# Amazon Cloud Lessons Learned

Team Sentinel Prime

Jen Goldbach, Robert Jones, Anne Markis, Nick Quaranto, Christopher White

Table of Contents

Amazon Cloud Lessons Learned 1

EC2 How-To 2

Bundling AMIs 2

Using the .NET SDK 2

EC2 Lessons Learned 3

AMIs 3

Instances 3

SSH 3

Database/SQL 4

Miscellaneous Issues 4

# EC2 How-To

## Bundling AMIs

#### Preparation

Before bundling an AMI, make sure that:

1. IIS has no sites currently running. If so, delete them.
2. SQL Server’s service and processes are running. Restart it through the “Administrative Tools > Services” menu.
3. Empty the recycle bin. If necessary, defragment the drive as well.

#### Bundling

1. Run Ec2ConfigService from the *Start Menu*, and select the second tab called “Bundle.”
2. Run “Sysprep and Shut Down”. This will stop the current instance and set it up to enable you to create an AMI out of it.
3. Watch the instance until it is in a “Stopped” state in the Amazon Management Console.
4. Under *Instance Actions*, click “Create Image.” Give it a name and start bundling.
5. To verify, create an instance of the new image and check that the settings were saved.

#### Other Notes

* Check the number of processes running on an instance before using that instance to create a new AMI. Stop processes that are duplicated or not necessary on the new AMI, as a more complex image may take longer to boot.
* On several of our bundling attempts, the Sysprep process did not maintain all of the settings we set up on the AMI. In some cases, if the sysprep step was skipped completely and the instance was shut down and bundled from the EC2 dashboard all settings were remembered correctly.
* Creating a new AMI will reset the administrator password, a new password will automatically be generated and can be obtained from the EC2 management console.

## Using the .NET SDK

#### Instance Management with the SDK

* When creating an instance with “ec2.RunInstances(instanceRequest)”, it can take several minutes for the instance to be recognized in the running-instances list
* Even if the instance was created successfully, immediately trying to extract data from the instance (eg. publicDNS) will cause exceptions. It takes a while for the DNS to resolve.
* The public DNS of the instance takes a long time to load. When you try to run the demo immediately after setting it up, an ‘unable to connect’ error will appear when navigating to the public DNS. Waiting a few moments resolves this.
* We used several try/catch blocks in the code to keep attempting to connect to the instance for SSH/database creation as the amount of time it takes instances to start can range from 10-20 minutes.
* Many problems we thought we had with the code were actually just attempts to connect to the instance too soon, so adding a sleep in as a test helped figure out if the issue was timing or the code itself. The sleeps were later refactored as a thread thrown onto the job queue.
* Ports the sites will be started on need to be opened for them to be able to be accessed.

#### Coding in the AWS SDK

* When starting an instance, create a new InstancesRequest object. Several details are necessary to include for instances to be able to be started (size, AMI ID, key name). The sser must have permission to launch the AMI (user information is stored in a separate Amazon EC2 object, which must be created first) and a key pair must be created in that Amazon account (done on the Amazon EC2 dashboard). It is also necessary to specify the minimum and maximum number of instances to start. We are setting both of these to one to start only one instance. While programmatically starting an instance includes many steps, it is one of the few well-documented EC2 examples out there.
* See CreateInstanceWorker for the AWS SDK code that creates an instance. Many of the things we needed to do with the AWS SDK follows the same pattern of creating a Request object, a Response object and then a Result object that we would actually use. The Reservation ID is the identifier of the actual instance, which can be accessed via Reservation.RunningInstance
* The state of the Amazon EC2 instance will be 'running' long before the instance can actually be accessed for SSH, SQL Server, etc.

# EC2 Lessons Learned

## AMIs

Bundling AMIs is a lengthy process, taking up to 20 minutes in some cases (the most extreme case we encountered was 30 minutes). Because of this, errors or incorrect results can be difficult to reproduce, diagnose, and track. This is a time bottleneck, and we suggest multi-tasking.

It is important to make sure that the ports you need are open. These must match Windows Firewall with the Security Group the instance is on with Amazon EC2. The ports specified in Windows Firewall do save back to the AMI, and do work fine with newly booted instances. If ports are not opened correctly, only a failed connection error will occur. This is regardless of whether the port isn’t opened in Windows Firewall or in the Security Group.

In our security group for the booted instances right now:

MS SQL Server: TCP 1433, UDP 1434, TCP 49152-65535  
TCP 1433 is the port used for communication. UDP 1434 is used to send back information on what port this SQL connection will use. The port range is where this random port will fall. The range might not be needed, depending on SQL settings.

FTP: TCP 20, 21

SSH: TCP 22

RDP: TCP 3389

HTTP: TCP 80, 8000-8010  
Each application loaded onto an instance gets its own port. This is how they are linked to from the Demolition application. It is possible to dynamically open ports over SSH when starting a demo, but Demolition uses a fixed number right now.

Users marked as ‘admin’ in IIS are different than those marked as ‘admin’ for the instance. Make sure all directories accessed by IIS give permissions to the IIS admin, specifically.

As a default, services run as the SYSTEM user. They need to be specially configured to run as Administrator instead.

## Instances

The number of instances gets big, quickly, and there is no good way to label them. From the management console, one of the most useful pieces of information is the create date/time on the instances. It was also useful to keep track of the Instances separate from AWS, naming each instance by their Demo-name.

Supposedly there is a way to inject metadata when instances are booted, but it does not show on the dashboard yet. Tagging, labeling, and grouping instances is on the “Future Roadmap” of the dashboard (<http://aws.amazon.com/console/>).

## SSH

We had to devise a way to connect to instances and interact with it. Several solutions for remote access were considered. Many examples online used a simple web service to accept commands. This seemed like too much overhead, when SSH does the job perfectly and can be expanded to run any command we needed.

The team first tried a Cygwin installer with OpenSSH. It didn’t start up after bundling, and it was generally a pain to setup and maintain.

CopSSH was the next solution we looked into, and it ran after bundling successfully and had a great installer wizard.

#### Administrator Password

Windows AMIs reset the Administrator password each time the instances were booted. This caused Admin SSH to fail. To get around this, we created another user (called SuperUser) with all of privileges whose password did not get reset each time.

## Database/SQL

#### Configuration

SQL Server needed the ports listed above in the security group to be opened to work. Also, by default the SQL Server services were stopped, we had to rebundle once because of that.

From the AMI we started with, SQL Server’s service was disabled. We had to start it up before actually being able to connect to it, just connecting with SQL Management Studio did not help. In order to enable the “sa” account to log into it, we had to follow a lot of steps:

* SQL Services need to be started before login with Windows Auth is allowed
* Log in with Windows Authentication
* Under the Security folder > Credentials, make a new credential for the “sa” user tied to Administrator.
* Under the Security folder > Login, right click on Properties
  + Set the password for the “sa” user.
  + Map the credentials previously made in this dialog
  + Under “Status” in this window make sure to click “Enabled” for login.
* Enable SQL Server authentication by right clicking on the Database, under Security > Server authentication make sure “SQL Server and Windows Authentication” is checked
* Restart the SQL Server service
* Connect to SQL Server with the “sa” account in SQL Management Studio

#### Connecting

When connecting to the SQL Server from the code, the connection string has two options, depending on OS or location:

instancesite or instancesite\\SQLEXPRESS

From our experience, instancesite has worked in all cases, instancesite\\SQLEXPRESS works only sometimes, therefore we have instancesite being used.

## Miscellaneous Issues

#### Capacity Errors

There are cases where EC2 will not allow you to boot a new instance when you have exceeded your account capacity. When creating an instance on the AWS management console, the error message reads “insufficient capacity” but the error the AWSSDK gives is a much less helpful “Maximum number of attempts reached: 3” (as 3 is the default). This can be resolved by terminating instances (remember to terminate both running AND stopped instances to free up space). It may take several minutes for Amazon to allow more instances to be created.

More info found on this ‘insufficient capacity’ error:  
<http://support.rightscale.com/06-FAQs/FAQ_0112_-_How_come_I_get_an_insufficient_capacity_error_when_launching_an_EC2_instance%3f>

This could either be an individual problem (i.e. too many instances running) OR an Amazon problem. Instance availability is based on the instance size you want to launch (i.e. small may be unavailable but medium is available). It may also be affected by the location of your AMI, geographically.

#### Billing for Instances/EC2 Administration

* Active Instances are billed by the hour, meaning an instance running for 1 hour and an instance running for 1 hour and 59 minutes will cost Paychex the same amount of money.
* All instances used in this project (as of 3/25/10) are very small Windows instances. Increasing the size of an instance may increase performance, but also cost
* When using SDK, the account name/number is used to start saved AMI’s for that user. Currently, one account is used. For multiple Amazon accounts, AMI sharing would have to be configured differently.
* After, an AMI is created, a list of other EC2 user accounts who have permission to view/use this AMI can be specified (under the “Permissions” button on the AMI’s screen