Apache JClouds

Cloud interfaces, simplified

Hiranya Jayathilaka
Dept. of Computer Science, UCSB

Roadmap

- Cloud computing
- Challenges
- Apache JClouds
- Demo
- Pros and cons
- Summary and conclusion

The Cloud Revolution

- Cloud computing is revolutionizing the way software is developed and delivered.
  - Software-as-a-Service (SaaS)
  - Platform-as-a-Service (PaaS)
  - Infrastructure-as-a-Service (IaaS)

Many Benefits

- Cost effective
- Reduced maintenance overhead
- Easy and fast provisioning – Improved time-to-market
- Autoscaling and elasticity
- Fault tolerance

The Other Side of the Coin

Cloud Services

Google
app engine

Windows Azure

EUCALYPTUS

vcloud

CloudOne

ElasticHosts
Flexible servers in the cloud

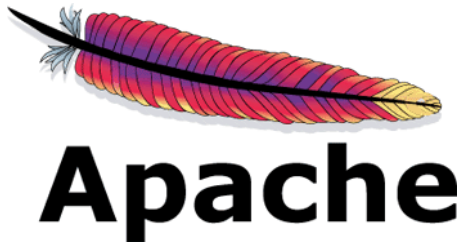
SOFTLAYER®

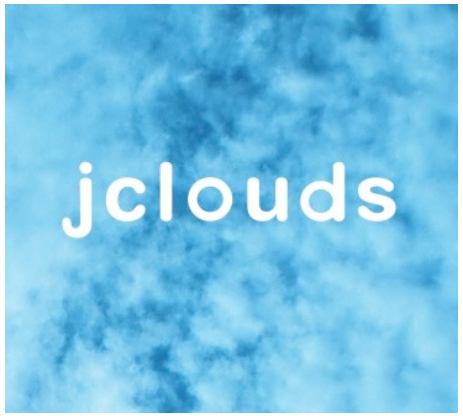
Challenges

- How do you select the cloud provider that's right for you?
- How do you port your application from one cloud provider to another?
- How to develop multi-cloud applications?

Enter, JClouds

- An open source library that facilitates developing applications for a wide range of cloud providers.
- Implement your application using JClouds, and run it on your favorite cloud without any code changes.
- Simple, intuitive APIs.
- Java and Clojure support.





History

- Started in March 2009 by Adrian Cole as an open source project.
- Initially based in London, developed mostly by European Java community.
- Contributed to Apache in April 2013.
  - http://wiki.apache.org/incubator/jcloudsProposal
- Already popular in the industry – Used by the likes of Adobe, CloudBees, RedHat, Twitter and SalesForce.
  - http://jclouds.incubator.apache.org/documentation/reference/apps-that-use-jclouds/

Getting Started

- No binary download ☺
- The documentation provides the necessary configurations for popular project management tools, so JClouds can be included in your project as a dependency.
  - Maven
  - ANT
  - Leiningen

JClouds APIs

- Compute Service API
  - For managing compute nodes (VMs) in the cloud
- Blobstore API
  - For storing data in the cloud

Compute Service API

- Key abstractions
  - Hardware
  - Operating system
  - Template
- Supported providers
  - AWS (EC2), Bluelock, CloudSigma, ElasticHosts, Go2Cloud, GoGrid, Green House Data, HP, Ninefold, OpenHosting, Rackspace, ServerLove, SkaliCloud, SoftLayer, Stratogen, TRMK, TryStack

Starting a VM from an Image

```
public static Set<? extends NodeMetadata> startVM(String region, String imageId, int count) {
    ComputeServiceContext context = ContextBuilder.newBuilder("ec2").
            credentials("my-access-key", "my-secret-key").
            modules(ImmutableSet.<~>of(new SshjSshClientModule())).
            buildView(ComputeServiceContext.class);

    ComputeService service = context.getComputeService();
    Template template = service.templateBuilder().
            imageId(region + "/" + imageId).
            smallest().build();

    try {
        return service.createNodesInGroup("jclouds", count, template);
    } catch (RunNodesException e) {
        handleException("Error starting the VMs", e);
    }
    return null;
}
```

If You Don't Have an Image?

```
public static Set<? extends NodeMetadata> startVM(String region, int count) {
    ComputeServiceContext context = ContextBuilder.newBuilder("ec2").
            credentials("my-access-key", "my-secret-key").
            modules(ImmutableSet.<~>of(new SshjSshClientModule())).
            buildView(ComputeServiceContext.class);

    ComputeService service = context.getComputeService();
    Template template = service.templateBuilder().
            osFamily(OsFamily.UBUNTU).
            osVersionMatches("12.04").
            locationId(region).
            smallest().
            build();

    try {
        return service.createNodesInGroup("jclouds", count, template);
    } catch (RunNodesException e) {
        handleException("Error starting the VMs", e);
    }
    return null;
}
```

More Control Over Templates

```
Template template = service.templateBuilder().
        osFamily(OsFamily.UBUNTU).
        osVersionMatches("12.04").
        os64Bit(true).
        minCores(2).
        minRam(1024).
        minDisk(2).
        locationId(region).
        build();
```

Access VM Metadata

```
public static void logNodeInfo(NodeMetadata node) {
    log.debug("New node started successfully.");
    log.debug("Node ID: " + node.getId());
    log.debug("Node Image: " + node.getImageId());
    log.debug("Node OS: " + node.getOperatingSystem().getFamily().value() +
            "-" + node.getOperatingSystem().getVersion());
    log.debug("Node Hostname: " + node.getHostname());

    String s = "Private addresses: ";
    for (String address : node.getPrivateAddresses()) {
        s += address + "; ";
    }
    log.debug(s);

    s = "Public addresses: ";
    for (String address : node.getPublicAddresses()) {
        s += address + "; ";
    }
    log.debug(s);
}
```

SSH to Remote VM

```
ComputeServiceContext context = ContextBuilder.newBuilder("ec2").
        credentials("my-access-key", "my-secret-key").
        modules(ImmutableSet.<~>of(new SshjSshClientModule())).
        buildView(ComputeServiceContext.class);

NodeMetadata node = startNode(context, imageId);
SshClient client = context.utils().sshForNode().apply(node);
client.connect();
ExecResponse response = client.exec("cat /etc/issue");
if (response.getExitStatus() == 0) {
    System.out.println(response.getOutput());
} else {
    System.err.println(response.getError());
}
client.disconnect();
```

Dealing with Package Managers

```
NodeMetadata node = startNode(context, imageId);
if (OperatingSystemPredicates.supportsApt().apply(node.getOperatingSystem())) {
    // Run apt-get commands
} else if (OperatingSystemPredicates.supportsYum().apply(node.getOperatingSystem())) {
    // Run yum commands
} else if (OperatingSystemPredicates.supportsZypper().apply(node.getOperatingSystem())) {
    // Run zypper commands
} else {
    // Default to something else - Or throw error
}
```

Managing Clusters

```
ComputeServiceContext context = ContextBuilder.newBuilder("ec2").
        credentials("my-access-key", "my-secret-key").
        modules(ImmutableSet.<~>of(new SshjSshClientModule())).
        buildView(ComputeServiceContext.class);
ComputeService service = context.getComputeService();
Template template = service.templateBuilder().
        imageId(region + "/" + imageId).
        minRam(4096).
        build();
service.createNodesInGroup("cassandra_cluster", 20, template);
service.runScriptOnNodesMatching(
        NodePredicates.runningInGroup("cassandra_cluster"),
        "/usr/local/cassandra/bin/run.sh start");

// Do work....

service.destroyNodesMatching(
        NodePredicates.runningInGroup("cassandra_cluster"));
```

Blobstore API

- Key abstractions
  - Container
  - Folder
  - Blob
- Supported providers
  - AWS (S3), CloudOne, HP, Azure, Ninefold, Rackspace, Synaptic

Downloading a Blob

```
public InputStream downloadBlob(String container, String fileName) {
    BlobStoreContext context = ContextBuilder.newBuilder("aws-s3")
            .credentials("my-access-key", "my-secret-key")
            .buildView(BlobStoreContext.class);
    BlobStore blobStore = context.getBlobStore();
    Blob blob = blobStore.getBlob(container, fileName);
    return blob.getPayload().getInput();
}
```

Write Blob

```
public void writeBlob(String container, String fileName, String data) {
    BlobStoreContext context = ContextBuilder.newBuilder("aws-s3")
            .credentials("my-access-key", "my-secret-key")
            .buildView(BlobStoreContext.class);
    BlobStore blobStore = context.getBlobStore();
    Blob blob = blobStore.blobBuilder(fileName).payload(data).build();
    blobStore.putBlob(container, blob);
}
```

Upload File

```
public void uploadFile(String container, File file) {
    BlobStoreContext context = ContextBuilder.newBuilder("aws-s3")
            .credentials("my-access-key", "my-secret-key")
            .buildView(BlobStoreContext.class);
    BlobStore blobStore = context.getBlobStore();
    Blob blob = blobStore.blobBuilder(file.getName()).payload(file).build();
    blobStore.putBlob(container, blob);
}
```

Access Blob Metadata

```
MutableBlobMetadata metadata = blob.getMetadata();
System.out.println("Container: " + metadata.getContainer());
System.out.println("Name: " + metadata.getName());
System.out.println("ETag: " + metadata.getETag());
System.out.println("Date Created: " + metadata.getCreationDate());
System.out.println("Date Modified: " + metadata.getLastModified());
System.out.println("Provider: " + metadata.getProviderId());
System.out.println("URL: " + metadata.getPublicUri());

MutableContentMetadata contentMetadata = blob.getMetadata().getContentMetadata();
System.out.println("Size: " + contentMetadata.getContentLength());
System.out.println("MIME type: " + contentMetadata.getContentType());
System.out.println("Checksum: " + contentMetadata.getContentMD5());
System.out.println("Encoding: " + contentMetadata.getContentEncoding());
```

Logging Support

```
return ContextBuilder.newBuilder(properties.getProperty(CLOUD_PROVIDER)).
        credentials(
                properties.getProperty(CLOUD_ACCESS_KEY),
                properties.getProperty(CLOUD_SECRET_KEY)).
        modules(ImmutableSet.<~>of(
                new Log4JLoggingModule(),
                new SshjSshClientModule())).
        buildView(ComputeServiceContext.class);
```

Demonstration

- Scenario
  - Start an Ubuntu VM in EC2
  - Upload a Python script and a Java application to the VM (Java application also based on JClouds)
  - Install Python and JRE on the VM using apt-get
  - Run the Java application on the VM to download a data file from S3
  - Run the Python script on the data file
  - Upload the output to S3

Pros

- Powerful abstractions: Simple + High-level + Convention over configuration
- Location awareness baked into the API
- Easily manage clusters of nodes
- Excellent portability
- Unit testable
- Free and open source (ASL 2.0)
- “It just works” (YMMV)

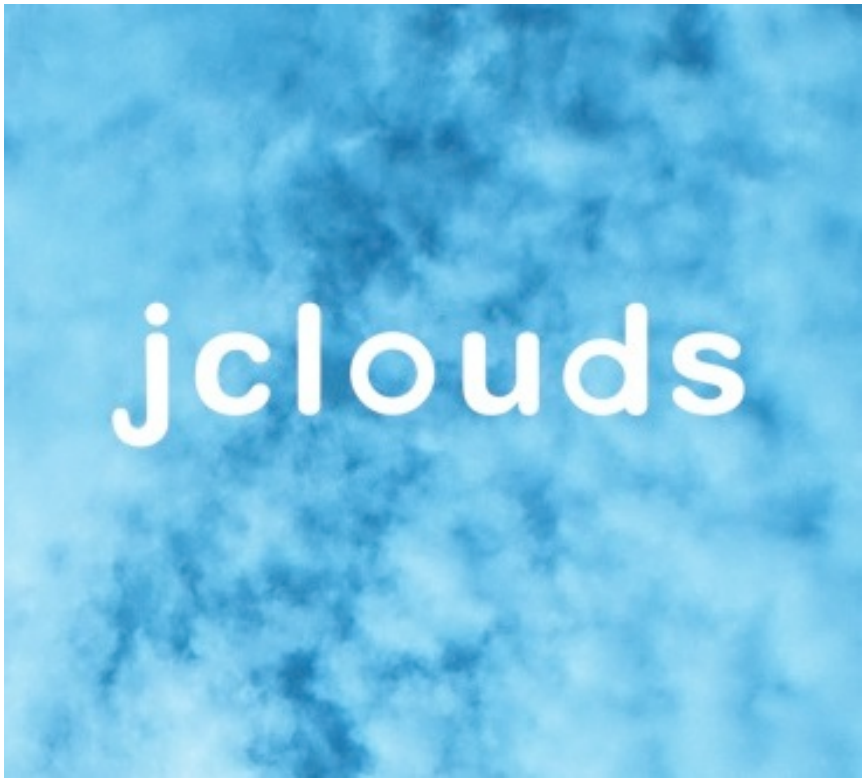
Cons

- Still in the Apache Incubator
- Limited documentation
- Potential “Jar hell”

Summary

- Cloud computing brings a horde of benefits – But the diversity can be overwhelming.
- Several challenges with respect to evaluating cloud vendors and porting application across cloud platforms.
- Apache JClouds provides a simple and feature-rich approach for developing cloud applications that are easily portable across multiple vendors.

Thank You & Questions



References

- Project website: http://jclouds.incubator.apache.org
- Apache Incubator proposal: http://wiki.apache.org/incubator/jcloudsProposal
- Getting started: http://jclouds.incubator.apache.org/documentation/gettingstarted/
- Demo code: https://github.com/hiranya911/jclouds-demo