Developing on a cloud

Chris Richardson

Author of POJOs in Action

Chris Richardson Consulting, Inc

http://www.chrisrichardson.net

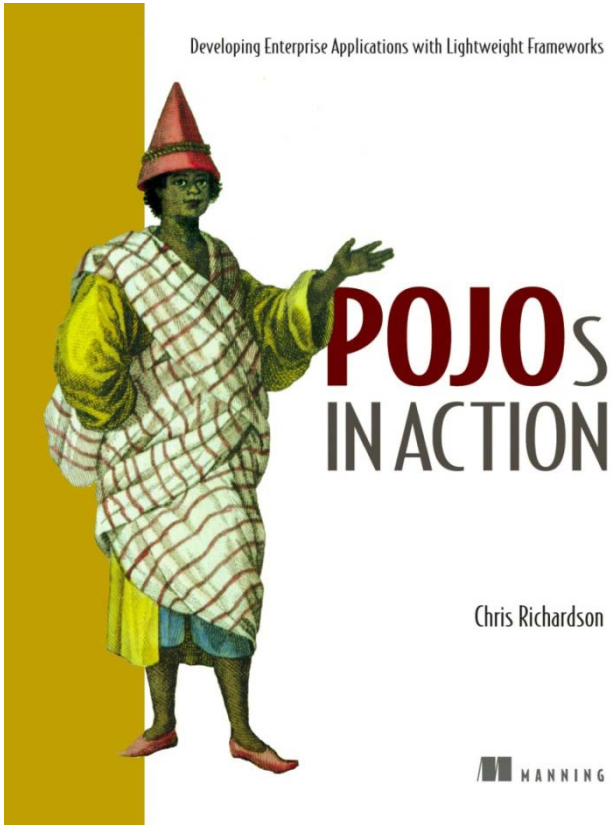
Overall presentation goal

Show how

Amazon's Elastic Compute Cloud improves developer productivity and

reduces hardware costs

About Chris



- Grew up in England
- 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, JavaPolis, NFJS, JUGs, ….
- Chair of the eBIG Java SIG in Oakland (www.ebig.org)
- Run a consulting and training company that helps organizations build better software faster

Agenda

- **Introduction to EC2**
- Using EC2
- Overview of Cloud Tools
- Developing on EC2
- Using EC2 in production
- A few thoughts about Groovy

Hardware has come a long way

Past

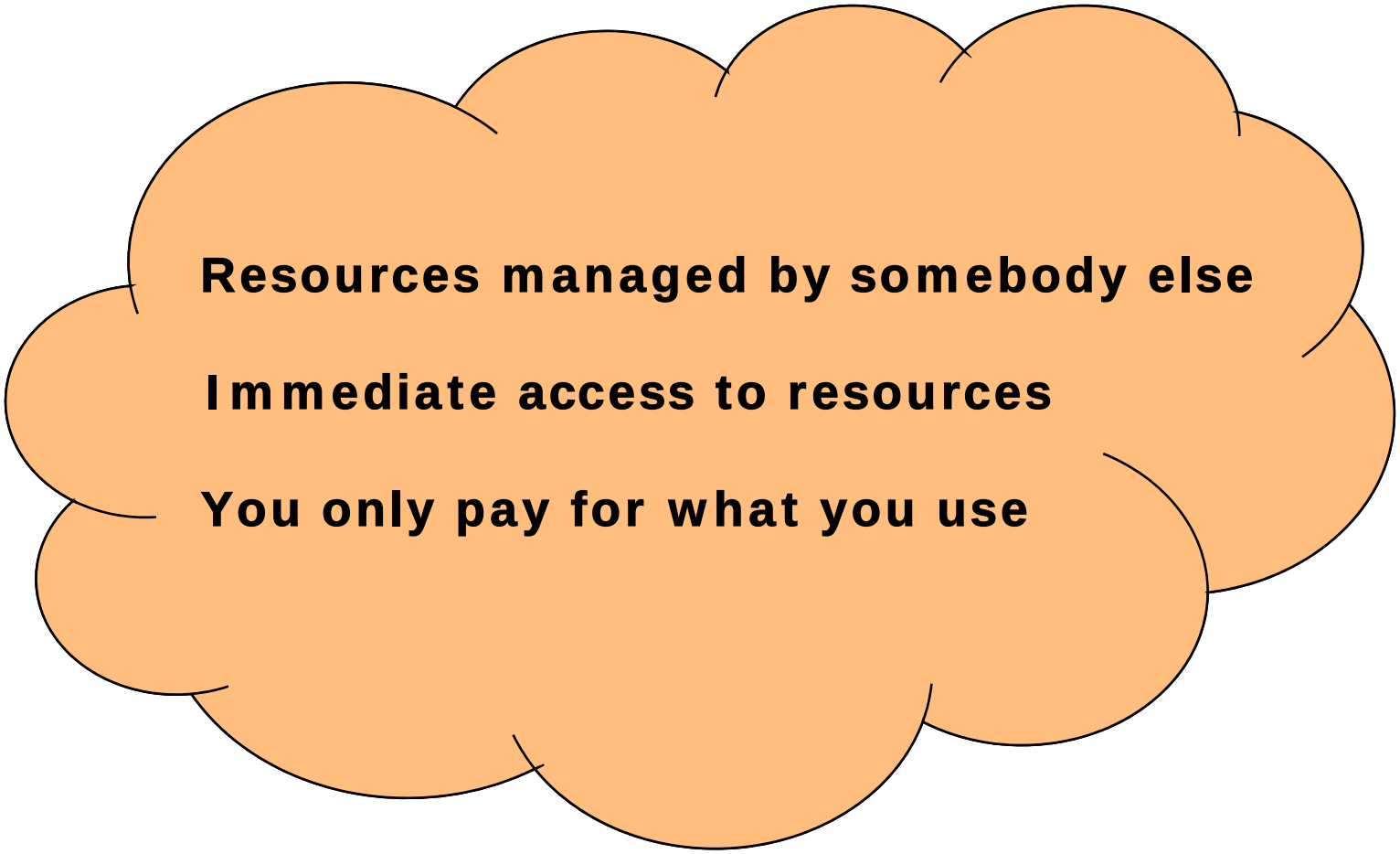
www.computermuseum.org.uk

Present

www.dell.com

- Yet:
  - Production – upfront investment, often under-utilized, procurement delays, …
  - Development /QA – one machine/developer, scavenge for machines for testing, …

Cloud computing



Cloud Computing with Amazon Web Services LLC

- Elastic Compute Cloud (EC2)
  - On-demand computing
- 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's EC2?

- Virtualized computing environment
- Server instances allocated and released through a web service API
- 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
```

```
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:~]
```

Three kinds of instances

	Virtual Cores	Compute Units /core*	32/ 64 Bit	Memory	Storage	$/hr
Small	1	1	32 bit	1.7G	160G	0.10
Large	2	2	64 bit	7.5G	850G	0.40
Extra Large	4	2	64 bit	15G	1690G	0.80

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

Linux Operating System

- 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
- Build your own
  - Install applications starting with someone else's AMI and save it
  - Create an AMI from scratch
- Run Windows via QEMU!?

One minor thing…

Terminate your instance

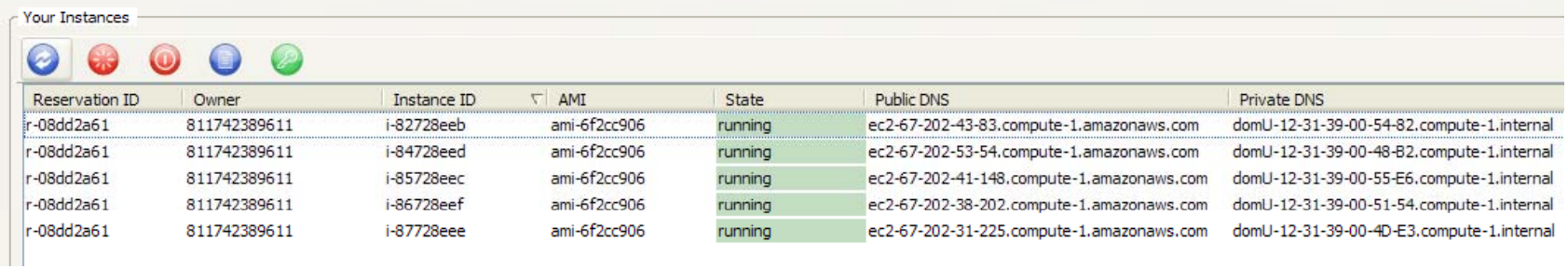
$\Rightarrow$

your data is lost.

Either very good or very bad

EC2 Networking

- IP and DNS assigned to an instance
- Private IP address: reachable with EC2
- Internal DNS name: resolves within EC2 to private IP
- Public IP address: publically accessible
- Public DNS name
  - Publically resolvable to public IP address



Configuring the firewall

- An instance belongs to a security group
- Security group = set of rules granting access
  - Members of a user/group
  - CIDR [IPAddr+mask] ⇒ protocol, ports
- Default group's default settings:
  - Allow members of the default group to access each other
  - Blocks all other traffic
  - You typically grant SSH access to outside world (or at least your machine)

Group Permissions

Protocol	From Port/ICMP Type	Source User:Group	To Port/ICMP Code	Source CIDR
icmp	-1	811742389611:default	-1	
udp	0	811742389611:default	65535	
tcp	0	811742389611:default	65535	
tcp	22		22	0.0.0.0/0

SLAs

(slide intentionally blank)

What is Amazon's 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

RESTful S3 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 can use S3 as storage:
  - Use Restful API
  - Store database snapshots in S3
  - Use 3rd party Linux file system that stores data in S3

So what does this mean for developers?

- Many machines instantly available
  - "Blank" machines
  - Throwaway machines
- Great for testing
  - Functional tests
  - Performance tests
  - Failover tests
  - Duration tests

…for deployment?

- Great for startups (without a business model)
  - Get up and running ready quickly
  - No upfront hardware costs
  - Scale up/down with load
- Easy upgrades
  - Launch a new set of instances
  - Reconfigure load balancer
  - Terminate old instances running only once you are sure that everything works
- S3 is a great way to deliver static content

Agenda

- Introduction to EC2
- **Using EC2**
- Overview of Cloud Tools
- Developing on EC2
- Using EC2 in production
- A few thoughts about Groovy

Using Amazon Web Services

- Two minute sign up:
  - Email/password
  - Credit card
- Get AWS access identifiers:
  - Account Id
  - Access Id
  - Secret key
  - Private key and certificate

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
- EC2UI
  - Awesome Firefox plugin
  - Launch and manage instances

Using the Query API

- https://ec2.amazonaws.com/?querypara meters...
- 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

Example EC2 requests

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

Amazon CLI Tools

- Wrappers around web services
- Java and shell scripts
- Require env vars to be set:
  - EC2_PRIVATE_KEY – path to private key
  - EC2_CERT – path to certificate

```
cer@arrakis /cygdrive/h/cer/personalDocs/technical/ec2
$ ec2-run-instances ami-6f2cc906 -n 1
RESERVATION     r-45c93f2c      811742389611    default
INSTANCE        i-42b24c2b      ami-6f2cc906                    pending         0

cer@arrakis /cygdrive/h/cer/personalDocs/technical/ec2
$
```

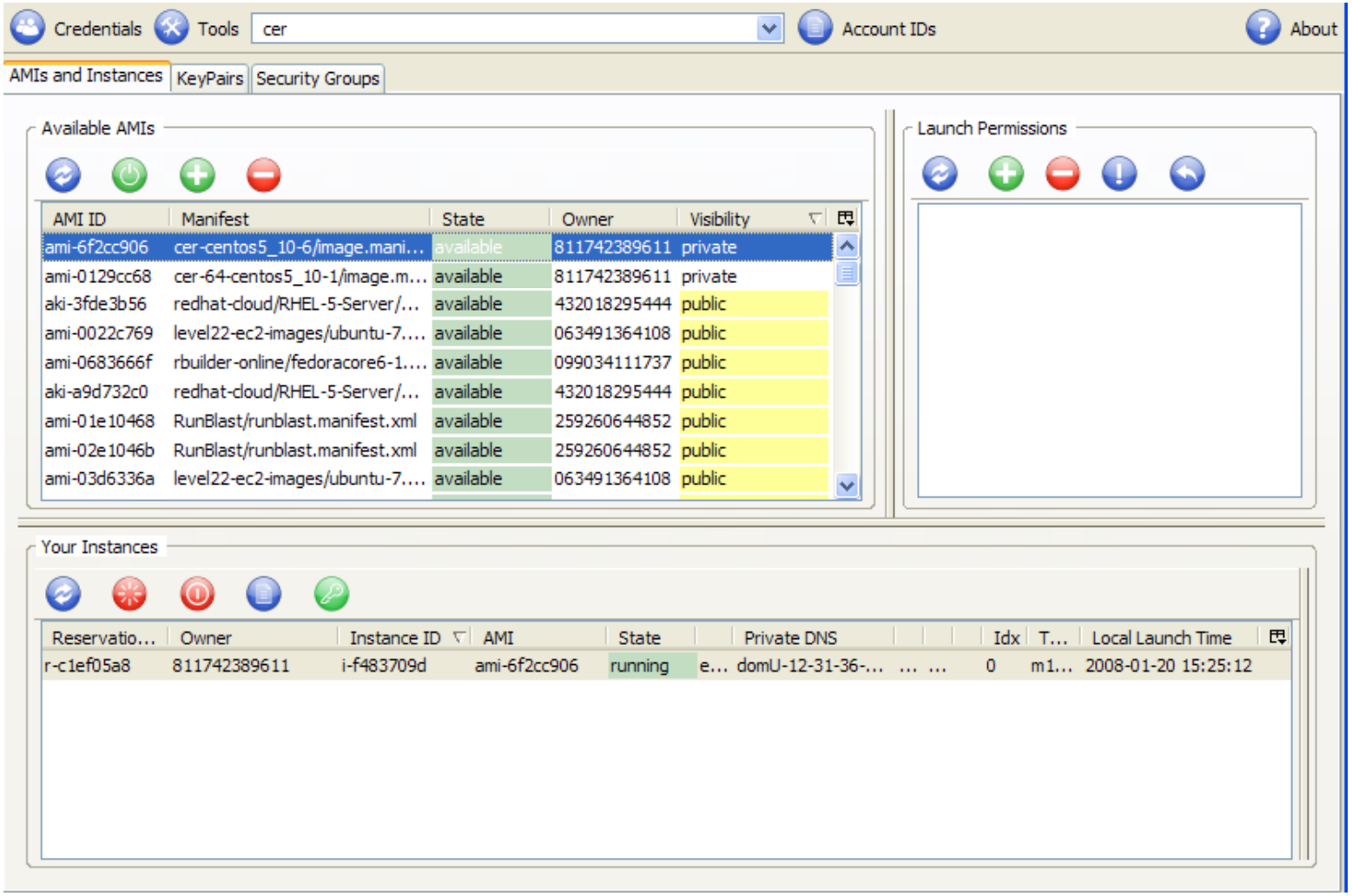
aws - simple access to EC2 and S3

- 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 | 
-02-14T01:42:42.000Z |
| i-85728eec | ami-6f2cc906 | code=48 name=terminated | 
```

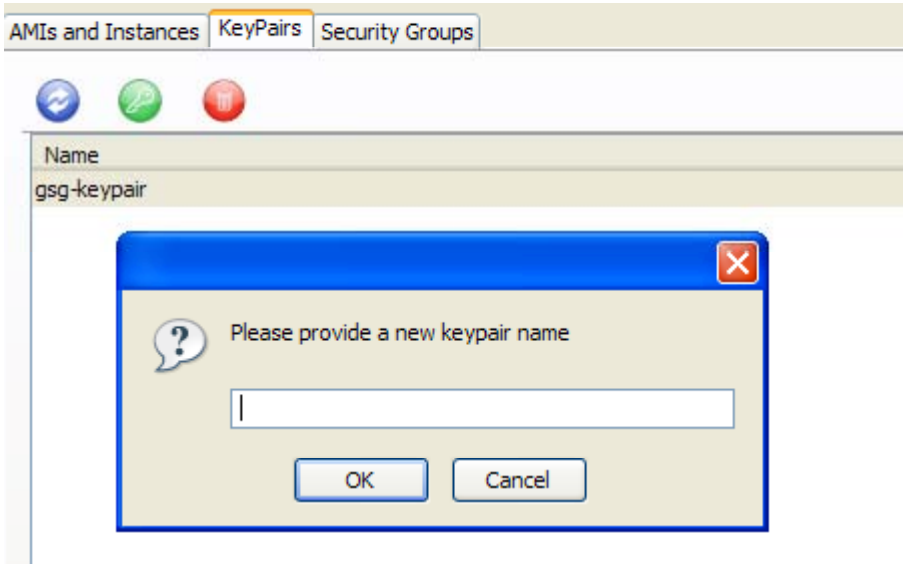
```
$ aws terminate-instances i-42b24c2b
+------------+-----------------------------+----------------------+
| instanceId |       shutdownState         |    previousState     |
+------------+-----------------------------+----------------------+
| i-42b24c2b | code=32 name=shutting-down | code=16 name=running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

cer@arrakis ~
$
```

EC2UI – Firefox plugin

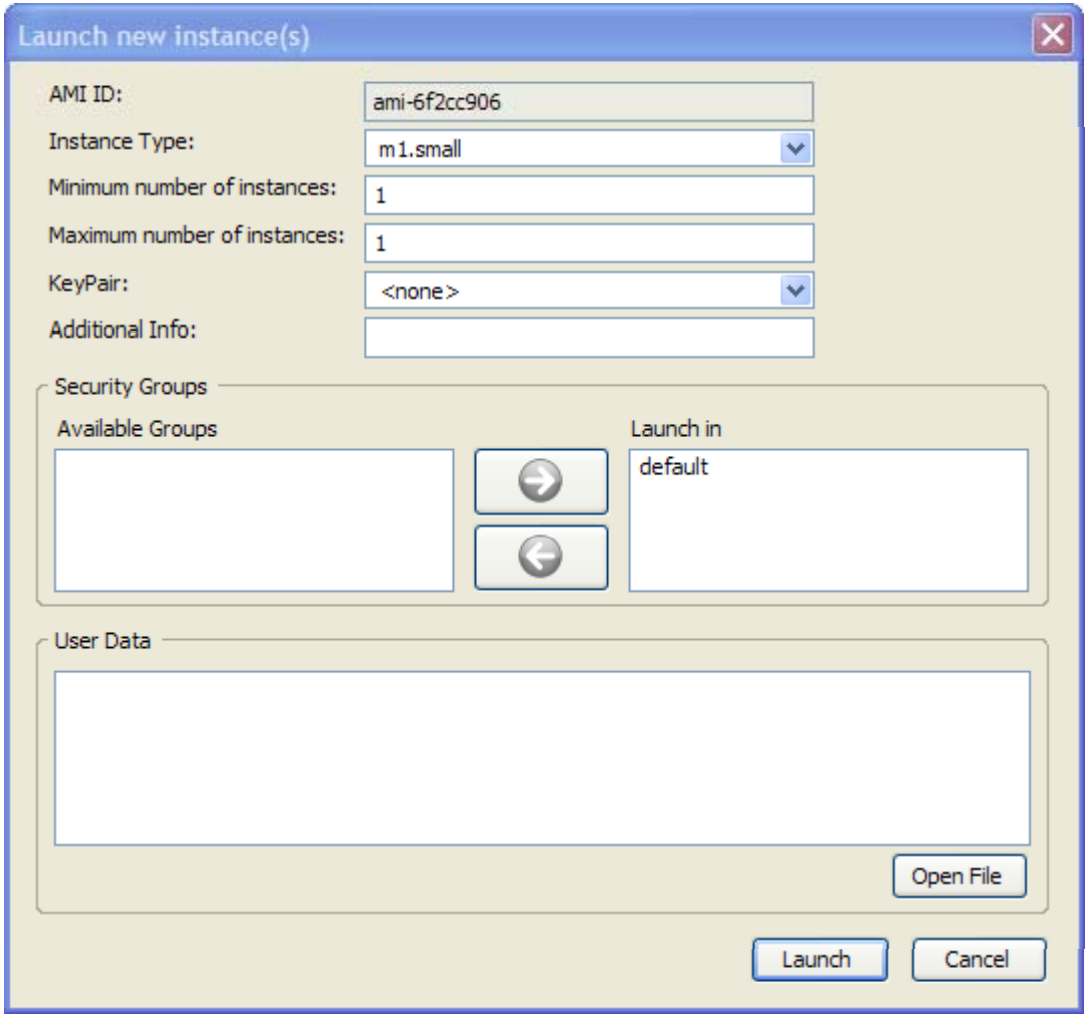


Launching: #1 Creating a key pair

- Creates a named 2048 RSA key pair
- Public key is stored by Amazon
- You store the private key in a file
- Used to:
  - Launch AMI – specify key pair name
  - SSH into instance – with private key



Launching: #2 launching instances



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

Accessing instance via SSH

- Ssh as root with private key file

```
cer@arrakis ~
$ ssh -i id_rsa-gsg-keypair root@ec2-67-202-35-3.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:~]
```

Creating your own image

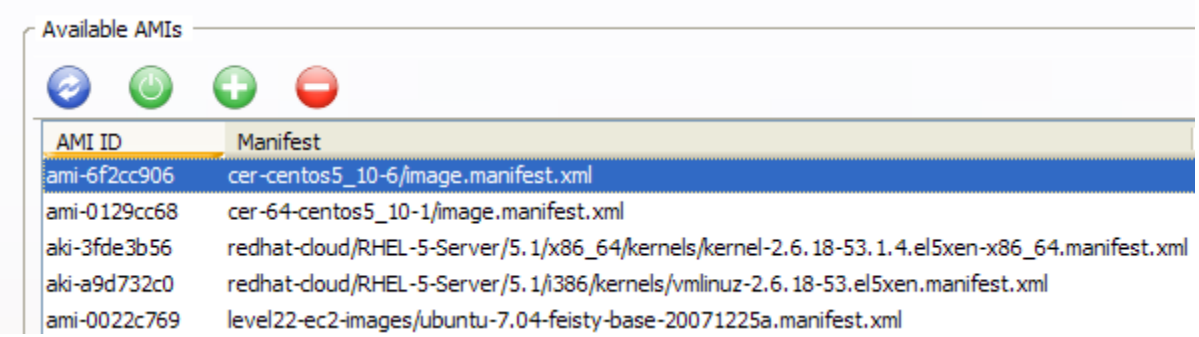
- Easier: Modify an existing AMI
  - Launch AMI
  - Configure: e.g. yum install …
  - Bundle and upload to S3
- Harder: Build one from scratch
  - Launch AMI
  - Create a file to contain OS installation
  - Mount as a loopback file
  - Install OS: yum –-installroot
  - Bundle and upload to S3

AMI tools

- ec2-bundle-vol
  - Compresses, encrypts and signs a snapshot of the root file system
  - Requires: private key, cert, account id
- ec2-bundle-image
  - Creates a bundle from a "loopback" file
  - Requires: private key, cert, account id
- ec2-upload-bundle
  - Upload a bundled AMI to an S3 bucket
  - Requires: access id, secret key

Registering an AMI

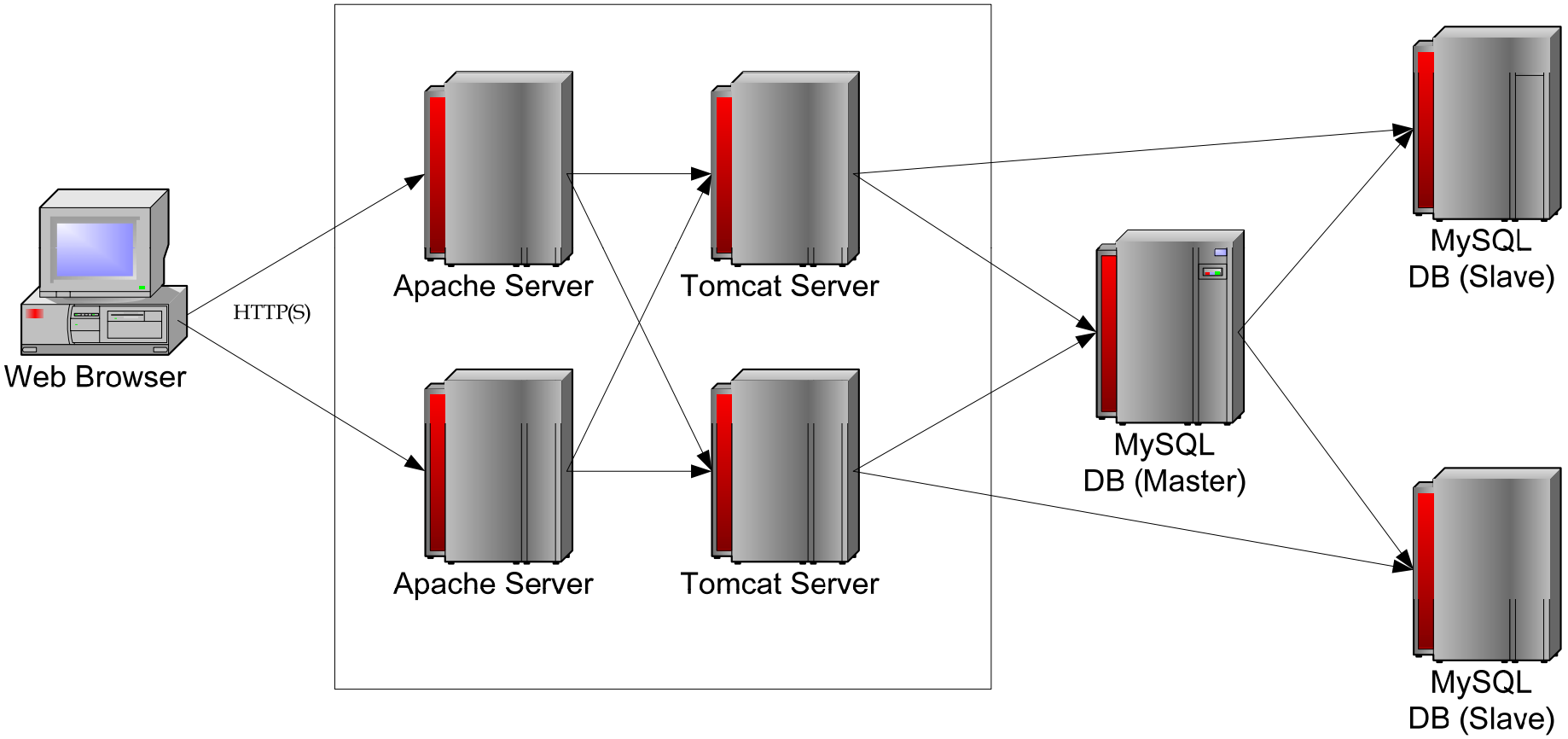
- Registers an AMI and assigns it an AMI id (used for launching)
- Web service or CLI or EC2UI
- S3 path to manifest file, e.g. <bucketName>/image.manifest.xml



Agenda

- Introduction to EC2
- Using EC2
- **Overview of Cloud Tools**
- Developing on EC2
- Deploying applications on EC2
- A few thoughts about Groovy

Deploying an web application on EC2



Not rocket science but you must…

- Create an AMI with the Java software installed
- Configure MySQL(s), Tomcat(s), Apache, Jmeter, … each time on startup
- Upload web applications, SQL scripts etc.

What's Cloud Tools?

- 32 and 64 bit AMIs
  - CENTOS 5.10
  - Tomcat/MySQL/etc installed
- EC2Deploy framework
  - Deploying web applications on EC2
  - Configures Tomcat, MySql, Apache
  - Runs Jmeter tests
  - Written in Groovy
- Maven and Grails plugins
  - Quick and easy deployment to EC2

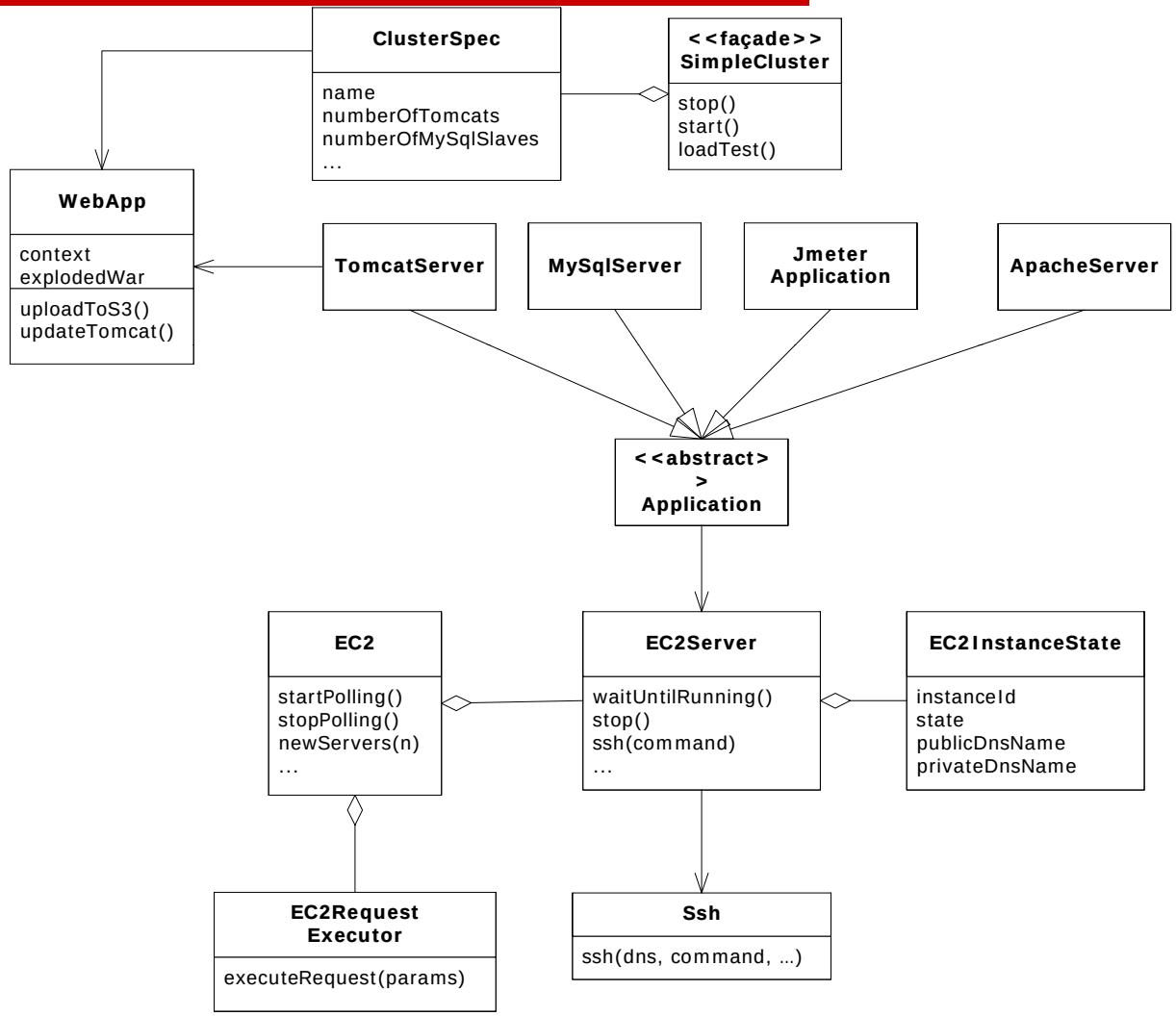
EC2Deploy framework

- Provides an API for launching and managing Tomcat applications on EC2
- You specify:
  - Number of MySQL slaves, Tomcats
  - DB scripts
  - Exploded wars
- EC2Deploy
  - Launches and configures the instances
  - Runs Jmeter tests

Example EC2Deploy Script

```
AWSProperties awsProperties = AWSPropertiesUtil.makeAWSProperties()

def ec2 = new EC2(awsProperties)

def explodedWar = 'h:/cer/code/j2eebook/sptrack/webapp/target/ptrack'

ClusterSpec clusterSpec = new ClusterSpec()
            .tomcats(1)
            .instanceType(EC2InstanceType.SMALL)
            .slaves(1)
            .webApp(explodedWar, "ptrack")
            .catalinaOptsBuilder({builder, databasePrivateDnsName ->
                    builder.arg("-Xmx500m")
                    builder.prop("jdbc.db.server", databasePrivateDnsName)})
            .schema("ptrack", ["ptrack": "ptrack"],
                        ["src/test/resources/testdml1.sql",
                   "src/test/resources/testdml2.sql"])
SimpleCluster cluster = new SimpleCluster(ec2, clusterSpec)

cluster.start(true)

cluster.dumpInfo()

def jmeterTest = "h:/cer/code/j2eebook/sptrack/functionalTests/jmeter/SimpleTest.jmx"

cluster.loadTest(jmeterTest, [1])
```

Domain model



EC2Deploy internals

- Uses EC2 Query API
  - RunInstances
  - DescribeInstances
  - TerminateInstances
- Uses SSH to execute commands
  - Edit configuration files
  - Start and stop services
  - ...
- Uses Jets3t (sync) to minimize uploads
  - WAR directory ⇒ S3 bucket
  - S3 bucket ⇒ Tomcat webapps directory

Maven Plugin

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

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

```
cer@arrakis /cygdrive/h/cer/code/j2eeBook/projecttrack/webapp
$ mvn -Dcloudtools.plugin.aws.properties=h:/cer/personalDocs/technical/ec2/aws.properties cloudtools:deploy
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'cloudtools'.
```

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

- Introduction to EC2
- Using EC2
- Overview of Cloud Tools
- **Developing on EC2**
- Using EC2 in production
- A few thoughts about Groovy

Why use EC2 for Development?

- Easy access to lots of machines
  - Just takes a few minutes
  - Use for as long as you want
  - Terminate when done
  - No need to scavenge for machines
  - No need to maintain expensive infrastructure
- Throwaway servers
  - Break the machine $\Rightarrow$ No problem
  - Just get a new one
- Readily available “blank” machines
  - Non-persistent storage can be a benefit
  - E.g. test a build on a fresh machine

Load testing with EC2Deploy

- Runs Jmeter
- Collects machine utilization stats
- Generates report

```
<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>
```

```
cer@arrakis /cygdrive/h/cer/code/j2eeBook/projecttrack/webapp
$ mvn -Dcloudtools.plugin.aws.properties=h:/cer/personalDocs/technical/ec2/aws.properties cloudtools:jmeter
```

Other kinds of testing

- Testing failover
  - Spin up cluster
  - Take down servers
  - Test recovery scripts, e.g. slave->master
- Testing DB upgrades
  - Spin up cluster
  - Install snapshot of production data
  - Apply DB migration script
  - Test

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
- Stay tuned....☺

Agenda

- Introduction to EC2
- Using EC2
- Overview of Cloud Tools
- Developing on EC2
- **Using EC2 in production**
- A few thoughts about Groovy

Use S3 to host images

- Your application:
  - Stores images etc. on S3
  - Hands out public S3 Urls
- Hide S3 with virtual hosting
  - DNS CNAME: images.acme.com ⇒ images.yoururl.com.s3.amazon.com
  - http:// images.yoururl.com/Foo.jpg ⇒ Bucket = images.yoururl.com, item=Foo.jpg
- Benefits
  - Reduces load
  - Reduces storage costs
- Drawbacks
  - Not a content distribution network

Benefits of running on EC2

- Immediate provisioning of servers
  - No need to wait 5 days or more
- No upfront costs
  - No setup fees or 12 month commitment
- Add/remove servers based on usage
  - Load increases $\Rightarrow$ add servers
  - Site is idle at 2am $\Rightarrow$ remove servers
- Easy upgrades:
  - Launch new set of servers
  - Configure load balancer to start new sessions on them
  - Terminate old servers later

But the disks aren't durable

- Database replication = some safety
- But no control over instance ⇔ hardware
  - Database master and slave might be on the same physical box
  - Hardware failure ⇒ lose both!
- Solutions
  - Backup to S3, e.g. every 10 minutes
  - Replicate to a remote database

EC2 networking issues

- No fancy hardware, e.g. content switches, firewalls:
  - Must use software solutions
  - e.g. HADeploy, IpTables/EC2 provided firewall
- Dynamically assigned IP addresses
  - Need to use dynamic DNS
  - Instance terminates $\Rightarrow$ IP address recycled = your traffic goes to another server!
  - No support for virtual IP
- No multicast
  - Used by popular Java clustering technologies
  - Need to use TCP-based frameworks, e.g. Terracotta

Bandwidth costs

- Hosting companies:
  - Bundle internet access
  - E.g. 1000G-2000G/month
- EC2 bandwidth fees:
  - $0.10/G up
  - $0.18/G down
  - 1000G/month = > $180/month

Agenda

- Introduction to EC2
- Using EC2
- Overview of Cloud Tools
- Developing on EC2
- Using EC2 in production
- **A few thoughts about Groovy**

About Groovy

- Object-oriented, dynamic language
- Java compatible
- Runs on the JVM

Things I like: Java compatible

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

class EC2RequestExecutor {

 def Log logger = LogFactory.getLog(getClass())

 public String calculateRFC2104HMAC(String data, String key) {
   try {
    SecretKeySpec signingKey = new
              SecretKeySpec(key.getBytes("UTF8"),
                    HMAC_SHA1_ALGORITHM);
    Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
    mac.init(signingKey);
    byte[] rawHmac = mac.doFinal(data.getBytes());
    return new String(Base64.encodeBase64(rawHmac));
   }
   catch (Exception e) {
    throw new RuntimeException("Failed to generate HMAC : ",
                e.getMessage());
   }
 }
```

Closures, Gpath exprs, xml

```
def client = new HttpClient()
...
def responseStream = getMethod.getResponseBodyAsStream()
def parser = new XmlParser(false, false)
def response = parser.parseText(responseStream)

def newServers = response.instancesSet.item.collect {
   new EC2Server(this, awsProperties, ssh,
            it.instanceId[0].text(),
            it.instanceState[0].children()[1].value()[0])
}
```

```
def sortServers(servers) {
  servers.sort {a, b -> a.instanceId <=> b.instanceId}
}

public EC2Server findInstance(String instanceId) {
  def server = servers.find {instanceId == it.instanceId}
  if (server)
    return server
  else  throw new RuntimeException("InstanceId not found: " + instanceId)
 }
```

Gstrings and templates

```
def schemaScript = """
   DROP SCHEMA IF EXISTS ${schemaSpec.name};
   CREATE SCHEMA ${schemaSpec.name};
      """
```

```
String process(String templateName, Map params) {
     InputStream stream = getClass().getResourceAsStream(templateName)
     def engine = new groovy.text.SimpleTemplateEngine()
     engine.createTemplate(new  InputStreamReader(stream)).make(params).toString()
  }
```

```
...
<%
      contexts.each {%>ProxyPass /$it balancer://mycluster/$it stickysession=jsessionid
<%}%>
<Proxy balancer://mycluster>
 <%
      tomcats.each {%>BalancerMember ajp://${it.hostPrivateDnsName}:8009
route=${it.jvmRoute} min=1 keepalive=On retry=5
<%                  }
%>
...
```

Builders

```
def report(String path, hosts, cpuCount, threadCount) {
  def builder = new groovy.xml.MarkupBuilder(new OutputStreamWriter(new FileOutputStream(path)))
  builder.performanceReport {
   cpus cpuCount
   threads threadCount
   hosts.entrySet().each { hostEntry ->
    host {
     name hostEntry.key
     cpuUtil hostEntry.value.getAverageBusy()
    }
   }
   requests {
    timings.entrySet().sort{ a, b-> a.key <=> b.key}.each{ pair ->
     request {
      name pair.key
      art pair.value.average()
      errors pair.value.errorPercentage()
     }
    }
   }
   def durationValue = ((float)(endTime - startTime))/1000.0
   duration durationValue
   def tpsValue = transactionCount/ durationValue
   tps tpsValue
   art averageResponseTime()
  }
 }
```

```
<performanceReport>
  <cpus>1</cpus>
  <threads>10</threads>
  <host>
    <name>database</name>
    <cpuUtil>3.27</cpuUtil>
  </host>
  <host>
    <name>tomcat0</name>
    <cpuUtil>94.32</cpuUtil>
  </host>
…
 <duration>557.943</duration>
  <tps>10.753786677133686</tps>
  <art>916.6578333333</art>
</performanceReport>
```

Groovy IDEs drive me nuts

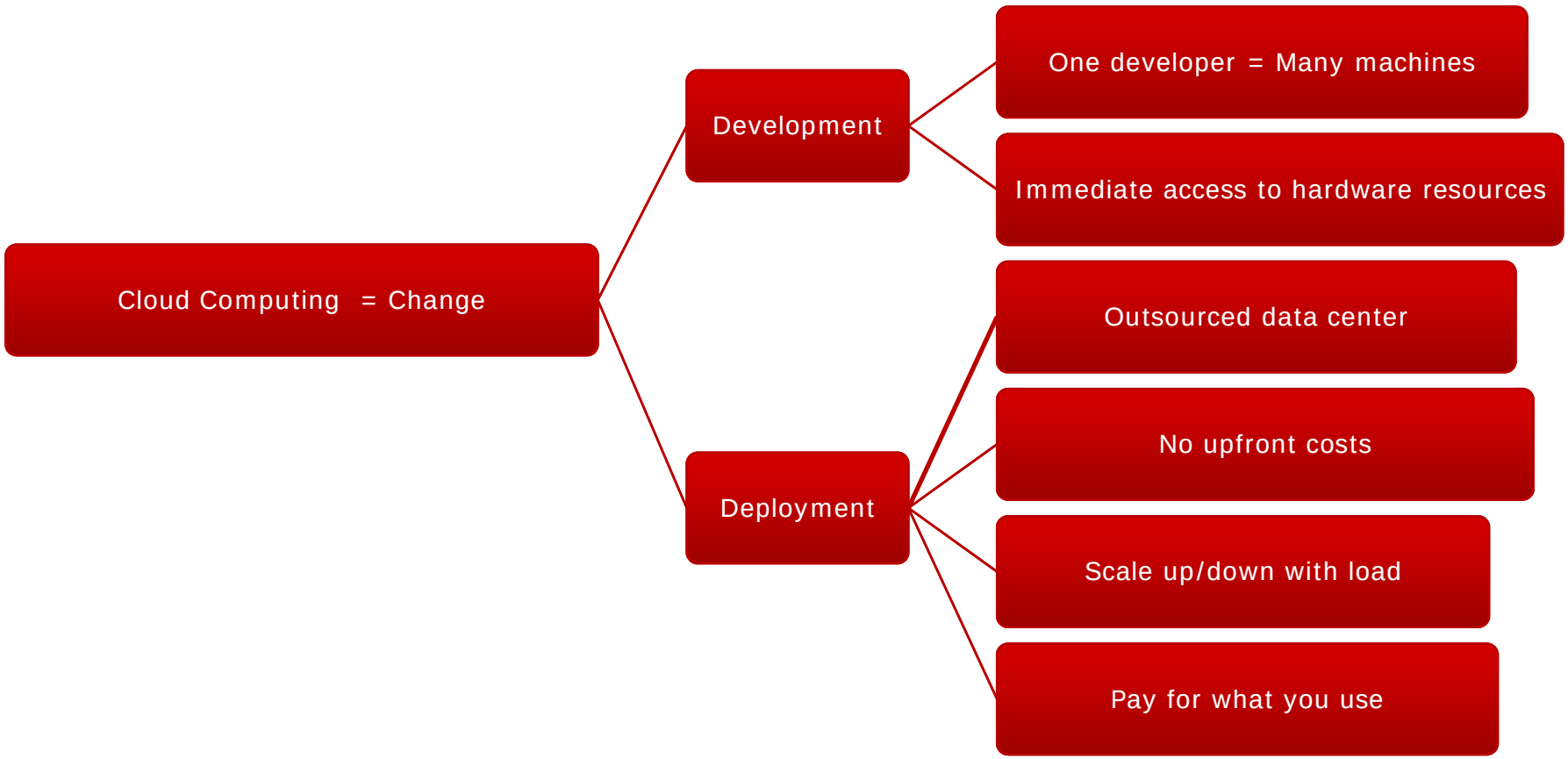
- Eclipse/IDE for Java are excellent
- But Groovy IDEs are very immature
- They have less static information to work with
- IDEA has the best support

- Limited/No refactoring
- Limited completion

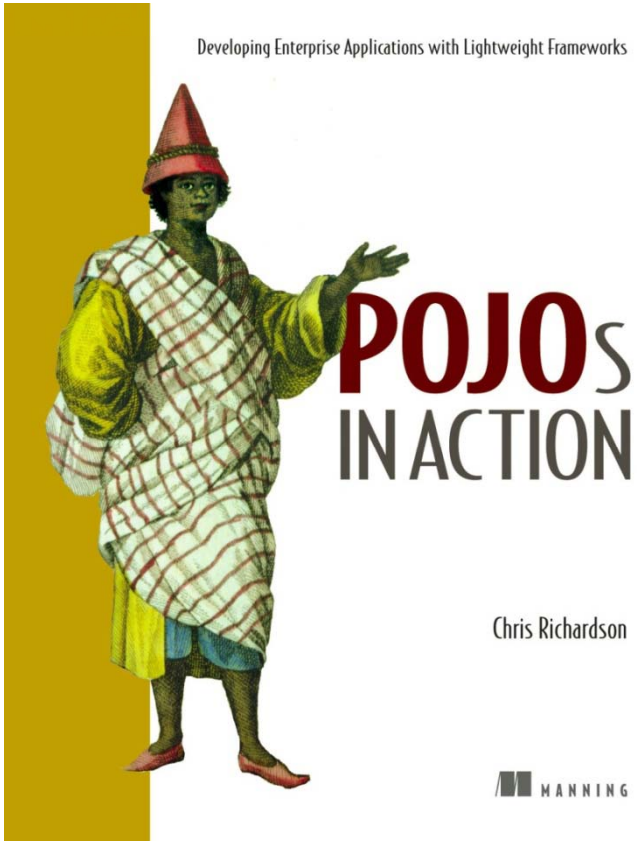
Groovy fan boys: "write unit tests"

- Unit tests are essential BUT
- When you mistype a property/field
  - Groovy:
    - Run test
    - Exception in test
    - Click a few times
    - Manually correct typo
  - Java (in Eclipse):
    - Immediate red squiggly
    - control-., control-1 (quick fix)
- Unit tests don't always catch problem
  - E.g. Mistyped method names in Object-Under-Test and mock object

Summary



For more information



- Buy my book ☺
- Send email:

chris@chrisrichardson.net

- Visit my website:

http://www.chrisrichardson.net

- Talk to me about consulting and training