EC2 Bootstrapper Design Specification

Dan Griffin
JW Secure, Inc.

Published: June 2009

Abstract: The EC2 Bootstrapper sample demonstrates the use of Windows Presentation Foundation (WPF) on Vista and Windows 7, as well as the use of Windows Server, ASP.NET, and Microsoft SQL Server on Amazon's Elastic Compute Cloud (EC2) platform.

Contents

1 Introduction

1.1 Overview

The main priority for this project is to make it as easy as possible for an ASP.NET developer, new to cloud computing, to run a custom web application on a Windows Server instance in the cloud.

The target cloud computing platform is Amazon Web Services (AWS) EC2. Amazon announced official support for Windows virtual machines (VMs), as well as for SQL Server, in the context of its EC2 service in late 2008. This capability is attracting Windows developers who were previously uninterested in the Linux-only cloud computing model.

EC2 exposes a management model based on Web service APIs that allow VMs to be created, modified, started, and stopped as necessary. Similar APIs are offered for persistent storage and message queuing. However, like any full-featured programming environment, there is complexity and inevitable ramp-up time. The goal of this project is to allow a Windows developer to skip the ramp-up required to provision a custom machine image, run it, upload a custom software package, and install it.

The console is implemented as a WPF client application.

1.2 Benefits

The project shows that the Windows client application development model supports and integrates well with the VM-based cloud computing model.

With the advent of full Windows and SQL support for EC2, Windows developers are looking for better tools to help them ramp-up and use this service. A rich, Windows-based management console that allows them to start and view machine images, and that provides a live dashboard to certain AWS services, helps them make this transition.

This project is also intended to be a showcase for WPF: cool, practical, inspirational, and consisting of significant reusable content for customers to create other dashboards down the road.

1.3 Components

This project consists of the following components.

1. A WPF client application exposing management capabilities for Windows machine instances running on EC2. It interfaces with the documented AWS web services programming model.
2. A sample ASP.NET web application that can be deployed to a virtualized Windows Server instance running on EC2. For convenience we chose the Bloggregator application on CodePlex.

2 Requirements

The main requirement is that the client application be WPF. It is expected to look attractive and compelling and it must run on Vista and Win 7. Finally, as stated above, it must interact with EC2 in a useful way for ASP.NET developers.

3 User Scenario

This is the target user scenario, written before EC2B was implemented. You can see from the User Interface walkthrough in the next section that it's very close, but not exact.

In this scenario, Jack is new to cloud computing. He has not yet created an AWS account.

1. Jack downloads the latest EC2 Console package from CodePlex and runs it on his Vista developer workstation.
2. The console opens. Since this is the first time it's been run, it prompts Jack for his AWS credentials.
    a. Since Jack hasn't registered with AWS yet, he clicks a provided link which takes him to the registration page.
    b. He then returns to the console and enters his EC2 credentials.
3. Jack doesn't have any machine instances registered. He uses the console menu option to launch a new one.
    a. Jack selects the default Windows server image recommended by the tool.
    b. In general, the tool provides acceptable defaults in order to lessen Jack's ramp-up.
4. Once the machine is running, Jack downloads and builds the sample web application included with the EC2B project on CodePlex.
5. Jack then uses the console to select the web app MSI, deploy it to the new machine instance, and install it (see development plan below on how this can be implemented).
6. Jack browses to the public DNS name of the machine instance and confirms that the web site is available and running.

4 Design

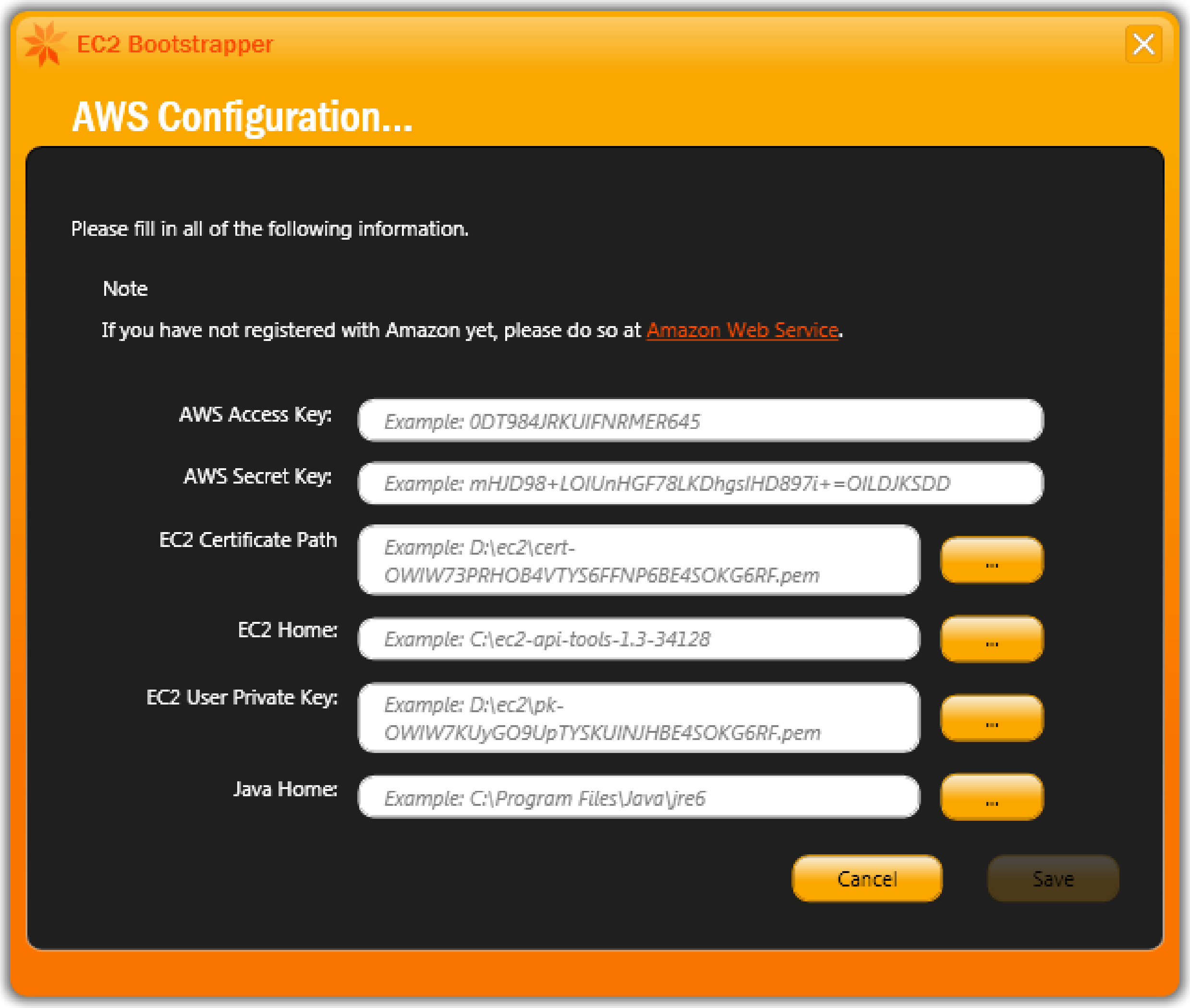
4.1 Architecture

The architecture for this project is straightforward, as it consists only of a Windows client application which consumes the AWS web service API. The architecture for the sample web application will be trivial as well.

4.2 User Interface

This section shows the flow of the EC2B user interface as implemented. It serves as a walkthrough of the core scenario described above: launching the user's first EC2 Windows machine instance, installing an ASP.NET application, and demonstrating that the new site is live.

To follow along, install and run EC2 Bootstrapper. If you haven't previously configured it, you'll see the following dialog. You must provide valid values for all six fields.



Please fill in all of the following information.

Note

If you have not registered with Amazon yet, please do so at Amazon Web Service.

AWS Access Key: Example: 0DT984JRKUIFNRMER645

AWS Secret Key: Example: mHJD98+LOIUnHGF78LKDhgslHD897i+=OILDJKSDD

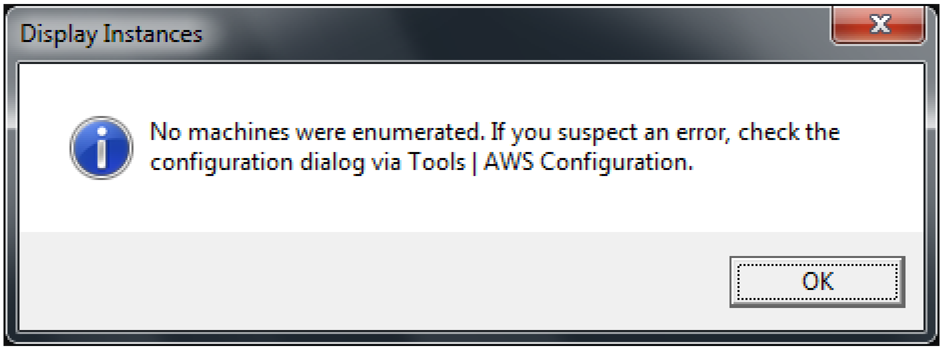
EC2 Certificate Path Example: D:\ec2\cert-OWIW73PRHOB4VTYS6FFNP6BE4SOKG6RF.pem

EC2 Home: Example: C:\ec2-api-tools-1.3-34128

EC2 User Private Key: Example: D:\ec2\pk-OWIW7KUyGO9UpTYSKUINJHBE4SOKG6RF.pem

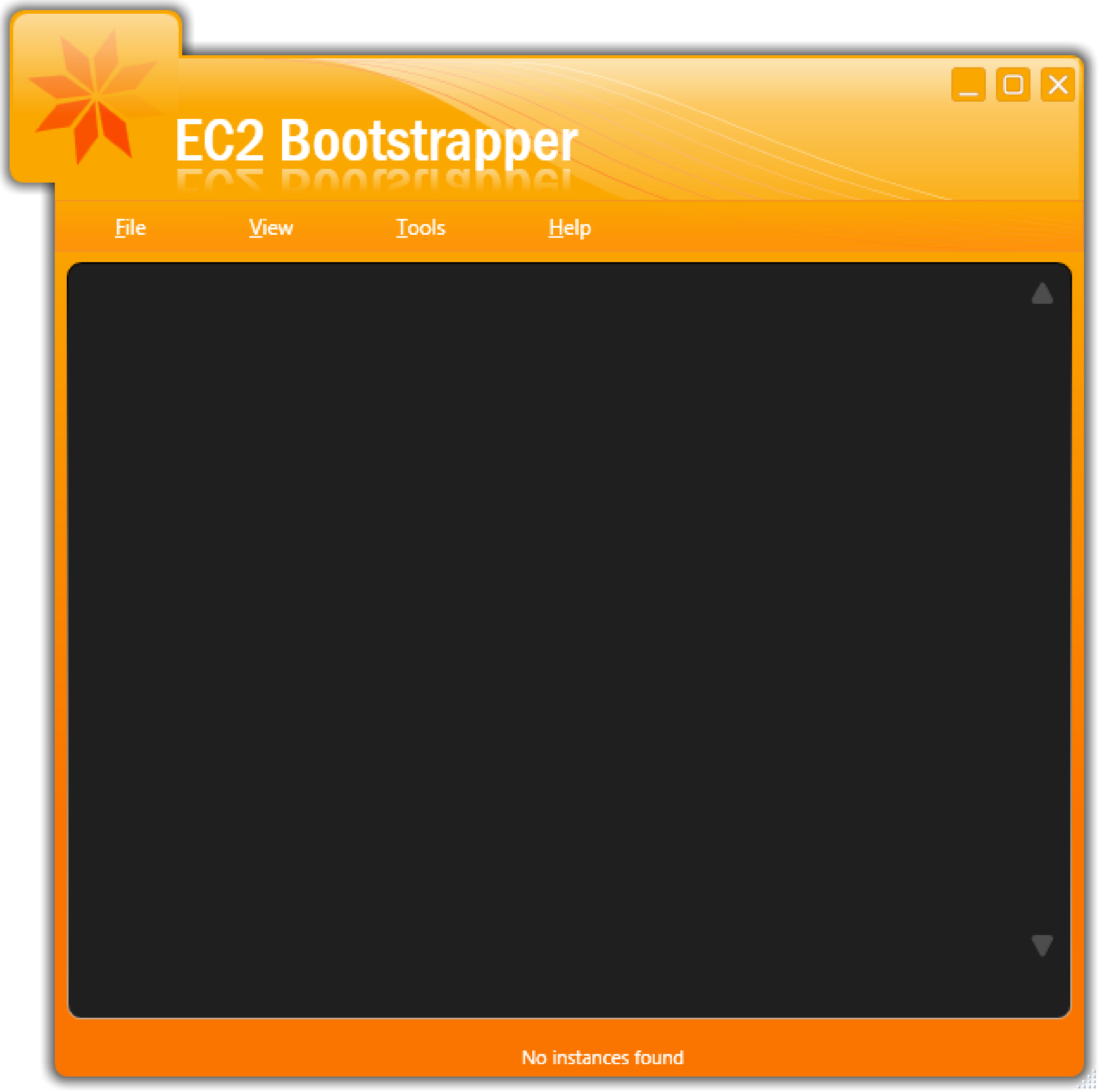
Java Home: Example: C:\Program Files\Java\jre6

After configuration is completed, click the Save button. If you don't already have any AMIs running, you'll see the following warning.



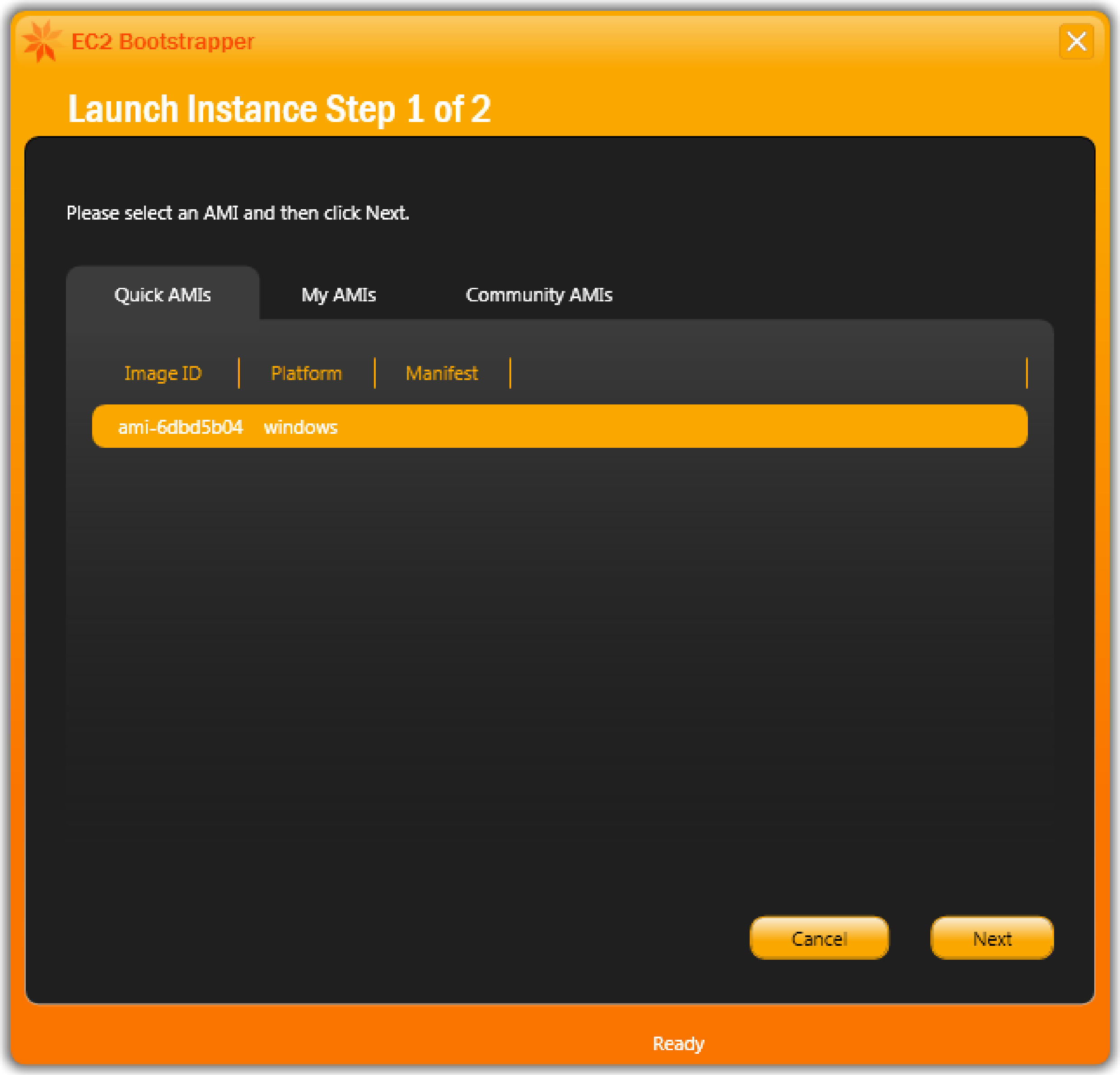
No machines were enumerated. If you suspect an error, check the configuration dialog via Tools | AWS Configuration.

Click OK. You'll next see the main EC2B window, shown below.



No instances found

Select File | Launch Instance. You’ll next see the following screen.



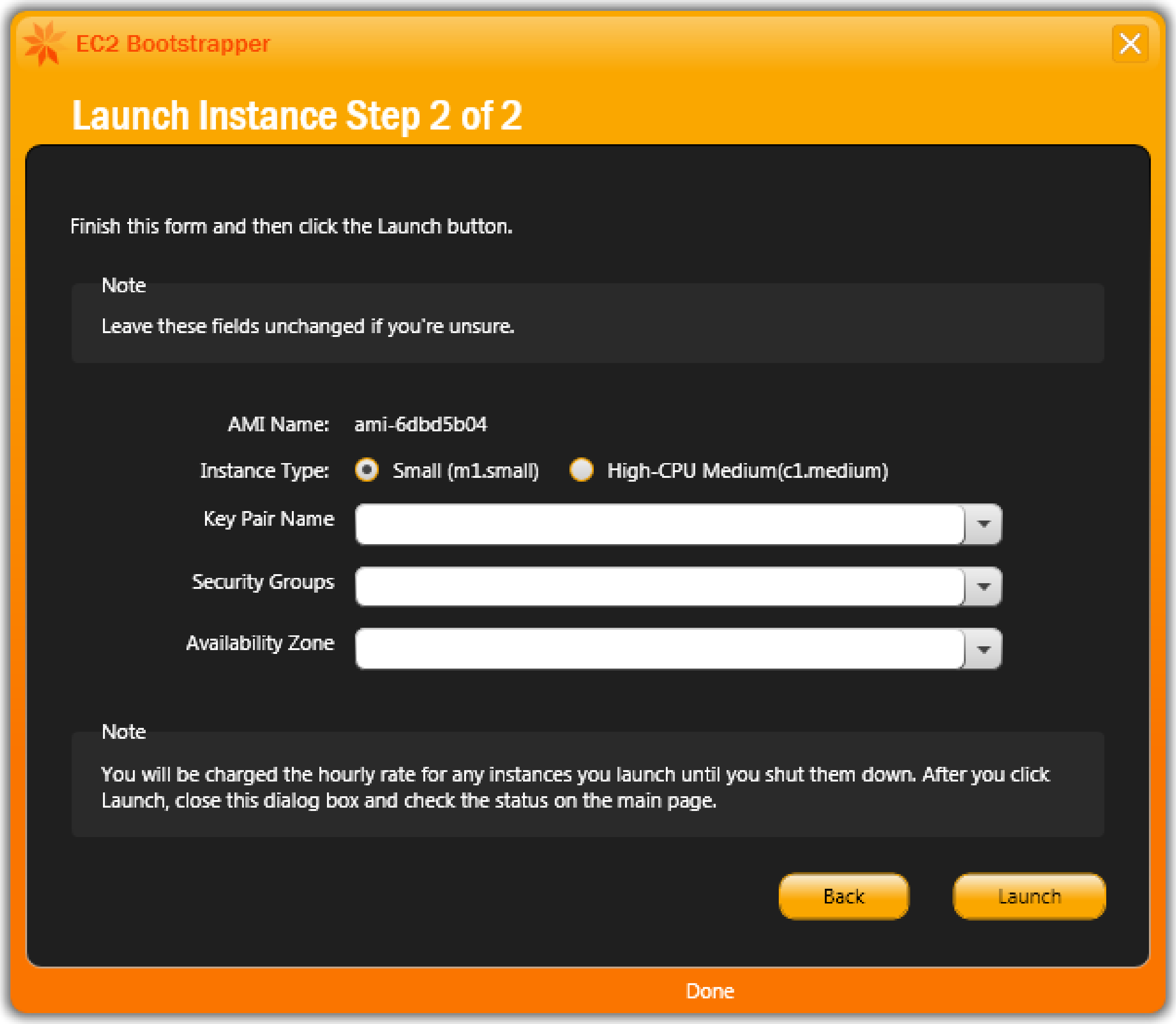
Launch Instance Step 1 of 2

Please select an AMI and then click Next.

Image ID	Platform	Manifest
ami-6dbd5b04	windows	

Ready

Click the Next button. The second Launch screen is shown below.



Launch Instance Step 2 of 2

Finish this form and then click the Launch button.

Note

Leave these fields unchanged if you're unsure.

AMI Name: ami-6dbd5b04

Instance Type: ● Small (m1.small) ○ High-CPU Medium(c1.medium)

Key Pair Name

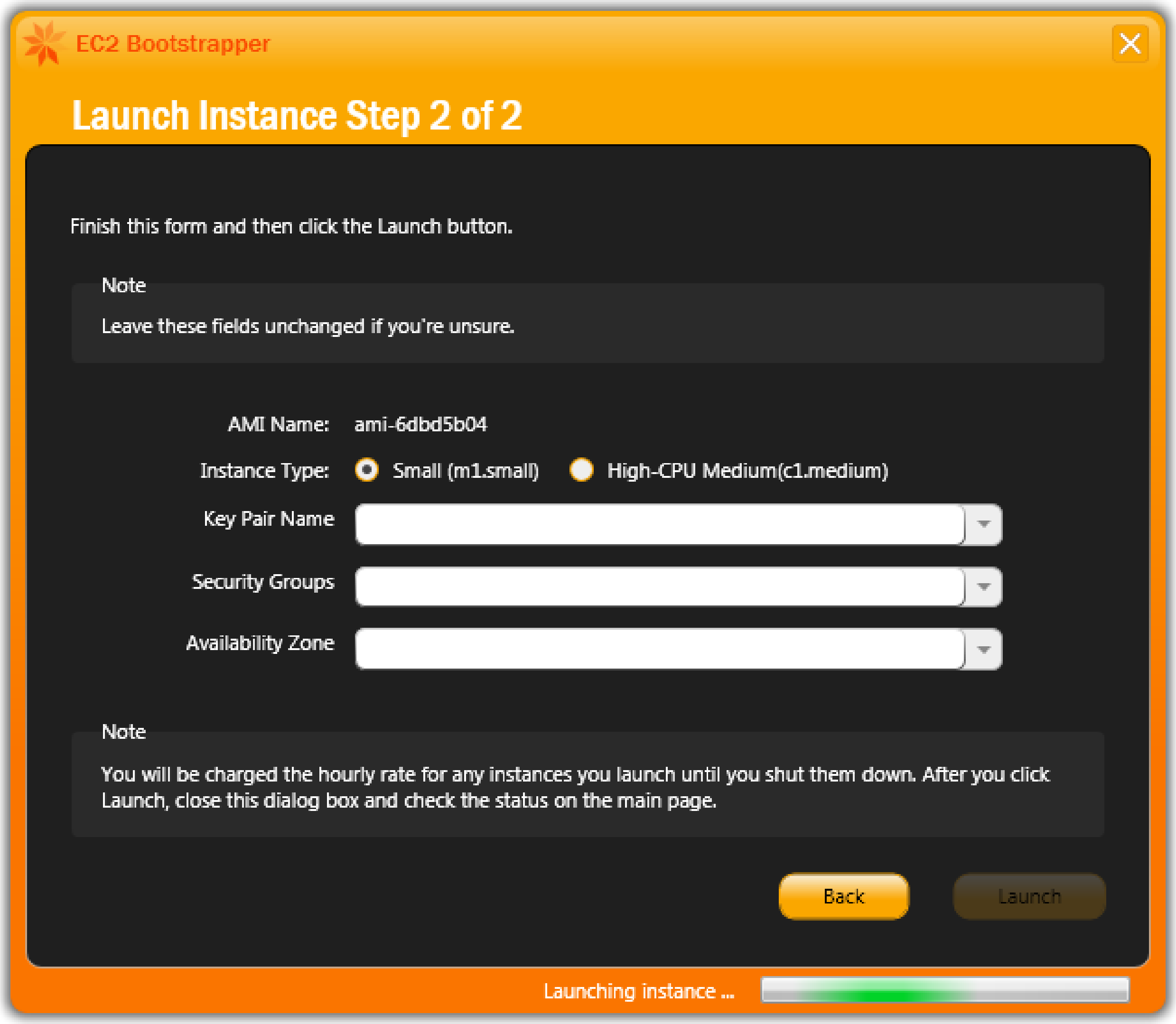
Security Groups

Availability Zone

Note

You will be charged the hourly rate for any instances you launch until you shut them down. After you click Launch, close this dialog box and check the status on the main page.

If you've already used EC2 and you have a pre-existing Key Pair for securely retrieving AMI administrator passwords, select it from the Key Pair Name dropdown. Otherwise, just accept the defaults and click the Launch button. A progress bar will appear, as shown below.



Launch Instance Step 2 of 2

Finish this form and then click the Launch button.

Note

Leave these fields unchanged if you're unsure.

AMI Name: ami-6dbd5b04

Instance Type: ● Small (m1.small) ○ High-CPU Medium(c1.medium)

Key Pair Name

Security Groups

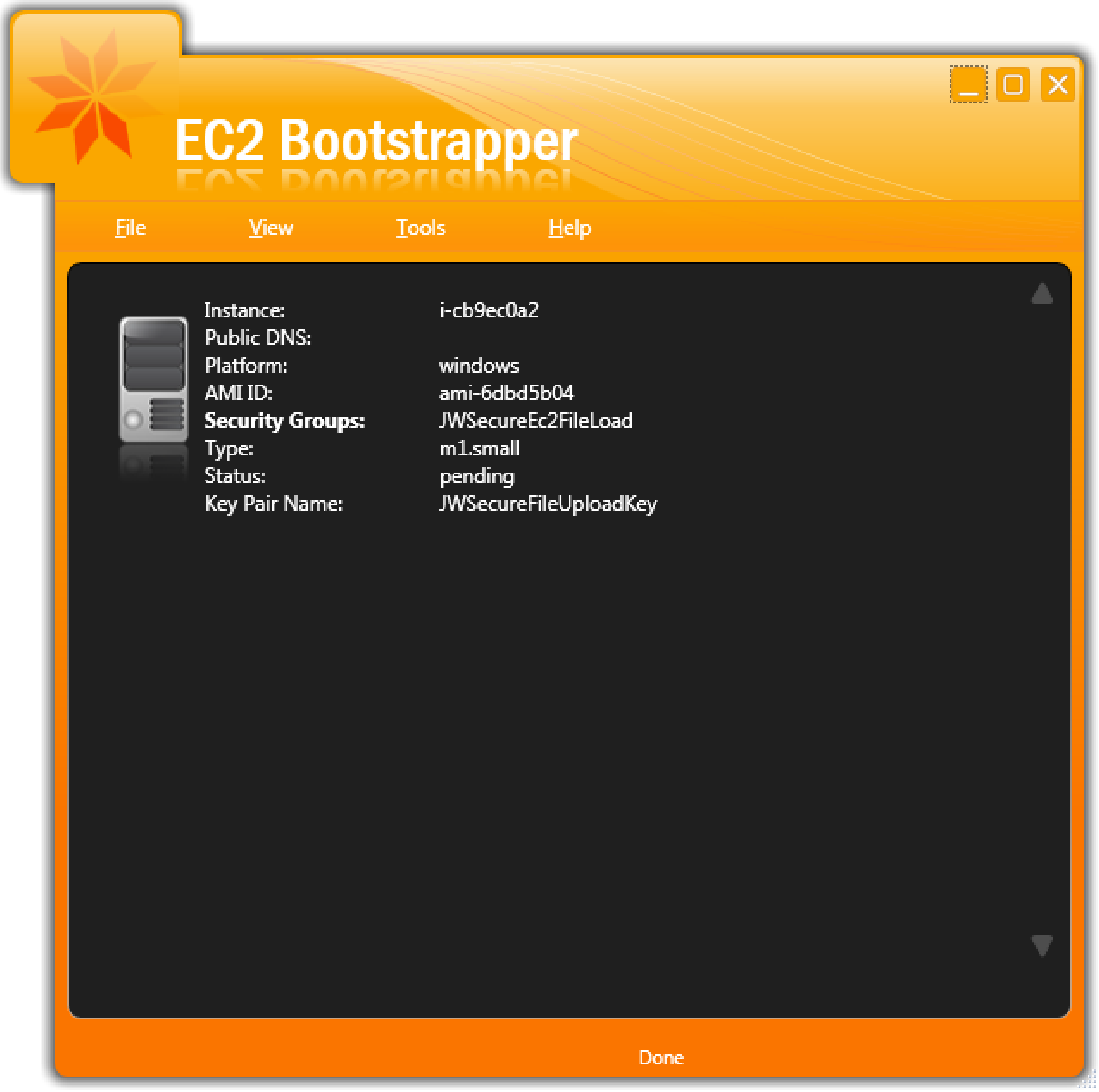
Availability Zone

Note

You will be charged the hourly rate for any instances you launch until you shut them down. After you click Launch, close this dialog box and check the status on the main page.

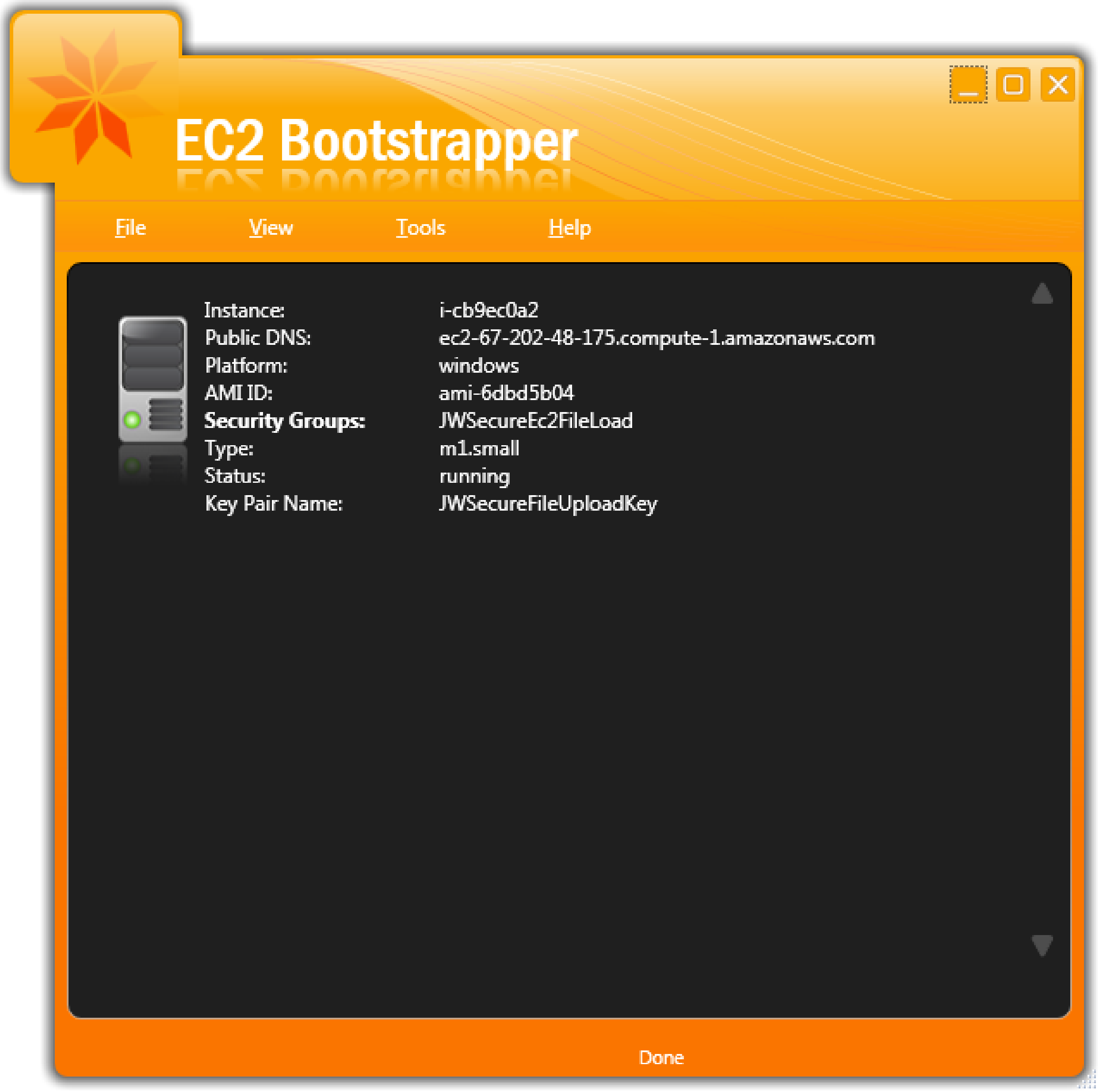
Launching instance ...

Close the above dialog by click the X or the Back button. Dismiss the Step 1 dialog in the same manner. You'll next see the main dialog, shown below. Select View | Refresh and you'll see the machine you've just launched. Note that it's in the **pending** state. Typically, it takes new AMIs around five minutes to become available. Refresh the display every few minutes to check for it.



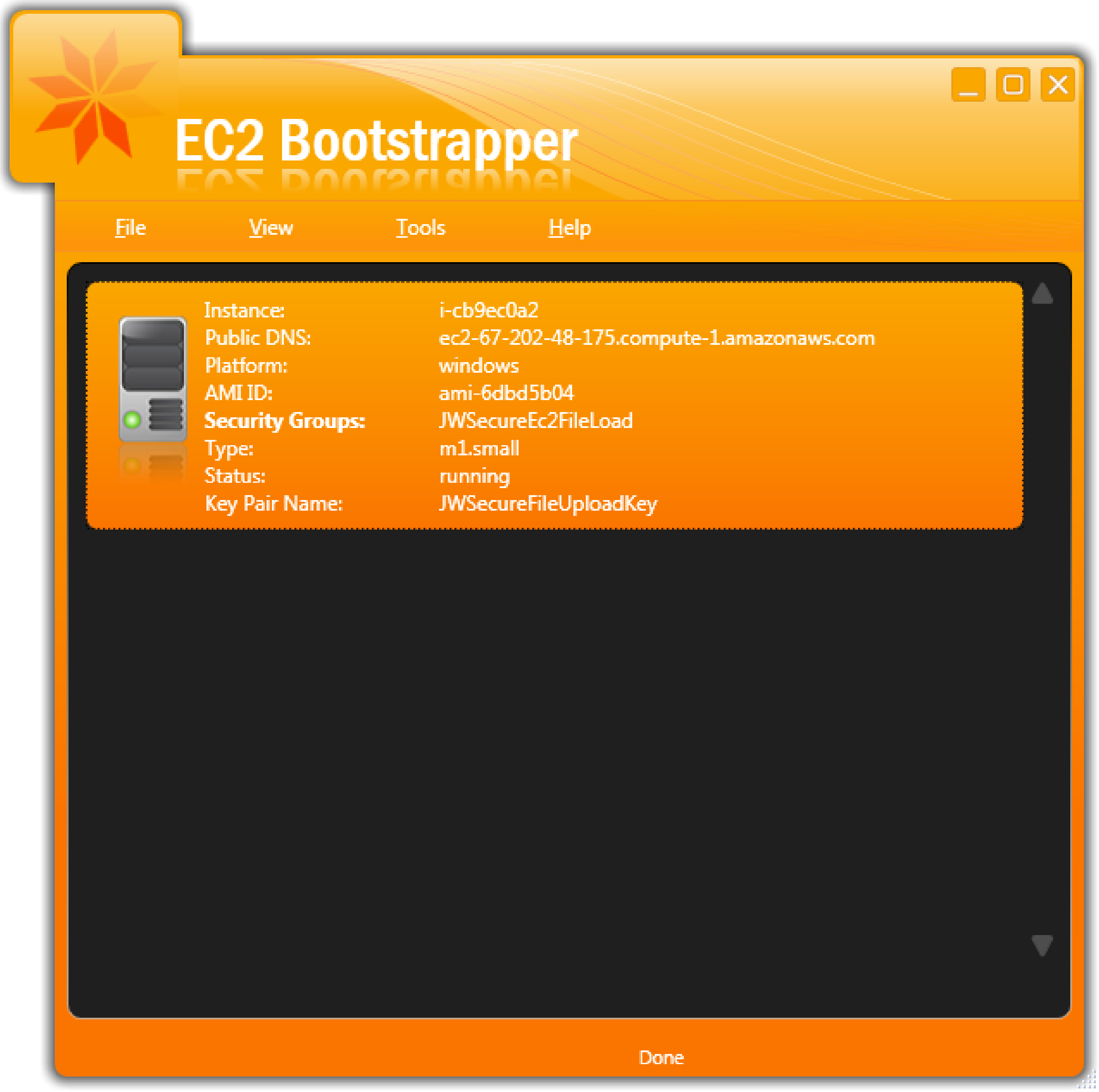
Instance: i-cb9ec0a2
Public DNS:
Platform: windows
AMI ID: ami-6dbd5b04
Security Groups: JWSecureEc2FileLoad
Type: m1.small
Status: pending
Key Pair Name: JWSecureFileUploadKey

This screen shows the new AMI as **running**:



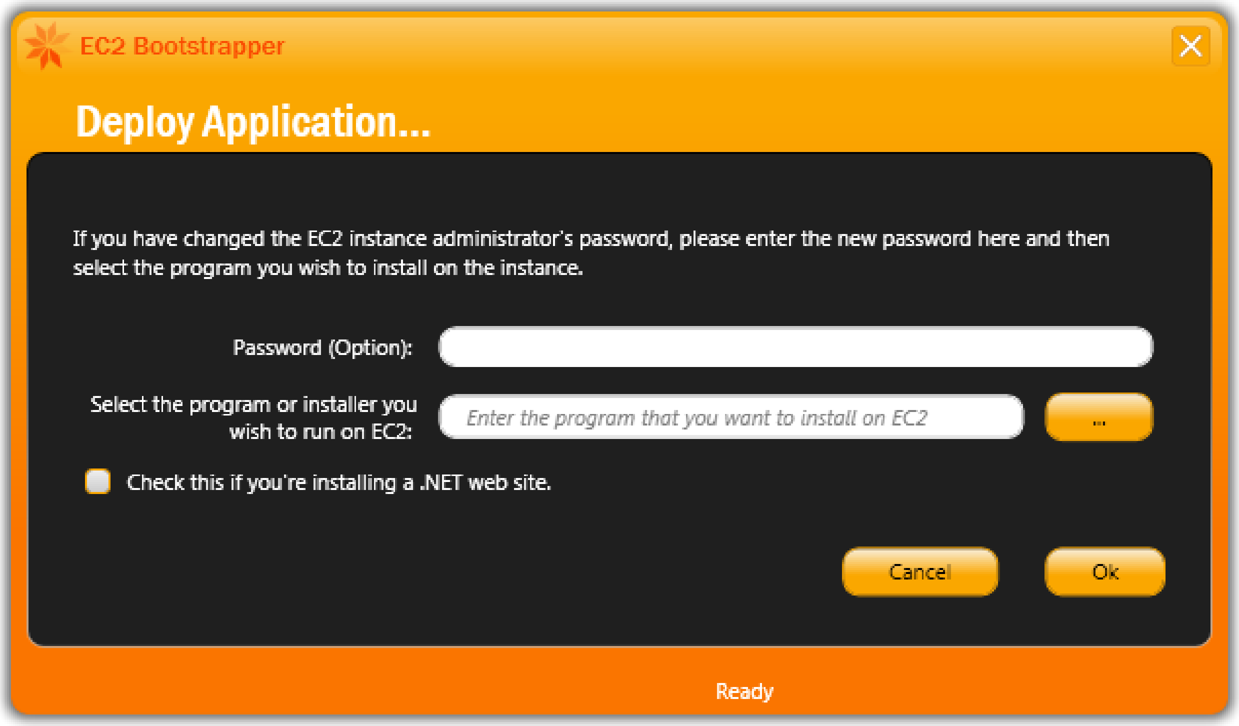
Instance:	i-cb9ec0a2
Public DNS:	ec2-67-202-48-175.compute-1.amazonaws.com
Platform:	windows
AMI ID:	ami-6dbd5b04
Security Groups:	JWSecureEc2FileLoad
Type:	m1.small
Status:	running
Key Pair Name:	JWSecureFileUploadKey

Select the instance – this highlights it orange, as shown below.



Instance: i-cb9ec0a2
Public DNS: ec2-67-202-48-175.compute-1.amazonaws.com
Platform: windows
AMI ID: ami-6dbd5b04
Security Groups: JWSecureEc2FileLoad
Type: m1.small
Status: running
Key Pair Name: JWSecureFileUploadKey

Click File | Deploy. The following dialog will appear.



If you have changed the EC2 instance administrator's password, please enter the new password here and then select the program you wish to install on the instance.

Password (Option):

Select the program or installer you wish to run on EC2: Enter the program that you want to install on EC2

☐ Check this if you're installing a .NET web site.

Ready

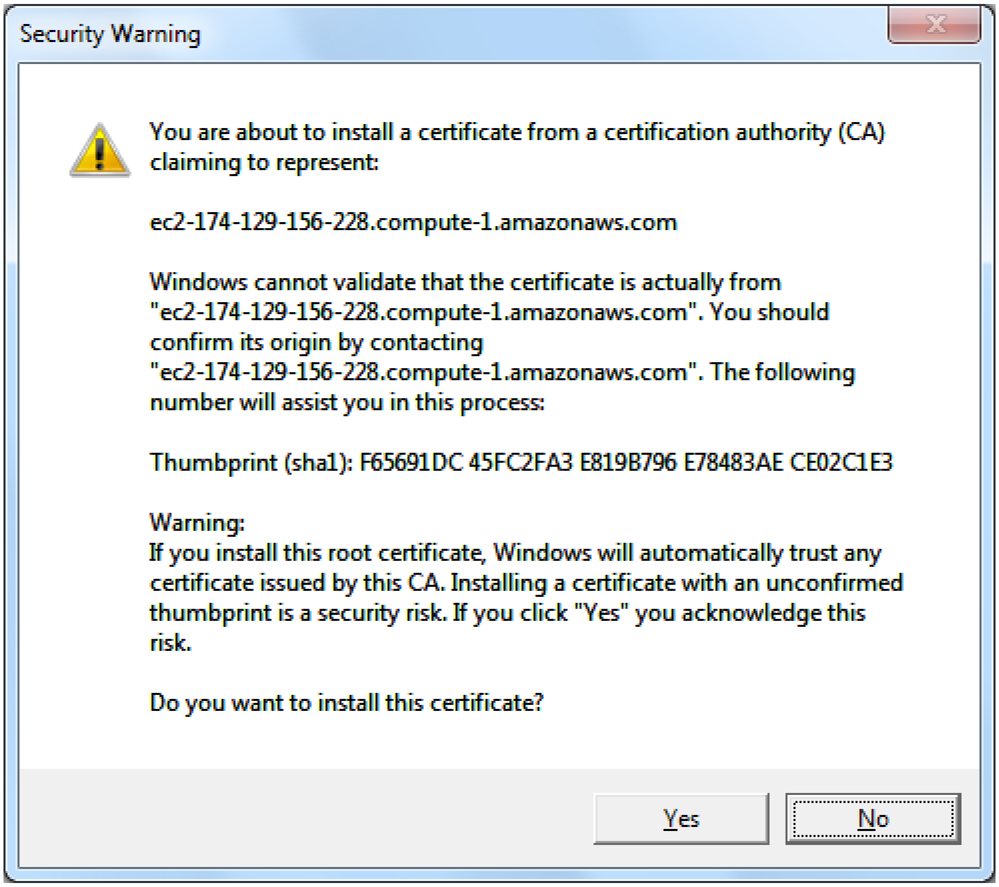
In the second field, select the path to the program or MSI you wish to run on the new AMI. For this demo, enter the full local path for Ec2SampleWebSiteSetup.msi, which is one of the projects included in the Ec2Bootstrapper.sln Visual Studio solution file available from the EC2B CodePlex site (see below). Click the check box, since the payload for that installer is indeed a .NET web site. Then click the Ok button.

If you see an “Invalid password” error at this point, the most likely cause is that you’ve used EC2 previously and an invalid or unmatched “Key Pair Name” was chosen in Launch Step 2, above. Log into https://console.aws.amazon.com, manually delete any keys you don’t need or don’t have any more, terminate the test instance you created above, start a new one, and be sure to select a valid key (i.e., in the **Key Pair Name** field in EC2B’s second Launch Instance dialog).

Another possible error is that the system time on the new instance is incorrect. If this happens, the dynamically generated server certificate may not yet be valid, in which case the client TLS connection will fail. The system time problem is documented elsewhere – see this AWS developer support link. If you run into this issue, you have two choices: log into the new instance and correct the time manually before attempting the remote install, or wait a little longer for the AMI to correct itself before attempting the install.

The MSI installation is performed via a web service on the new machine instance. The caller authenticates using HTTP basic authentication, so SSL is used in order to avoid sending the client credentials in the clear. The service then impersonates the caller to run the uploaded MSI.

In order for SSL server authentication to work on the new instance, a server certificate is created and installed “on the fly”. The client must trust this new cert in order for authentication to succeed (i.e. before the client credentials can be transmitted). The following screen shows the built-in Windows dialog for trusting a new root cert. Note the EC2 DNS name in the cert subject.



You are about to install a certificate from a certification authority (CA) claiming to represent:

ec2-174-129-156-228.compute-1.amazonaws.com

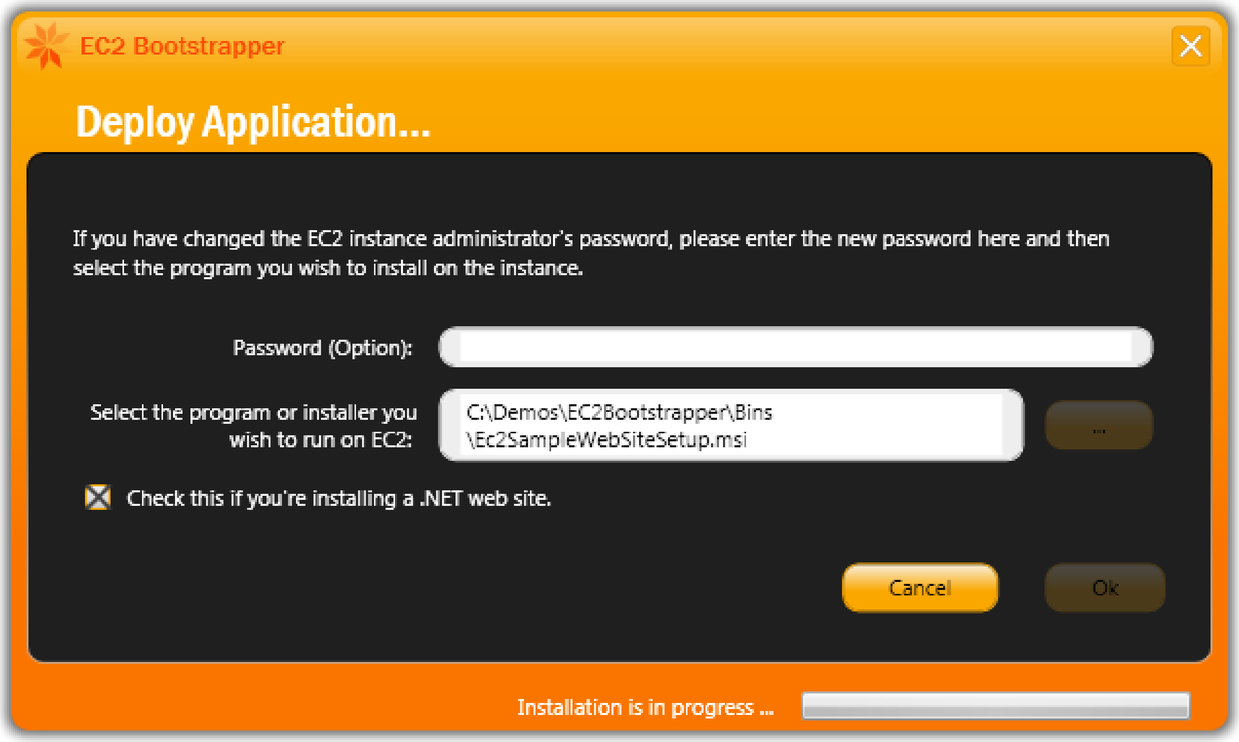
Windows cannot validate that the certificate is actually from "ec2-174-129-156-228.compute-1.amazonaws.com". You should confirm its origin by contacting "ec2-174-129-156-228.compute-1.amazonaws.com". The following number will assist you in this process:

Thumbprint (sha1): F65691DC 45FC2FA3 E819B796 E78483AE CE02C1E3

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?

After a few moments, if the AMI is successfully contacted by the client, you’ll see the prompt above. Again, this is a built-in Windows dialog, confirming that you wish to trust the indicated root certificate. In order for TLS server authentication to succeed from the client to the new AMI, you must click Yes.



Deploy Application...

If you have changed the EC2 instance administrator's password, please enter the new password here and then select the program you wish to install on the instance.

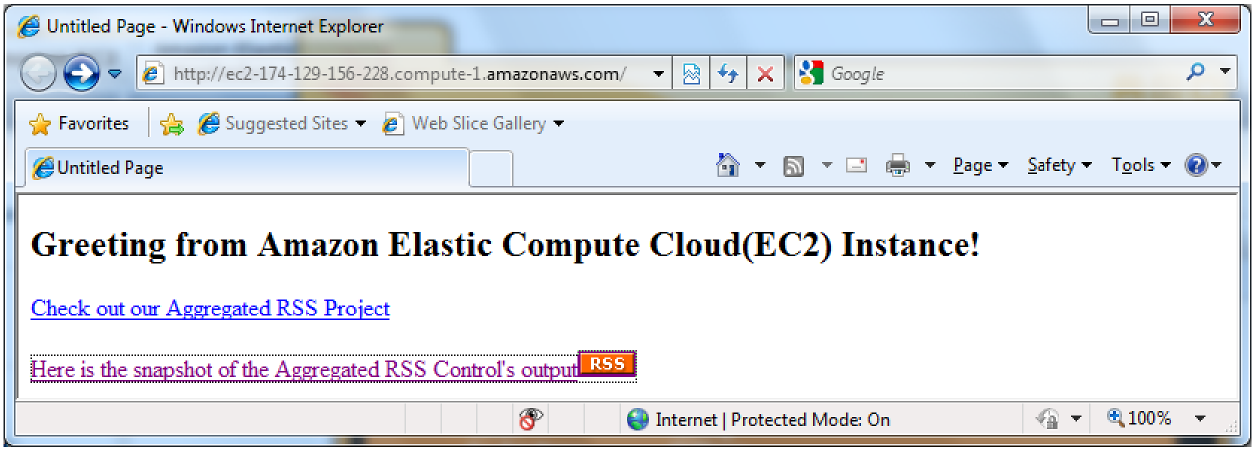
Password (Option):

Select the program or installer you wish to run on EC2: C:\Demos\EC2Bootstrapper\Bins\Ec2SampleWebSiteSetup.msi

☒ Check this if you're installing a .NET web site.

Installation is in progress ...

You'll next see the "Installation is in progress" message next to the status bar on the Deploy Application dialog as shown above. Once installation is complete, EC2B automatically launches the web browser to the default site on the target AMI. This works in this demo because the MSI we used installs its .NET web site as the default site in IIS. Any other virtual directory could be used instead by your installer, but then you'll have to fix the address in the browser manually in order to preview the site. And, of course, if the MSI isn't a web site at all, this feature is irrelevant.



http://ec2-174-129-156-228.compute-1.amazonaws.com/

Greeting from Amazon Elastic Compute Cloud(EC2) Instance!

Check out our Aggregated RSS Project

Here is the snapshot of the Aggregated RSS Control's output RSS

5 Implementation

5.1 Bootstrapping

The overall bootstrapping problem was this – how to securely upload and install the MSI to a Windows Server 2003 machine for which the RPC endpoint-mapper and SMB features are permanently blocked by the EC2 firewall. The following solution was implemented.

1. Start with one of the basic Windows AMIs already available.
2. Create and install a run-once start-up script that performs the following actions:
    a. Run SelfSSL.exe from the IIS 6.0 Resource Kit to create and install a self-signed SSL server certificate.
    b. Export the certificate (i.e., public part) and save it to a known location on the default website.
3. Create and install a Bootstrapper web service that performs the following actions:
    a. Listen on a high port.
    b. Require integrated authentication (via IIS).
    c. Impersonate the caller.
    d. Optionally, explicitly check that the caller is a member of the local administrators group (this appears to be redundant, due to the previous step).
    e. Allow (at least, if not require) SSL.
    f. Expose an endpoint that takes an MSI, saves it as a temporary file, installs it (synchronously) via msiexec.exe, and returns the result.
4. Save the above as a new AMI.

Then, when the EC2B client application requests a new instance of the above AMI, the client application performs the following steps:

A. Wait for the new instance to heartbeat.
B. Connect via HTTP to the expected location, download the server certificate, and install it in the current user trusted root certificate store.
C. Retrieve the local administrator credentials for this machine instance from the EC2 web service. (This is why a valid EC2 key pair had to be specified in the walkthrough above – it's the only way to programmatically obtain the randomized admin password.)
D. Retrieve the machine instance public DNS name
E. Connect to the Bootstrapper web service on the new machine, via HTTPS, using the administrator credentials, and passing the MSI previously selected by the user
F. Wait for the web service call to complete and report the result to the user.
G. Launch the browser to show the root web directory on the machine public DNS name.

5.2 Client

The EC2B client application interacts with AWS via the public web service API.

5.3 Sample Web Application

As stated above, the sample web application consists of an ASP.NET application, packaged using a Visual Studio Web Setup Project, and deployable via MSI to Windows Server.

6 Testing

The basic test case consists of the User Scenario and walkthrough sequences described above.

7 Developer Diary

1. You can't upload your own Windows AMI from scratch: http://developer.amazonwebservices.com/connect/message.jspa?messageID=109674. Instead, you have to start with one of Amazon's, modify it, and save it. That's how they address the licensing issue.
2. Here's how to bundle a Windows AMI: http://docs.amazonwebservices.com/AWSEC2/2008-12-01/DeveloperGuide/bundling-an-ami-windows.html.
3. Behavior of instance storage - http://docs.amazonwebservices.com/AWSEC2/2008-12-01/DeveloperGuide/instance-storage.html
4. Information can be passed to an instance via the launch metadata for a given instance - http://docs.amazonwebservices.com/AWSEC2/2008-12-01/DeveloperGuide/AESDG-chapter-instancedata.html. There's a size limit on the user-data metadata, though, so we wouldn't have been able to reliably pass a whole MSI with it, for example.

8 Resources

- Amazon Elastic Compute Cloud
- AWS Management Console
- C# Library for Amazon EC2
- Deploying ASP.NET Applications on Amazon EC2
- EC2 Bootstrapper on CodePlex
- EC2 WSDL

9 Providing Feedback

To provide feedback or report bugs in this solution, please create a new discussion using the solution's CodePlex site. You may also provide feedback by sending an e-mail to feedback@jwsecure.com.

10 About JW Secure, Inc.

JW Secure, Inc., founded in 2006, provides custom development services to software companies with security-related products. JW Secure is a Microsoft Gold Certified Partner. More information about the company’s services can be found on JW Secure’s Web site. To contact JW Secure, please see the company’s Contact page.