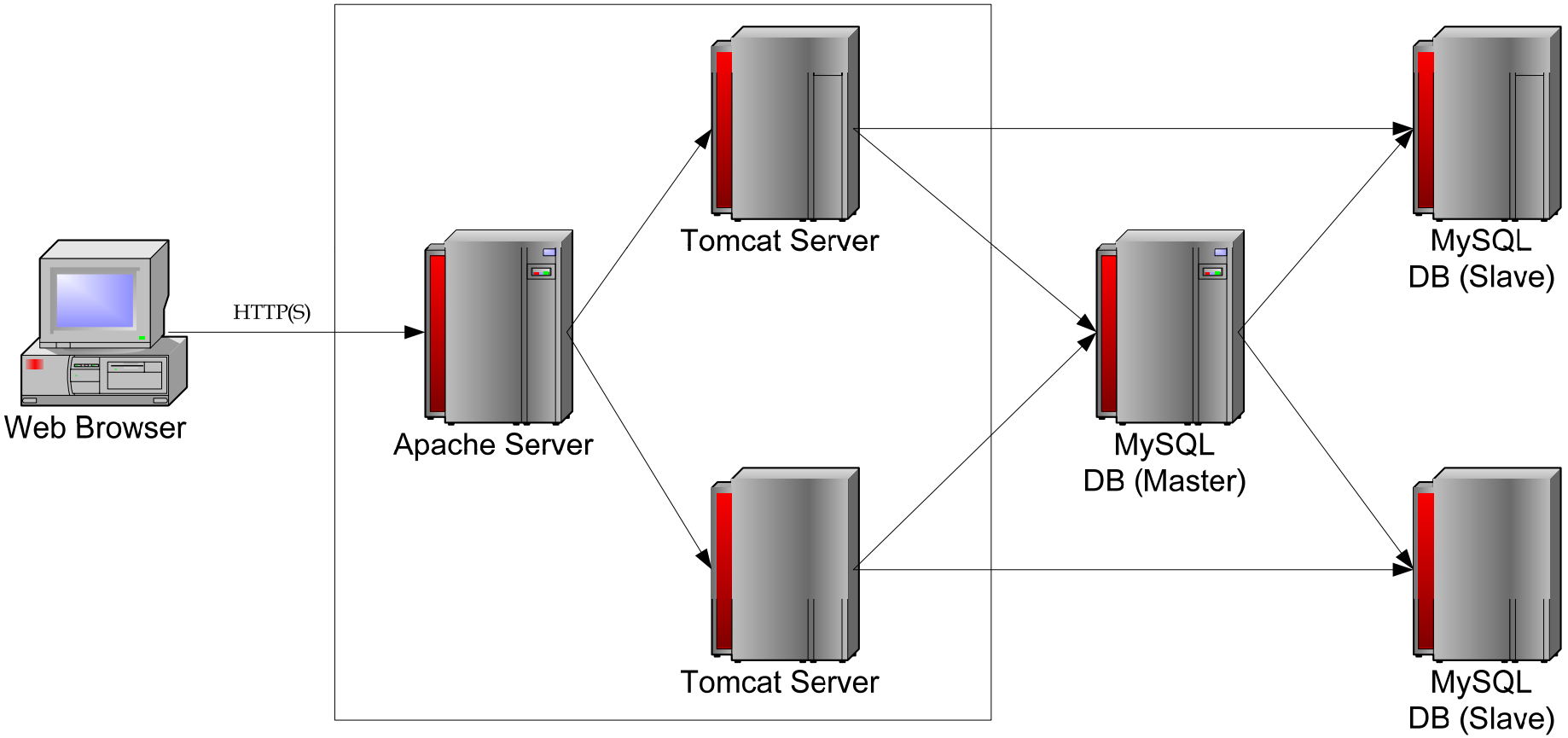
Creating your own image

- Easier: Modify an existing AMI
  - Launch AMI
  - Configure: e.g. yum install ...
- Harder: Build one from scratch
  - Launch AMI
  - Create a file to contain OS installation
  - Mount as a loopback file
  - Install OS: yum –-installroot
- Use AMI tools to bundle and upload to S3

Agenda

- Cloud computing with Amazon EC2
- Using Amazon EC2
- **Overview of Cloud Tools**
- Developing on Amazon EC2
- Deploying on Amazon EC2

Deploying a web application on EC2



Not rocket science but there are many servers to configure and multiple files to upload

What's Cloud Tools?

- Open source project
- 32 and 64 bit AMIs
  - CentOS 5.10
  - Apache/Tomcat/MySQL/JMeter/JetS3t installed
- EC2Deploy framework
  - Launches instances
  - Configures Tomcat, MySQL, Apache
  - Deploys web applications
  - Runs Jmeter tests
  - Written in Groovy
- Maven and Grails plugins
  - Quick and easy deployment to EC2

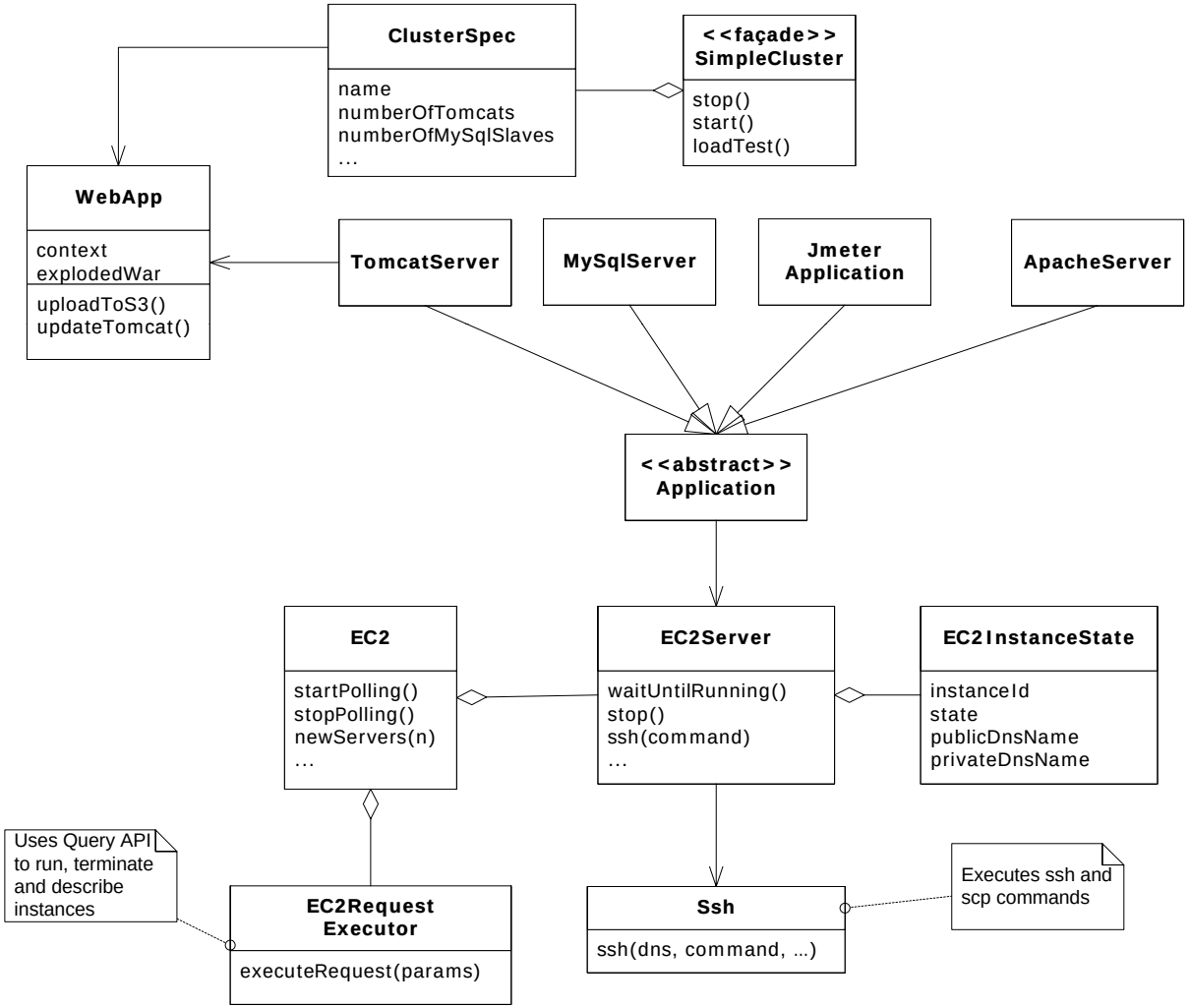
EC2Deploy framework

- Provides a DSL for describing a cluster:
  - Number of Tomcats, MySQL slaves
  - Database scripts
  - Location of web applications
- Launches EC2 instances
- Configures MySQL
- Configures Tomcat and deploys web applications
- Configures Apache to proxy the Tomcat servers
- Runs JMeter tests

Example EC2Deploy Script

```
def ec2 = new EC2(…)

ClusterSpec clusterSpec = new ClusterSpec()
          .tomcats(1)
          .instanceType(EC2InstanceType.SMALL)
          .slaves(1)
          .webApp('/home/cer/…/ptrack', "ptrack")
          .catalinaOptsBuilder({optsBuilder, databaseHost, slaves ->
                    optsBuilder.arg("-Xmx500m")
                    optsBuilder.prop("jdbc.db.server", databaseHost)})
          .schema("ptrack", ["ptrack": "ptrack"],
                    ["src/test/resources/testdml1.sql",
                     "src/test/resources/testdml2.sql"])

SimpleCluster cluster = new SimpleCluster(ec2, clusterSpec)

cluster.start()

cluster.loadTest("/home/cer/…/jmeter/SimpleTest.jmx", [5, 10, 15])

cluster.stop()
```

Domain model



Configuration DSL

```
class ApacheServer extends Application {

 def configure() {
        writeFile fileName: "$apacheConfDir/cluster.conf",
             templateName: "/templates/cluster.conf",
             templateArgs:  [tomcats: tomcats]
        exec "$apacheBinDir/apachectl restart"
        waitForHttp port:80, path: tomcats[0].contexts[0]
  }
…
}
```

```
class MySqlServer extends Application {

 def configureAsMaster() {
        writeFile fileName: "/etc/my.cnf", templateName: "/templates/master.my.cnf"

        restartService "mysqld"

        exec command: "mysql -u root",
            templateName: "/templates/createSchema.sql",
            templateArgs: [schemaSpec: schemaSpec]

        executeSchemaScripts()
    }
}
```

Efficiently uploading web applications, etc.

- Non-durable disks = upload the entire web application
  - 20+ MBs of jars, etc.
  - Takes a long time (over a DSL connection)
- Web application consists of:
  - 90% 3rd party libraries – rarely changing
  - 10% application code and content – only some of it changes
- Use JetS3t to accelerate uploads
  - Incremental upload of exploded web application to S3 bucket
  - Incremental download to Tomcat webapps/ directory
  - First upload is slow but subsequent uploads are fast

Maven Plugin

```
<plugin>
   <groupId>net.chrisrichardson</groupId>
   <artifactId>cloudtools-maven-plugin</artifactId>
   <configuration>
     <schemaName>ptrack</schemaName>
     <schemaUsers>
       <param>ptrack:ptrack</param>
     </schemaUsers>
     <catalinaOptsBuilder>
       {builder, databasePrivateDnsName ->
            builder.arg("-Xmx1000m")
          builder.prop("jdbc.db.server", databasePrivateDnsName)}
       </catalinaOptsBuilder>
     </configuration>
</plugin>
```

Goals:
- deploy
- redeploy
- stop
- dbsave
- dbrestore
- jmeter

```
mvn cloudtools:deploy
```

Grails Plugin

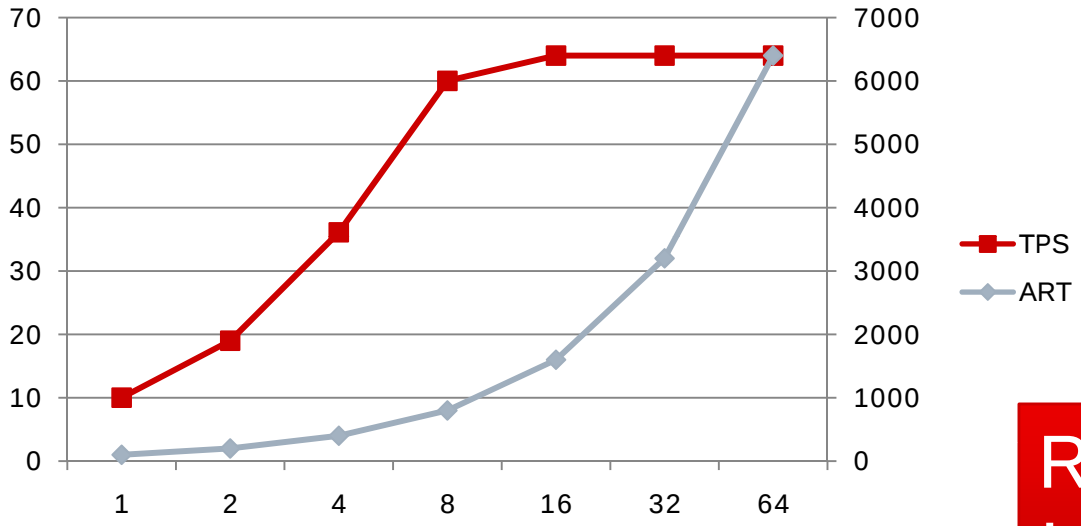
- Packages E2Deploy as a Grails framework plugin
- Deploys a Grails application to EC2

```
$ grails install-plugin <path to plugin>
$ grails cloud-tools-deploy
```

Agenda

- Cloud computing with Amazon EC2
- Using Amazon EC2
- Overview of Cloud Tools
- **Developing on Amazon EC2**
- Deploying on Amazon EC2

Collecting performance metrics



Requires hardware
Time consuming

Measure transactions/second (TPS), average response time (ART), utilization, etc.

Multiple test runs with different loads, number of servers, etc.

Load testing with Cloud Tools

- Runs JMeter with specified number of threads
- Collects machine utilization stats
- Generates reports
- Executes multiple test runs simultaneously

```
<performanceReport>
  <cpus>1</cpus>
  <threads>10</threads>
  <host>
    <name>database</name>
    <cpuUtil>3.2757014224403784</cpuUtil>
  </host>
  <host>
    <name>tomcat0</name>
    <cpuUtil>94.32473318917411</cpuUtil>
  </host>
  <host>
    <name>apache</name>
    <cpuUtil>0.12280614752518504</cpuUtil>
  </host>
  <host>
    <name>jmeter</name>
    <cpuUtil>7.033683910704496</cpuUtil>
  </host>
  …
  <duration>557.943</duration>
  <tps>10.753786677133686</tps>
  <art>916.6578333333</art>
</performanceReport>
```

```
mvn cloudtools:jmeter -Dcloudtools.thread.count=1,4,8
```

Other kinds of testing

- Testing failover
  - Launch cluster
  - Take down servers
  - Test recovery scripts, e.g. MySQL slave->master
- Testing database upgrades
  - Launch cluster
  - Install snapshot of production data
  - Apply database migration script
  - Verify that it works

Functional testing

- Tests can be slow, e.g.
  - Web tests
  - Database intensive tests
- Run tests in parallel on EC2
  - Multiple test drivers, app servers, DBs
  - Relatively cheap: >$75/hour developer vs. $0.10/hour machine
- Selenium Grid from Thoughtworks
  - Open Source framework
  - Runs Selenium web tests in parallel on EC2
- Stay tuned for more general solutions

Building on a fresh machine

- Debug builds that fail because of a missing dependency
  - Maven dependency
  - Manually installed 3rd party library
- Build on a fresh EC2 instance
- Great for open-source projects

Agenda

- Cloud computing with Amazon EC2
- Using Amazon EC2
- Overview of Cloud Tools
- Developing on Amazon EC2
- **Deploying on Amazon EC2**

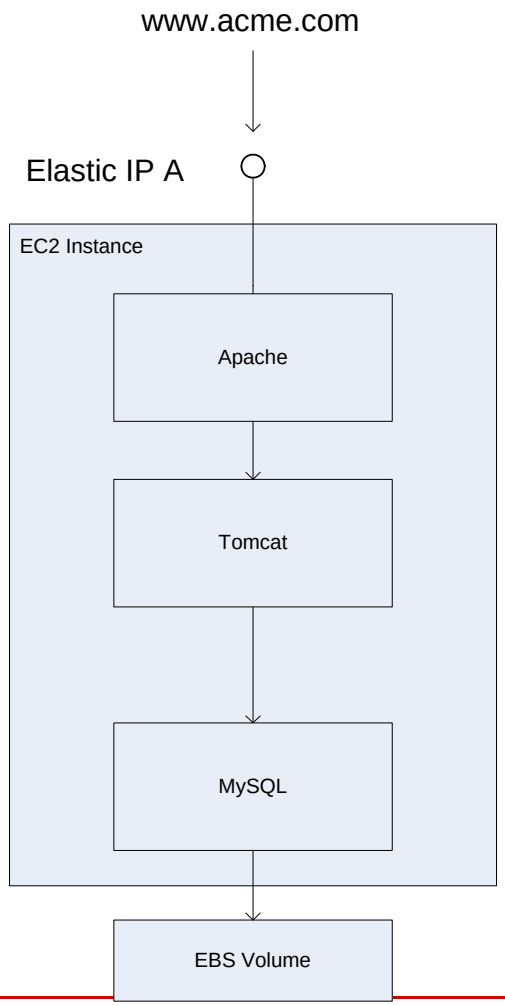
Deploying applications on Amazon EC2

- Great for startups (especially those without a business model)
  - Get up and running ready quickly
  - No upfront hardware costs
  - Scale up/down with load
  - Reduces the risk of a **success catastrophe**
- Great for enterprises
  - No need to wait for corporate IT
  - Use for short-term projects

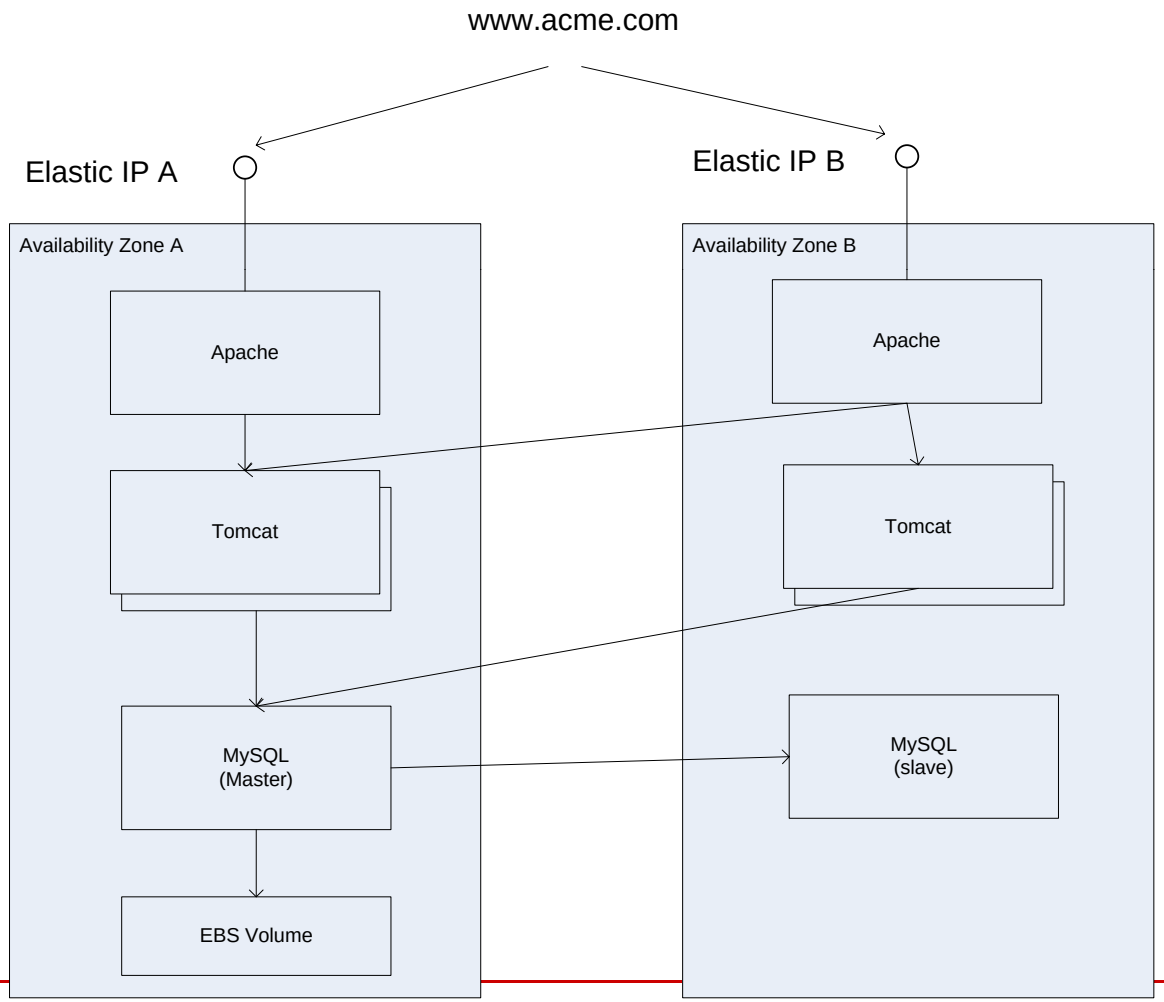
Issues with AWS

- Security:
  - Lack of PCI compliance
  - Discomfort with sending customer data to a 3rd party
- Technology:
  - Not yet suitable for extremely large relational databases
  - Lack of very large machines, e.g. 64G memory
  - Lack of multicast
- Financials:
  - Cost of bandwidth
  - Steady state costs > your own hardware

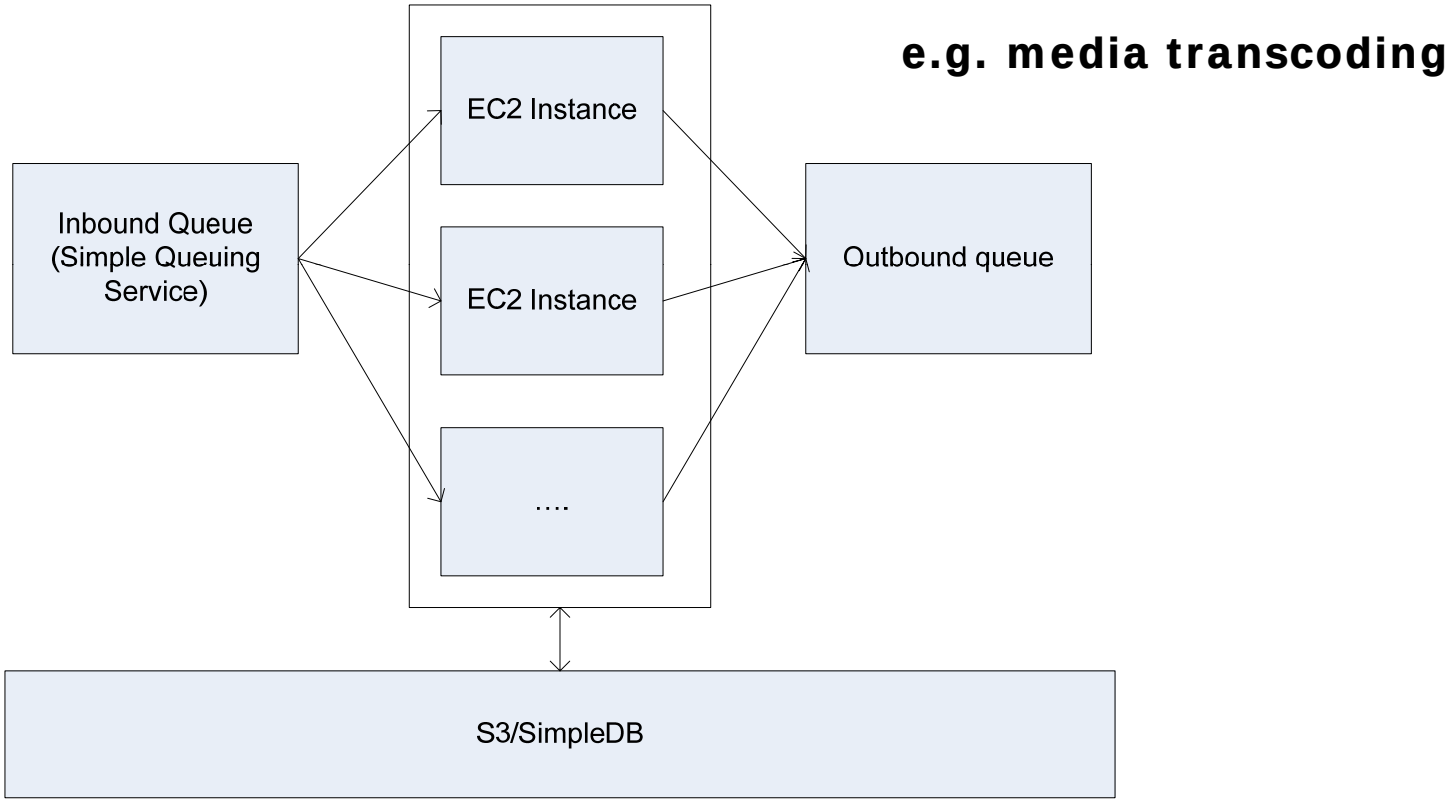
Starter website - $



Highly available - $$



Batch processing architecture



Easy upgrades

- Clone production environment
- Apply upgrades
- Terminate old instances once you are sure that everything works

Cloud bursting

- Host application on your own hardware
- Use AWS for short-term spikes
  - e.g. use EC2 instances with slave DBs to handle read-only requests
- Periodic batch jobs
  - E.g. content rendering/transformation

Using AWS in your application

- Simple Storage Service (S3)
  - Stores blobs of data
  - Eg. Photo sharing website
  - Store media
  - Hand out URLs to S3 objects
- Simple Queue Service (SQS)
  - Hosted queue-based messaging system
  - Alternative to JMS
  - Loosely coupling between systems
- SimpleDB
  - Store data sets
  - Execute queries

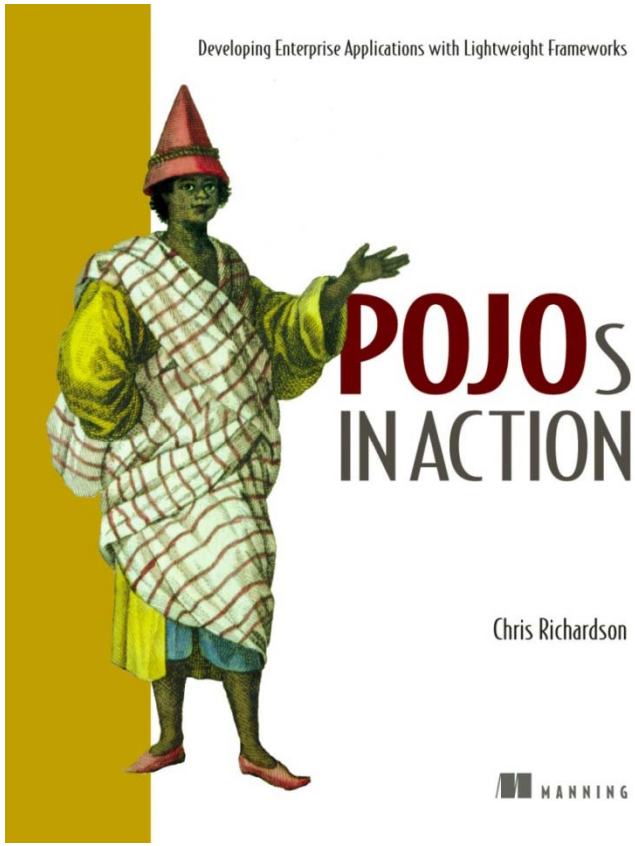
Java libraries for AWS

- Generate the REST/SOAP requests manually
- JetS3t
  - Rich API for accessing S3
  - https://jets3t.dev.java.net/
- Typica
  - API for SQS, EC2, SimpleDB
  - http://code.google.com/p/typica/
- SimpleJPA
  - Subset of JPA on Simple DB
  - http://code.google.com/p/simplejpa/

Summary

- Cloud Computing
  - Immediate access to many servers
  - Pay as you go – no upfront investment/commitment required
  - Easily scale up/down
- Cloud Tools
  - Easy deployment and testing from Maven and Grails
  - Configure multiple clusters
  - Run JMeter tests in parallel

Final thoughts



- Download Cloud Tools today:

http://code.google.com/p/cloudtools

- Buy my book ☺
- Send email:

chris@chrisrichardson.net

- Visit my website:

http://www.chrisrichardson.net

- Talk to me about consulting and training