DiCloud Deliverable S2c – Jan 29, 2010

Orca Handlers to Allocate Cloud Resources

Michael Zink, Prashant Shenoy, Jim Kurose, David Irwin and Emmanuel Cecchet
{zink, shenoy, kurose, irwin, cecchet}@cs.umass.edu

University of Massachusetts, Amherst
140 Governors Drive
Amherst, MA 01003-9264

This work implements Orca handlers to allocate and revoke resources from the Amazon public clouds. Three different resource types are considered:

- Elastic Compute Cloud (EC2) instances [3]: servers with local storage that are accessible through the Internet via a public IP address and from other EC2 instances via a local private IP address.
- Simple Storage Service (S3) buckets [4]: a bucket is a storage space varying from 1 byte to 5 GB that can store objects. The objects can only be accessed through a put/get interface.
- Elastic Block Storage (EBS) volumes [5]: EBS is a Network Attach Storage (NAS) that can be mounted and accessed by only one EC2 instance at a time. Any filesystem can be created on an EBS volume.

1. Software pre-requisite

The Orca EC2 and EBS handlers directly invoke the EC2 command line API to allocate and revoke resources. The software must be installed on the machine where the handlers are running. A tutorial on how to setup the EC2 command line tools can be found in [1].
The Orca S3 handler relies on the s3cmd command line tools [6] to manipulate S3 buckets. The tools must be installed and pre-configured (*s3cmd --configure*) with the access keys on the machine where the handler is running.

2. General architecture

The Orca handlers extend the Config class but do not rely on the Ant task mechanism. Instead the handlers directly invoke EC2/S3 command line tools without requiring any additional Java library that may conflict with other Orca libraries.
Each handler implements the *join* (resource allocation), *leave* (resource de-allocation) and *probe* methods. Note that some information about the resource might not be available at allocation time (starting a new EC2 instance requires downloading first the image on the node before starting the system and allocating IP addresses). It is necessary for the plugin to continue probing the resource to gather all information that will become available when the resource is online.

Parameters are passed to a handler via a set of properties and the results are returned to the plugin through a set of properties. The properties used by Amazon handlers are described in Table 1.

Table 1. Orca - Amazon handler properties

Field Summary	
static java.lang.String	**AWS_CERT_FILE** Location of the Amazon AWS X.509 certificate file (cert-xxx.pem)
static java.lang.String	**AWS_PRIVATE_KEY_FILE** Location of the Amazon AWS private key file (pk-xxx.pem)
static java.lang.String	**EBS_CREATION_TIME** Creation time of an EBS volume
static java.lang.String	**EBS_SIZE_IN_GB** Size of an EBS volume in GB
static java.lang.String	**EBS_STATUS** Status of an EBS volume (creating, available...)
static java.lang.String	**EBS_VOLUME_ID** Id of an EBS volume that has been created
static java.lang.String	**EC2_AMI_ID** Identifier of the Amazon Image (AMI)
static java.lang.String	**EC2_AVAILABILITY_ZONE** Availability zone in which instances are created (run 'ec2-describe-availability-zones' for a list)
static java.lang.String	**EC2_INSTANCE_ID** Identifier of a running Amazon instance
static java.lang.String	**EC2_INSTANCE_NUMBER** Number of instances to create
static java.lang.String	**EC2_INSTANCE_STATE** Current instance state as reported by ec2-describe-instances
static java.lang.String	**EC2_INSTANCE_TYPE** EC2 Instance type.
static java.lang.String	**EC2_KEY_PAIR** Key pair that has been generated with ec2-add-keypair
static java.lang.String	**EC2_KEY_PAIR_NAME** Key pair name that has been registered with ec2-add-keypair
static java.lang.String	**EC2_PRIVATE_IP** Private IP of an EC2 instance (only accessible within EC2)
static java.lang.String	**EC2_PRIVATE_NAME** Private name of an EC2 instance (only accessible within EC2)
static java.lang.String	**EC2_PUBLIC_IP** Public IP of an EC2 instance (registered in DNS)
static java.lang.String	**EC2_PUBLIC_NAME** Public name of an EC2 instance (registered in DNS)
static java.lang.String	**EC2_REGION** Region in which instances are created (run 'ec2-describe-regions' for a list)
static java.lang.String	**EC2_START_TIME** Time at which the instance was started

static java.lang.String	**S3_BUCKET_NAME** S3 bucket name
static java.lang.String	**S3_LOCATION** S3 Location (none for standard US, us-west-1 or EU)

3. Orca EC2 Handler

The Authority owns the Amazon AWS private key and X509 certificate. The handler must be able to access these files to pass them as parameters to the EC2 command line tools. The handler allocates EC2 instances with CloudWatch [2] enabled for monitoring purposes. A new key-pair is automatically generated if none is provided.
The following Javadoc describes the inputs and outputs of the various methods of the handler.

3.1. join

```
public void join(java.lang.Object token,
                 java.util.Properties p)
          throws java.lang.Exception
```

Create a new EC2 instance by invoking ec2-run-instances.

```
 Mandatory properties:
   - AmazonProperties.AWS_PRIVATE_KEY_FILE: AWS account private key
   - AmazonProperties.AWS_CERT_FILE: AWS account X509 certificate
   - AmazonProperties.EC2_AMI_ID: AMI id to use to create instances
   - AmazonProperties.EC2_INSTANCE_TYPE: type of the instance to create
   - AmazonProperties.EC2_INSTANCE_NUMBER: number of instances to create
 Optional properties:
   - AmazonProperties.EC2_REGION: region where to create the instance
   - AmazonProperties.EC2_AVAILABILITY_ZONE: availability zone where to create the
instance
   - AmazonProperties.EC2_KEY_PAIR_NAME: registered key pair name to use (one will
be generated
   automatically and returned in AmazonProperties.EC2_KEY_PAIR if none is provided)
```

Returns results in the following properties:

```
 Config.PropertyTargetResultCode: ResultCodeOK on success, ResultCodeException on
error
 Config.PropertyTargetResultCodeMessage: command output (if success)
 Config.PropertyExceptionMessage: Error message (if error)
 Config.PropertyExceptionStack: stderr content if command failed

 AmazonProperties.EC2_INSTANCE_ID: Instance id
 AmazonProperties.EC2_KEY_PAIR_NAME: Name of the key pair that has been registered
 AmazonProperties.EC2_KEY_PAIR: Key pair created for this instance
```

3.2. leave

```
public void leave(java.lang.Object token,
                  java.util.Properties p)
           throws java.lang.Exception
```

Stop an EC2 instance by invoking ec2-terminate-instances

```
 Mandatory properties:
   - AmazonProperties.AWS_PRIVATE_KEY_FILE: AWS account private key
   - AmazonProperties.AWS_CERT_FILE: AWS account X509 certificate
   - AmazonProperties.EC2_INSTANCE_ID: AMI instance id to probe
 Optional properties:
   - AmazonProperties.EC2_REGION: region where to create the instance
   - AmazonProperties.EC2_KEY_PAIR_NAME: Key pair name used for this instance
   (if the keypair is a temporary keypair starting with KEY_PAIR_PREFIX, it is
deleted)
```

Returns results in the following properties:

```
 Config.PropertyTargetResultCode: ResultCodeOK on success, ResultCodeException on
error
 Config.PropertyTargetResultCodeMessage: command output (if success)
 Config.PropertyExceptionMessage: Error message (if error)
 Config.PropertyExceptionStack: stderr content if command failed
```

3.3. probe

```
public void probe(java.lang.Object token,
                  java.util.Properties p)
           throws java.lang.Exception
```

Invoke ec2-describe-instances to probe the instance. The expected properties are:

```
 Mandatory properties:
   - AmazonProperties.EC2_AWS_PRIVATE_KEY_FILE: AWS account private key
   - AmazonProperties.EC2_AWS_CERT_FILE: AWS account X509 certificate
   - AmazonProperties.EC2_INSTANCE_ID: AMI instance id to probe
 Optional properties:
   - AmazonProperties.EC2_REGION: region where to create the instance
```

Returns results in the following properties:

```
 Config.PropertyTargetResultCode: ResultCodeOK on success, ResultCodeException on
error
 Config.PropertyTargetResultCodeMessage: AMI status (if success)
 Config.PropertyExceptionMessage: Error message (if error)
 Config.PropertyExceptionStack: stderr content if command failed

 AmazonProperties.EC2_INSTANCE_ID: Instance id
 AmazonProperties.EC2_PUBLIC_NAME: Public name
 AmazonProperties.EC2_PRIVATE_NAME: Private name
 AmazonProperties.EC2_PUBLIC_IP: Public IP
 AmazonProperties.EC2_PRIVATE_IP: Private IP
 AmazonProperties.EC2_INSTANCE_STATE: Instance state
 AmazonProperties.EC2_KEY_PAIR_NAME: Name of the key pair used
 AmazonProperties.EC2_START_TIME: Start time
 AmazonProperties.EC2_AVAILABILITY_ZONE: Availability zone
```

4. Orca S3 Handler

Note that an Amazon account can only have a maximum of 100 buckets. Buckets can store any number of objects with a maximum size of 5GB per object. The storage cost is charged per GB per month. The handler only create/delete buckets.
A proxy will be provided to allow put/get operations on objects.
The following Javadoc describes the inputs and outputs of the various methods of the handler.

4.1. join

```
public void join(java.lang.Object token,
                 java.util.Properties p)
          throws java.lang.Exception
```

Create a new S3 bucket invoking 's3cmd mb'.

```
 Mandatory properties:
   - AmazonProperties.S3_BUCKET_NAME: name of the S3 bucket to create
 Optional properties:
   - AmazonProperties.S3_LOCATION: region where to create the bucket
```

Returns results in the following properties:

```
 Config.PropertyTargetResultCode: ResultCodeOK on success, ResultCodeException on
error
 Config.PropertyTargetResultCodeMessage: command output (if success)
 Config.PropertyExceptionMessage: Error message (if error)
 Config.PropertyExceptionStack: stderr content if command failed
```

4.2. leave

```
public void leave(java.lang.Object token,
                  java.util.Properties p)
           throws java.lang.Exception
```

Delete an S3 bucket by invoking 's3cmd rb'

```
 Mandatory properties:
   - AmazonProperties.S3_BUCKET_NAME: name of the S3 bucket to create
 Optional properties:
   - AmazonProperties.S3_LOCATION: region where to create the bucket
```

4.3. probe

```
public void probe(java.lang.Object token,
                  java.util.Properties p)
           throws java.lang.Exception
```

Invoke 's3cmd info' to probe an S3 bucket. The expected properties are:

```
 Mandatory properties:
   - AmazonProperties.S3_BUCKET_NAME: name of the S3 bucket to create
 Optional properties:
   - AmazonProperties.S3_LOCATION: region where to create the bucket
```

5. Orca EBS Handler

EBS volumes can range from 1GB to 1TB in size. Storage is charged per GB per month. IOs to an EBS volume are also charged per million.
The following Javadoc describes the inputs and outputs of the various methods of the handler.

5.1. join

```
public void join(java.lang.Object token,
                 java.util.Properties p)
          throws java.lang.Exception
```

Create a new EBS volume by invoking ec2-create-volume.

```
 Mandatory properties:
   - AmazonProperties.S3_BUCKET_NAME: name of the S3 bucket to create
 Optional properties:
   - AmazonProperties.S3_LOCATION: region where to create the bucket
```

Returns results in the following properties:

```
 Config.PropertyTargetResultCode: ResultCodeOK on success, ResultCodeException on
error
 Config.PropertyTargetResultCodeMessage: command output (if success)
 Config.PropertyExceptionMessage: Error message (if error)
 Config.PropertyExceptionStack: stderr content if command failed
```

5.2. leave

```
public void leave(java.lang.Object token,
                  java.util.Properties p)
           throws java.lang.Exception
```

Delete an EBS volume by invoking ec2-delete-volume

```
 Mandatory properties:
   - AmazonProperties.AWS_PRIVATE_KEY_FILE: AWS account private key
   - AmazonProperties.AWS_CERT_FILE: AWS account X509 certificate
   - AmazonProperties.EBS_VOLUME_ID: EBS volume id to delete
 Optional properties:
   - AmazonProperties.EC2_REGION: region where to create the instance
```

Returns results in the following properties:

```
 Config.PropertyTargetResultCode: ResultCodeOK on success, ResultCodeException on
error
 Config.PropertyTargetResultCodeMessage: command output (if success)
 Config.PropertyExceptionMessage: Error message (if error)
 Config.PropertyExceptionStack: stderr content if command failed
```

5.3. probe

```
public void probe(java.lang.Object token,
                  java.util.Properties p)
           throws java.lang.Exception
```

Invoke ec2-describe-volumes to probe the EBS volume. The expected properties are:

```
 Mandatory properties:
   - AmazonProperties.EC2_AWS_PRIVATE_KEY_FILE: AWS account private key
   - AmazonProperties.EC2_AWS_CERT_FILE: AWS account X509 certificate
   - AmazonProperties.EBS_VOLUME_ID: EBS volume id to probe
 Optional properties:
   - AmazonProperties.EC2_REGION: region where to create the instance
```

Returns results in the following properties:

```
 Config.PropertyTargetResultCode: ResultCodeOK on success, ResultCodeException on
error
 Config.PropertyTargetResultCodeMessage: AMI status (if success)
 Config.PropertyExceptionMessage: Error message (if error)
 Config.PropertyExceptionStack: stderr content if command failed

 AmazonProperties.EBS_VOLUME_ID: Id of the EBS volume
 AmazonProperties.EBS_CREATION_TIME: Creation time of the volume
 AmazonProperties.REGION: region where the volume has been created
 AmazonProperties.EBS_STATUS: status of the volume
 AmazonProperties.EBS_CREATION_TIME: volume creation time
```

6. Code availability

The code can be downloaded at:
http://vise.cs.umass.edu/trac/attachment/wiki/CloudControl/orca_amazon_handlers.tgz

This documentation can be found on the DiCloud project wiki at:
http://groups.geni.net/geni/wiki/DICLOUD

7. References

[1] Amazon Elastic Compute Cloud Getting Started Guide (API Version 2009-11-30) - http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/index.html?setting-up-your-tools.html.

[2] Amazon CloudWatch - http://aws.amazon.com/cloudwatch/.

[3] Amazon EC2 - http://aws.amazon.com/ec2/.

[4] Amazon S3 - http://aws.amazon.com/s3/.

[5] Amazon EBS - http://aws.amazon.com/ebs/.

[6] s3cmd: command line S3 client - http://s3tools.org/s3cmd.