Using Amazon Web Services

Introduction

Amazon Web Services (AWS) is a collection of Cloud Computing services from Amazon. The services can be used on demand, which means that customers can start/stop using them whenever they want, and they only pay for what they have used. Amazon also gives new customers access to *AWS Free Tier*, which allow them to use many of their services for free for 12 months after they have signed up (as long as the services are not used too much; for more information about AWS Free Tier, see https://aws.amazon.com/free).

The intention of this document is to teach you how to use:

- Elastic Beanstalk, EB (a service running your applications on different platforms).
- Relational Database Service, RDS (a service running different databases).

Elastic Beanstalk in turn makes use of the service Elastic Compute Cloud (EC2), which is a service running your own virtual computer you can do whatever you want with.

This document contains step-by-step instructions for fundamental usage of the different services. In addition to just "getting it to work", you as a student is encouraged to explore the services on your own. If you are unsure what to do, you can always get help at the lab sessions.

Using the AWS Management Console

The easiest way to administer the AWS services you want to use is by using the *AWS Management Console*. The AWS Management Console is simply a website at https://console.aws.amazon.com/console/home you can open in any web browser. At that website, you can also create a new AWS account. After you have done that, you can access the different services by pointing the mouse at Services in the upper left corner in the console, as shown in Figure 1 below.

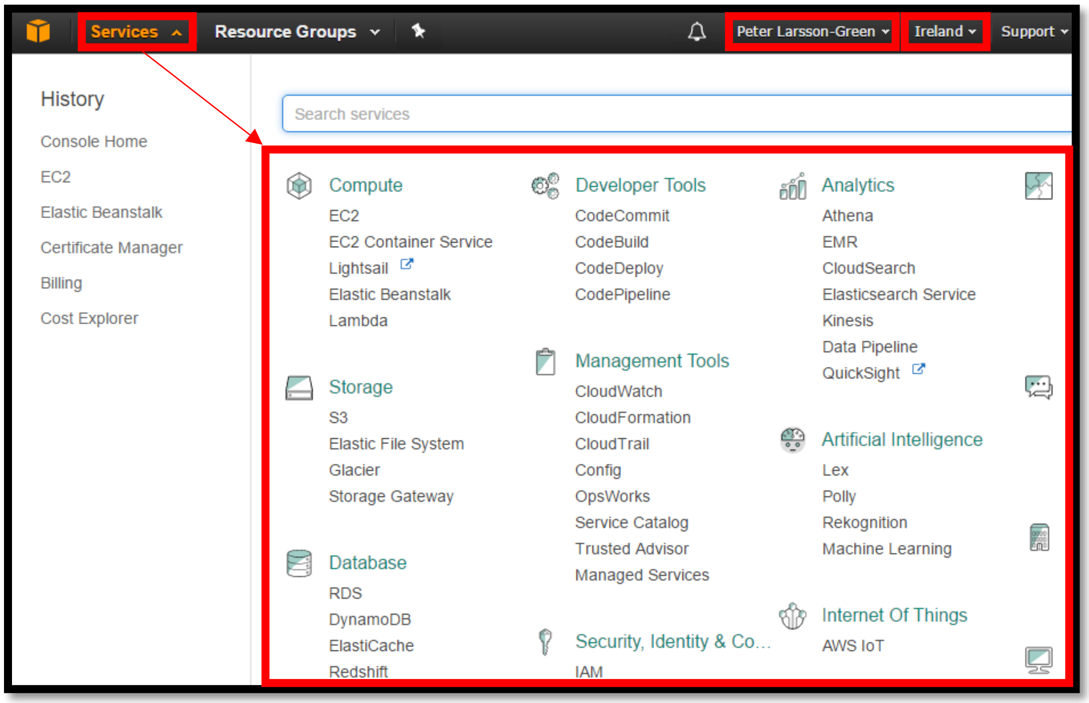


Figure 1, Move the mouse to Services in the upper left corner to see all the services. Choose desired region in the upper right corner. For more information about your account, click on your name in the upper right corner.

Most of the services you use will be served by computers from one of their data centers. In the upper right corner, you can choose the location of the data center you want to use for your services. Different locations have different pricing, but you typically want to use a location close to you/your customers. The first step for you is to choose a location close to you/your customers.

As a precaution of being charged for more than what you expect, it is a very good idea to create a budget. In the budget, you specify how much money you plan to pay for the services you are using each month, and if Amazon then discovers that they will charge you for more than this, they will notify you. Then, if you don't want to pay for more than what you have planned, you can act and shut down the services you are using.

You can create a budget by clicking on your name in the upper right corner, and then clicking on *My Billing Dashboard*, and then clicking on *Budgets* in the menu to the left.

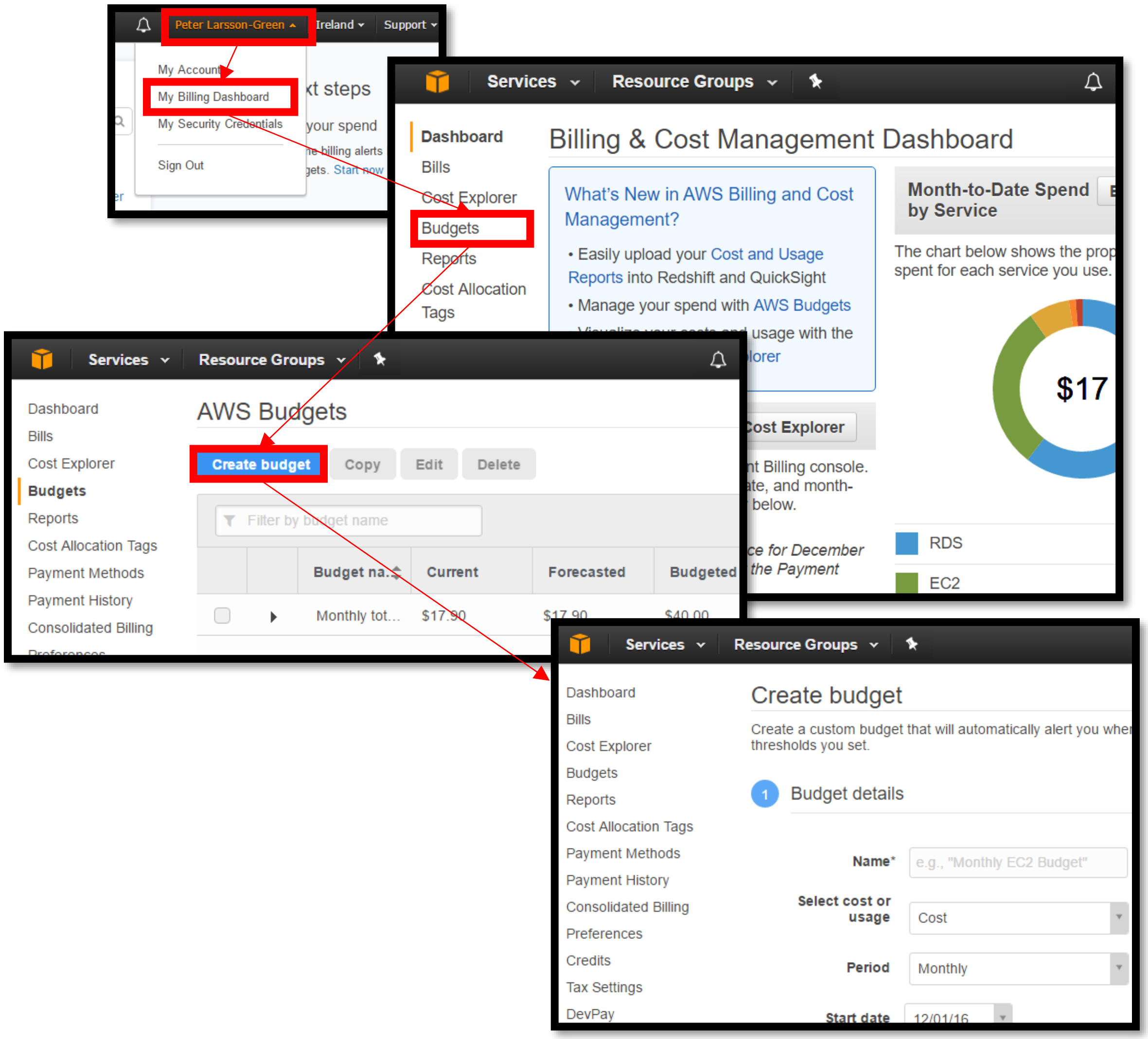


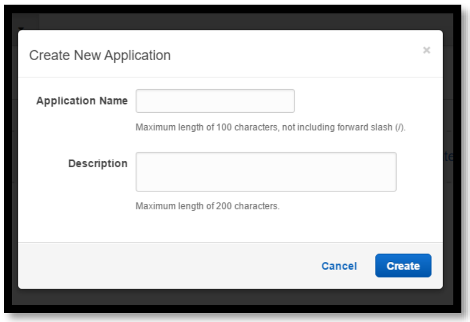
Figure 2, After moving the mouse to your name in the upper right corner, click on My Billing Dashboard, and then on Budgets. Thereafter, click on Create budget and then fill in the information about your budget.

When you have created your budget, you should be all set to start using the different services. However, **we strongly recommend you to** check your Billing Dashboard every now and then to make sure that AWS will not charge you for your usage.

Using Elastic Beanstalk

Elastic Beanstalk is Amazon's closest option to Platform-as-a-Service. By using it, you get a platform on which you directly can run the application you have created without having to worry about the underlying techniques, such as the hardware and the operating system. When you create a new Elastic Beanstalk instance, you choose which platform you would like to use. If you need a platform not supported by Elastic Beanstalk, you can setup your own by using the service Elastic Compute Cloud, but that requires you to do much more work on your own.

Start by creating a new Elastic Beanstalk Application. Each application in turn consists of environments. The idea with environments is that you can have one environment you use to develop your application, another you use to test your application, a third you use for running it in production mode, etc. So, the next thing you need to do is to create a new Environment for your Elastic Beanstalk Application.



Create New Application

Application Name

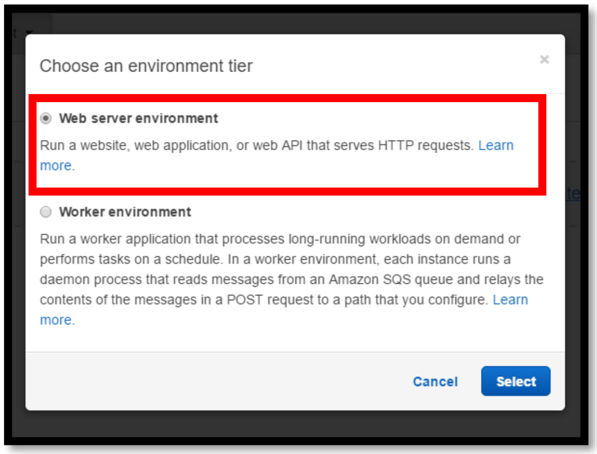
Maximum length of 100 characters, not including forward slash (/).

Description

Maximum length of 200 characters.

Figure 3, Fill in a name and description for your application.

There exist two different types of environments: *Web server environment* and *Worker environment*. Most applications will use the *Web server environment*, which is used for applications serving clients in real time (clients send requests to the server, the server handles the requests immediately and then sends back responses to the clients). A Worker environment is instead typically used for long-running operations that results in side effects, so clients don't wait for a response to be sent back to them, which means that the server doesn't need to handle the requests immediately (e.g. to send 100.000 emails).



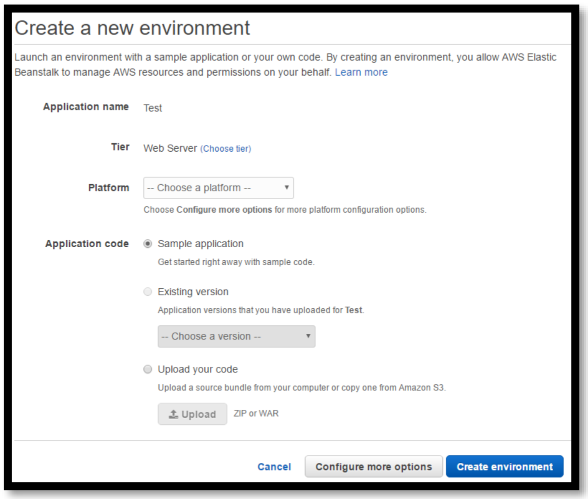
Choose an environment tier

[✓ Web server environment]

Run a website, web application, or web API that serves HTTP requests. Learn more.

Worker environment

Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. In a worker environment, each instance runs a daemon process that reads messages from an Amazon SQS queue and relays the contents of the messages in a POST request to a path that you configure. Learn more.



Create a new environment

Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. Learn more

Application name: Test

Tier: Web Server (Choose tier)

Platform: -- Choose a platform --

Choose **Configure more options** for more platform configuration options.

Application code: [✓ Sample application]

Get started right away with sample code.

Existing version

Application versions that you have uploaded for **Test**.

-- Choose a version --

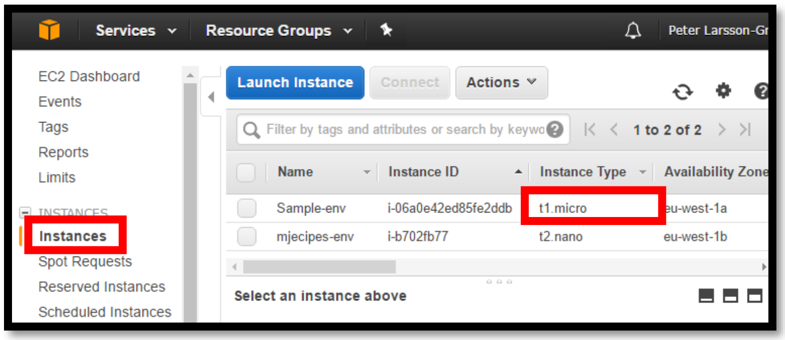
Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Figure 4, Choose Web Server environment, and then the platform you want to use.

For web applications implemented in ASP.NET, choose *.NET (Windows/IIS)* as the platform, and initially go with the *Sample application*. We will get back to how you can deploy your own ASP.NET application later. You do not need to configure any more options unless you want to.

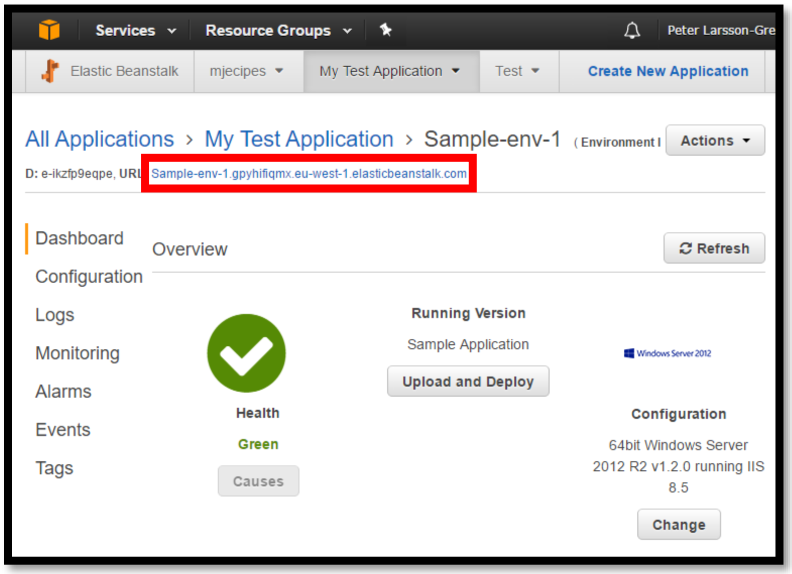
When your Elastic Beanstalk Application Environment has been created, a new Elastic Compute Cloud instance should have been created for you as well. It is this EC2 instance that will run your application. Double check that the instance is of type `t1.micro` or `t2.micro`; it is only these two that are part of the AWS Free Tier. You do this by going to the EC2 page and then click on *Instances* in the menu to the left.



	Name	Instance ID	Instance Type	Availability Zone
	Sample-env	i-06a0e42ed85fe2ddb	t1.micro	eu-west-1a
	mjecipes-env	i-b702fb77	t2.nano	eu-west-1b

Figure 5, Double check that your EC2 instance is part of the AWS Free Tier.

On the Dashboard for your Elastic Beanstalk Application Environment, you find a URL you can use to test the web application deployed to the environment (at the moment, the Sample application). Try open it in a web browser and see if it works.



D: e-ikzfp9eqpe, URL Sample-env-1.gpyhifiqmx.eu-west-1.elasticbeanstalk.com

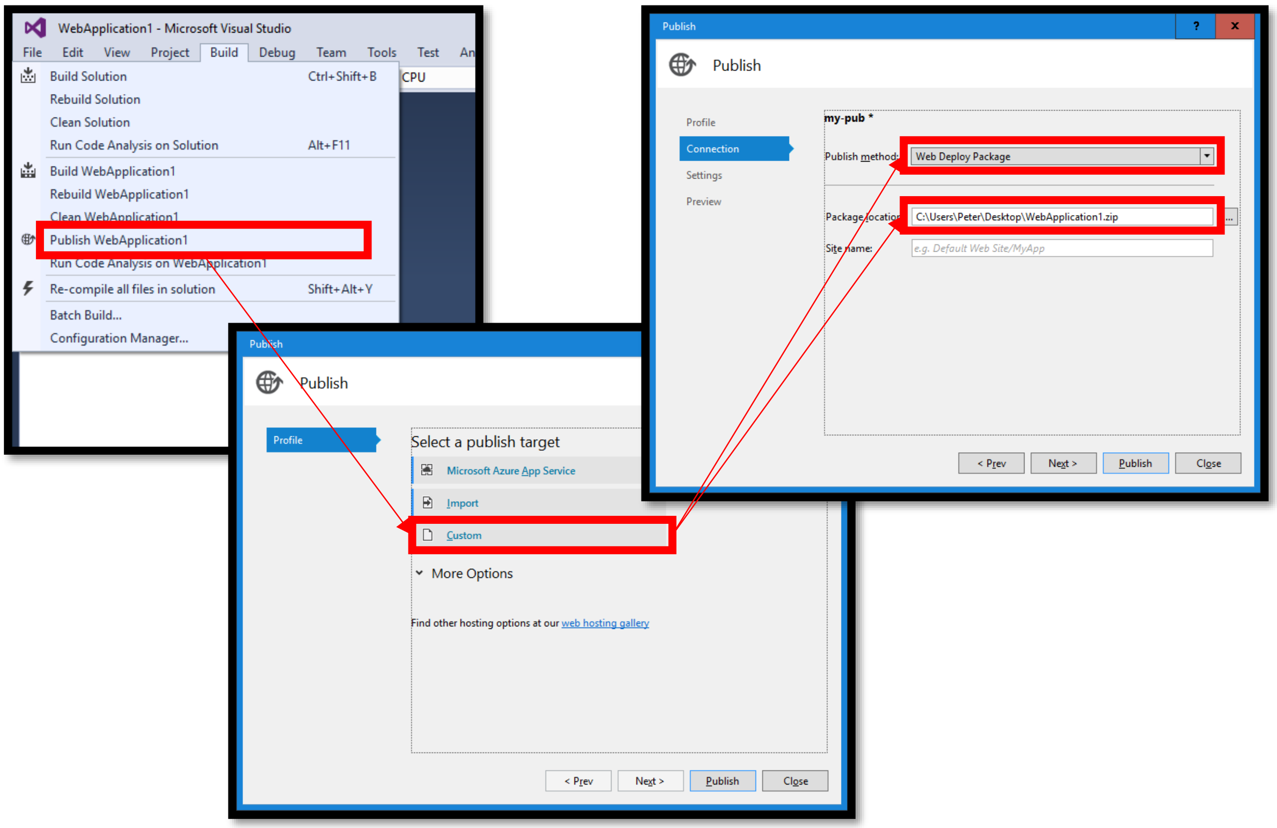
Running Version
Sample Application

Health
Green

Configuration
64bit Windows Server 2012 R2 v1.2.0 running IIS 8.5

Figure 6, Test your application by clicking on the hyperlink.

The platform .NET (Windows/IIS) will install the IIS program on your EC2 instance. This is the program on the server that listens for incoming HTTP requests, and to which you should deploy your ASP.NET application (IIS can run multiple ASP.NET applications; when it receives an incoming HTTP request it will simply forward it to the right ASP.NET application). To deploy your ASP.NET to IIS, you first need to build your ASP.NET application. That is done with the program `msBuild`. You can run this program from the Command Prompt or Windows PowerShell yourself, but a more convenient solution is to run it from Visual Studio, as shown in Figure 7 below.



my-pub *

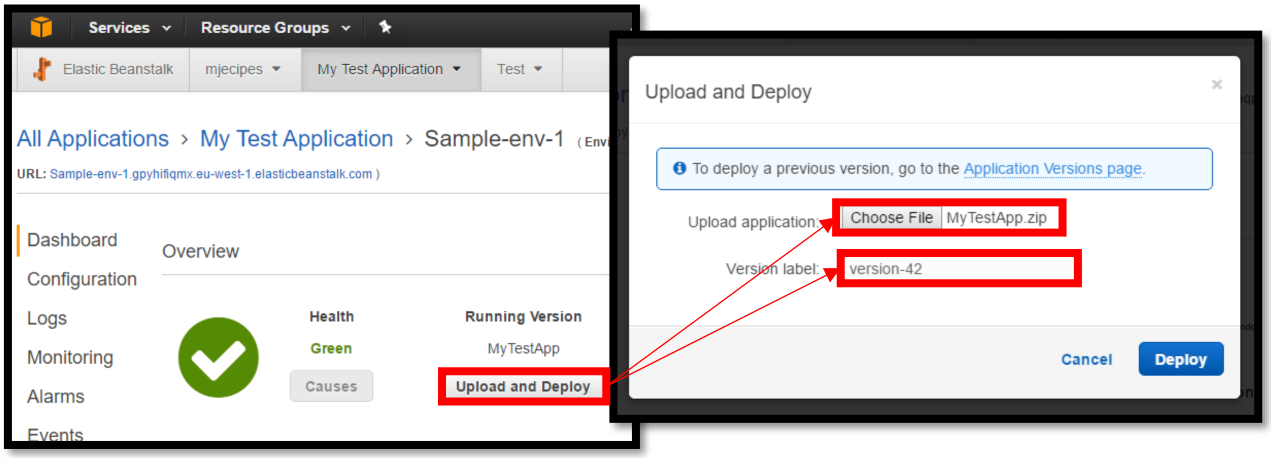
Publish method: Web Deploy Package

Package location: C:\Users\Peter\Desktop\WebApplication1.zip

Site name: e.g. Default Web Site/MyApp

Figure 7, Prepare your application to be uploaded to an IIS server by building (publishing) your application in Visual Studio.

When you publish your ASP.NET application from Visual Studio you end up with a ZIP file containing the build of your application. Simply upload this ZIP file to your Elastic Beanstalk Application Environment.



URL: Sample-env-1.gpyhifiqmx.eu-west-1.elasticbeanstalk.com)

To deploy a previous version, go to the Application Versions page.

Upload application: Choose File MyTestApp.zip

Version label: version-42

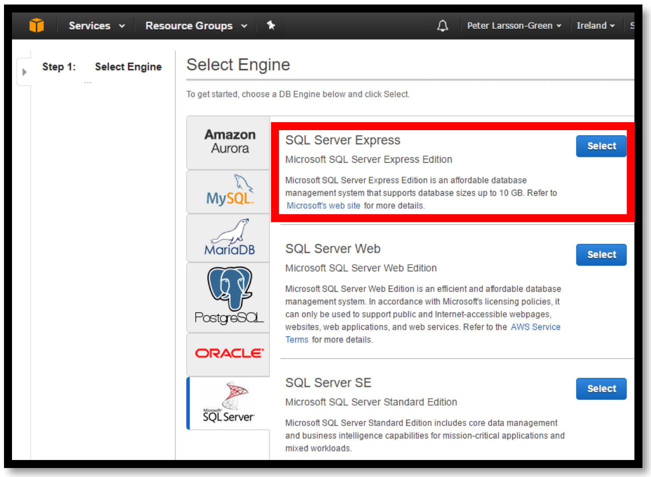
Figure 8, Upload your build (ZIP file) to your Elastic Beanstalk Application Environment instance.

It is also possible to publish multiple different ASP.NET applications (as well as ASP.NET Core applications) to a single IIS server, but this includes putting the build-ZIP files for all ASP.NET applications in a new ZIP file, also containing a manifest file. If you are curious about this, you can read more about it at http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/dotnet-manifest.html.

Using Relational Database Service

Relational Database Service (RDS) is Amazon's service that hosts relational databases, such as MySQL and SQL Server. When you use it, you simply get a virtual computer running a database for you, and to which you can connect from any computer you want.

Start by creating a new DB Instance. When you are asked to select an engine, make sure you select one that is part of the AWS Free Tier.



Select Engine

To get started, choose a DB Engine below and click Select.

SQL Server Express

Microsoft SQL Server Express Edition

Microsoft SQL Server Express Edition is an affordable database management system that supports database sizes up to 10 GB. Refer to Microsoft's web site for more details.

SQL Server Web

Microsoft SQL Server Web Edition

Microsoft SQL Server Web Edition is an efficient and affordable database management system. In accordance with Microsoft's licensing policies, it can only be used to support public and Internet-accessible webpages, websites, web applications, and web services. Refer to the AWS Service Terms for more details.

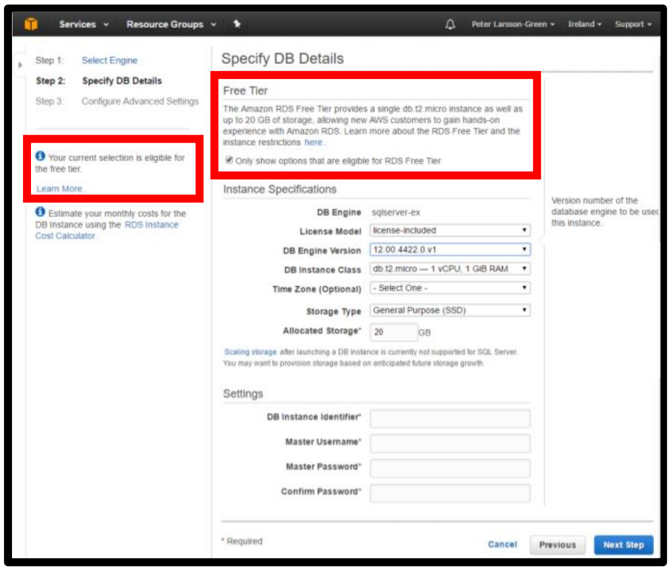
SQL Server SE

Microsoft SQL Server Standard Edition

Microsoft SQL Server Standard Edition includes core data management and business intelligence capabilities for mission-critical applications and mixed workloads.

Figure 9, When creating the database, make sure you choose a database part of the Free Tier.

When specifying the details about your database, make sure you use a configuration that is part of the AWS Free Tier. At the top, you can check a checkbox which will prevent you from using a configuration that is not part of the AWS Free Tier. To the left, you can also double check that your configuration is part of the AWS Free Tier.



Specify DB Details

Your current selection is eligible for the free tier.

Learn More

Estimate your monthly costs for the DB Instance using the RDS Instance Cost Calculator.

Free Tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions here.

[✓] Only show options that are eligible for RDS Free Tier

Instance Specifications

DB Engine	sqlserver-ex
License Model	license-included
DB Engine Version	12.00.4422.0.v1
DB Instance Class	db.t2.micro — 1 vCPU, 1 GiB RAM
Time Zone (Optional)	- Select One -
Storage Type	General Purpose (SSD)
Allocated Storage*	20 GB

Version number of the database engine to be used this instance.

Scaling storage after launching a DB Instance is currently not supported for SQL Server. You may want to provision storage based on anticipated future storage growth.

Settings

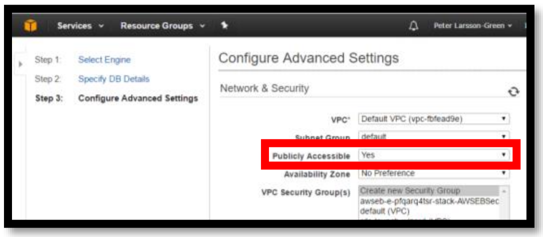
DB Instance Identifier*	
Master Username*	
Master Password*	
Confirm Password*	

* Required

Figure 10, Make sure to use a configuration part of the Free Tier.

Remember the *DB Instance Identifier*, *Master Username* and *Master Password* you use. You will use these again later when you connect to your database (e.g. from Visual Studio or from your application in Elastic Beanstalk).

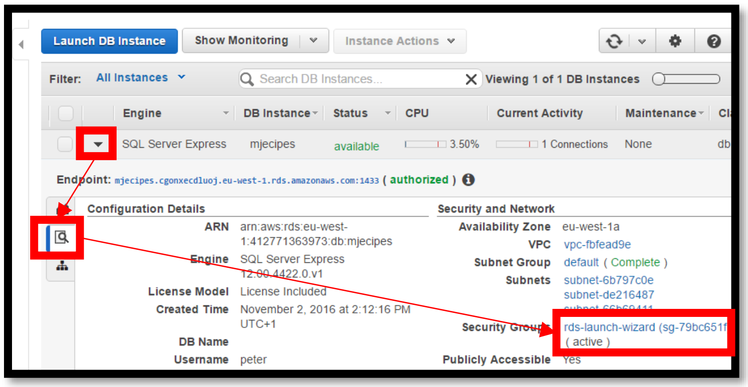
In the next step, make sure your Database Instance is publicly accessible. If it is not, you will not be able to connect to it over the internet (e.g. from Visual Studio on your own computer).


Services
Resource Groups
Peter Larsson-Green
Step 1: Select Engine
Step 2: Specify DB Details
Step 3: Configure Advanced Settings
Configure Advanced Settings
Network & Security

VPC*	Default VPC (vpc-fbfead9e)
Subnet Group	default
Publicly Accessible	Yes
Availability Zone	No Preference
VPC Security Group(s)	Create new Security Group awseb-e-pfqarq4tsr-stack-AWSEBSec default (VPC)

Figure 11, Make sure your Database Instance is publicly accessible.

Although you make your Database Instance publicly accessible, only computers with an IP address on a white-list can connect to it. The white-list is part of a security group assigned to your Database Instance. So, to change the white-list, you need to change the configuration for the security group.



Engine	DB Instance	Status	CPU	Current Activity	Maintenance	Cl
SQL Server Express	mjecipes	available	3.50%	1 Connections	None	db

Endpoint: mjecipes.cgonxecdluoj.eu-west-1.rds.amazonaws.com:1433 (authorized)

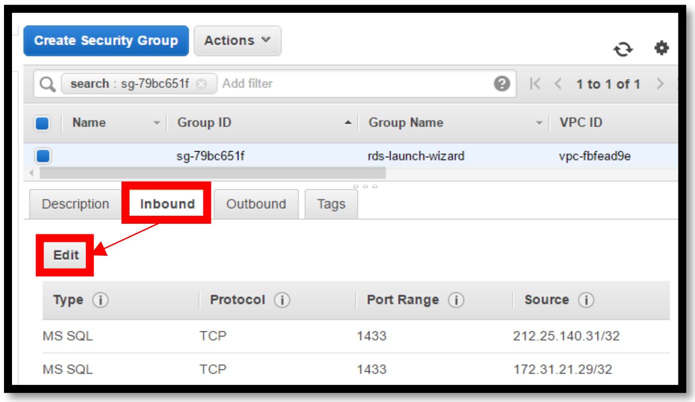
Configuration Details

ARN	arn:aws:rds:eu-west-1:412771363973:db:mjecipes
Engine	SQL Server Express 12.00.4422.0.v1
License Model	License Included
Created Time	November 2, 2016 at 2:12:16 PM UTC+1
DB Name	
Username	peter

Security and Network

Availability Zone	eu-west-1a
VPC	vpc-fbfead9e
Subnet Group	default (Complete)
Subnets	subnet-6b797c0e subnet-de216487
Security Groups	rds-launch-wizard (sg-79bc651f (active)
Publicly Accessible	Yes

Figure 12, For your Database Instance, click on the security group to modify it.



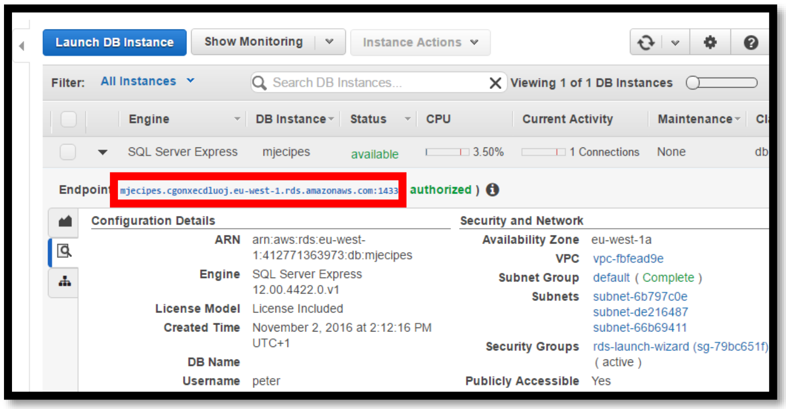
Name	Group ID	Group Name	VPC ID
	sg-79bc651f	rds-launch-wizard	vpc-fbfead9e

Type	Protocol	Port Range	Source
MS SQL	TCP	1433	212.25.140.31/32
MS SQL	TCP	1433	172.31.21.29/32

Figure 13, The Inbound Rules determine which IP addresses that may connect to the database. Click on Edit to add more IP addresses.

The IP address for the computer that was used to create the database through the AWS Management Console (your own computer) is already added. However, each time your computer connects to a Wi-Fi Network (such as eduroam) it will most likely get a new IP address, so you may not be able to connect to it from your computer in the future unless you modify these Inbound Rules.

To connect to your database, you need to have the connection string to it. The easiest way to retrieve it is by using Visual Studio. Start by locating the endpoint (URL) to your database in the AWS Console.



	Engine	DB Instance	Status	CPU	Current Activity	Maintenance
	SQL Server Express	mjecipes	available	3.50%	1 Connections	None

Endpoint mjecipes.cgonxecdluoj.eu-west-1.rds.amazonaws.com:1433 authorized)

Configuration Details

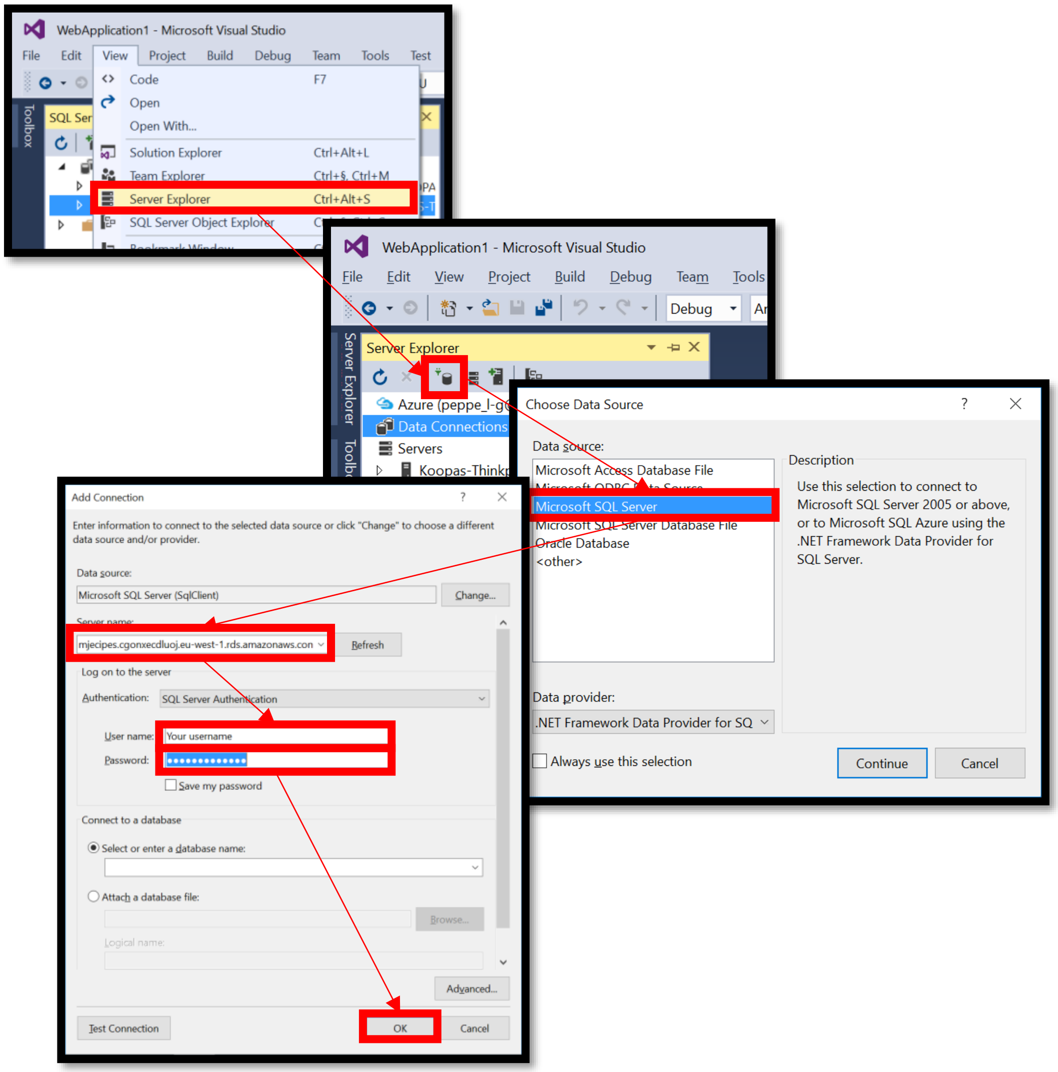
ARN	arn:aws:rds:eu-west-1:412771363973:db:mjecipes
Engine	SQL Server Express 12.00.4422.0.v1
License Model	License Included
Created Time	November 2, 2016 at 2:12:16 PM UTC+1
DB Name	
Username	peter

Security and Network

Availability Zone	eu-west-1a
VPC	vpc-fbfead9e
Subnet Group	default (Complete)
Subnets	subnet-6b797c0e subnet-de216487 subnet-66b69411
Security Groups	rds-launch-wizard (sg-79bc651f) (active)
Publicly Accessible	Yes

Figure 14, The endpoint to your Database Instance.

Then open Server Explorer in Visual Studio and connect to your database by entering your endpoint (without the port number at the end) and the credentials you entered when you created the database.



Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.

Data provider: .NET Framework Data Provider for SQ

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: mjecipes.cgonxecdluoj.eu-west-1.rds.amazonaws.con

Authentication: SQL Server Authentication

User name: Your username

Password: ••••••••••••

Figure 15, How to connect to your database instance.

When you have connected to the database, Visual Studio can show you the connection string it used to connect to it.

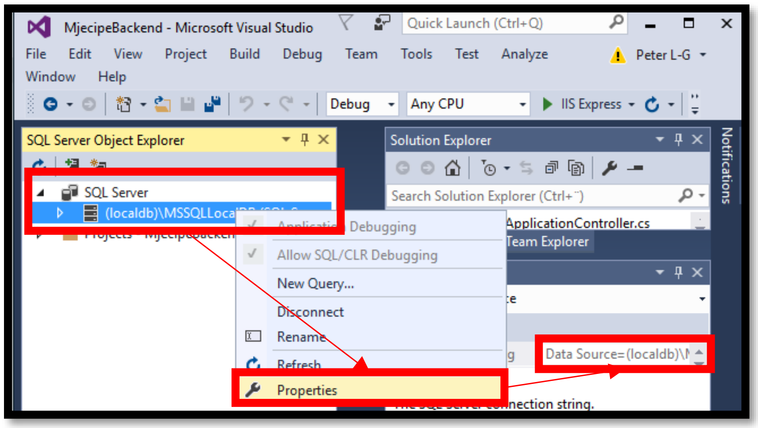


Figure 16, How to find the connection string to your Database Instance.

Note: When you copy the connection string, the password in it has been replaced with ******. Before you use this connection string, you need to replace these stars with your actual password. You may also need to add the property `Initial Catalog=`**`your-database-name`** to the connection string, specifying the name of the database in your Database Instance to use (properties in the connection string are separated by semi colon).

Encrypting the communication

HTTP sends the requests and responses in plain text. This means that the owners of the routers which your HTTP requests/responses pass through (e.g. the owner of a coffee shop whose free Wi-Fi you are using) can spy on you. To prevent this, the traffic to and from your web application needs to be encrypted. This can be done using HTTPS.

When HTTPS is used, most web browsers indicate this with some extra text/icon, usually near the address bar. In Figure 17 below, you can see what it looks like in Google Chrome.

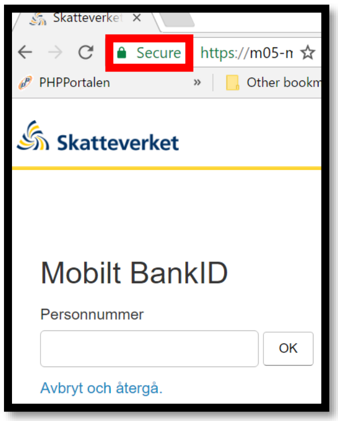
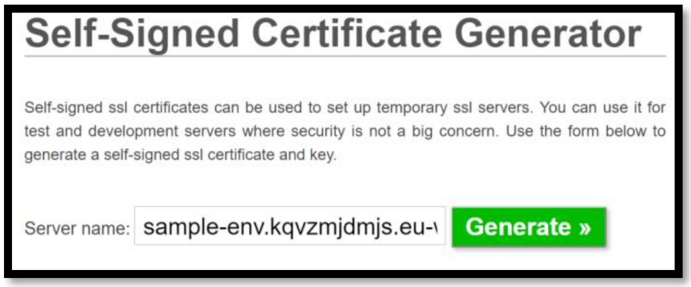


Figure 17, Google Chrome indicates that the web site is using HTTPS.

To use HTTPS on your own web site, you first need to create a certificate. This can be done online at http://www.selfsignedcertificate.com. When you create it, you will receive two files; one file containing the private encryption key the server will use, and one file containing the certificate itself, which your server will send to the clients the first time they try to send a request to your server.



Self-Signed Certificate Generator

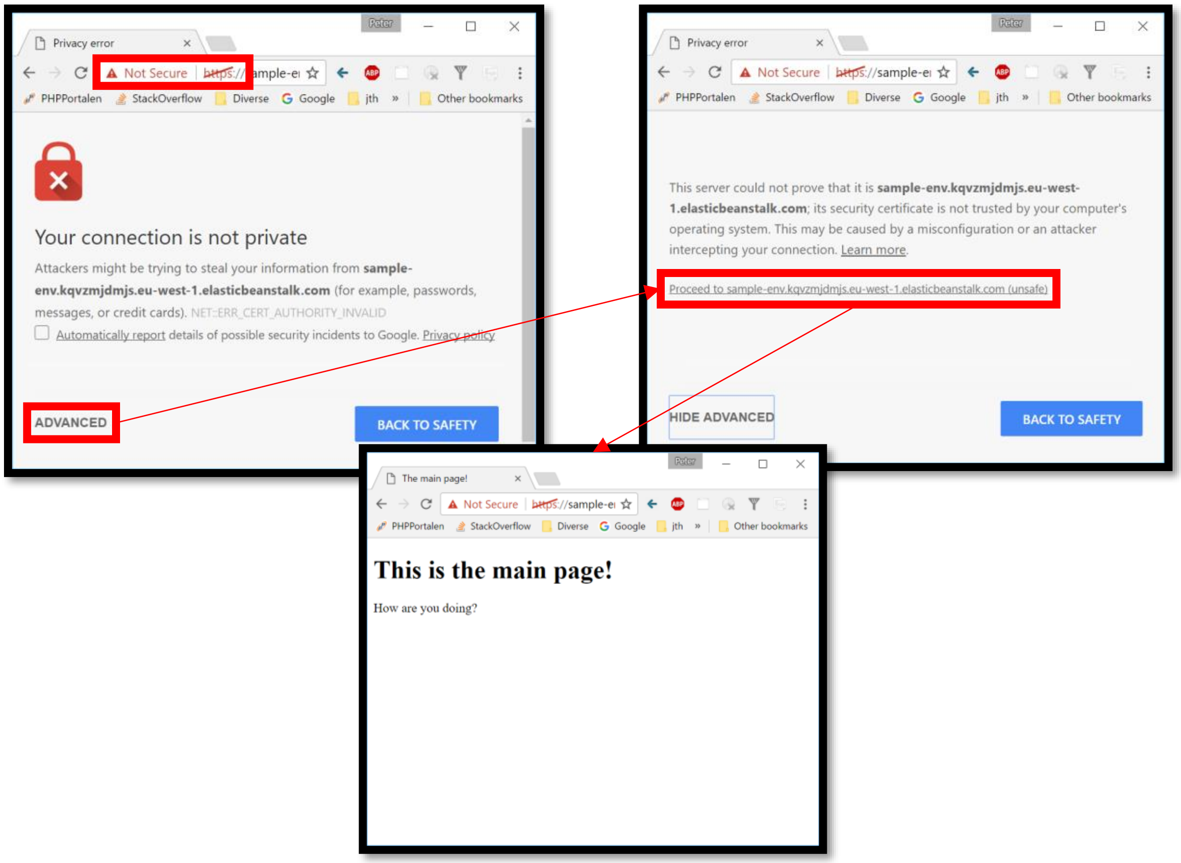
Self-signed ssl certificates can be used to set up temporary ssl servers. You can use it for test and development servers where security is not a big concern. Use the form below to generate a self-signed ssl certificate and key.

Server name: sample-env.kqvzmjdmjs.eu-\ Generate »

Figure 18, When you create a new self-signed certificate at http://www.selfsignedcertificate.com, you need to enter your own domain name.

The certificate itself consists of the public encryption key, as well as a list of all the certificate authorities that have signed the certificate. In this case, you receive a self-signed certificate, which means that it has only been signed by the certificate itself. Since no certificate authority which the clients trust has signed your certificate, no client will actually trust this certificate. This makes it, of course, insecure for the clients to use, but as soon as you get it signed by a certificate authority, your clients will start to trust it.

Getting a certificate signed by a trusted certificate authority usually costs money, so we will not do that in this course. A consequence of not getting it signed by a trusted certificate authority is that web browsers can not trust the certificate, and advice the user to not visit your website (it will still be just as secure as HTTP).



Your connection is not private

Attackers might be trying to steal your information from **sample-env.kqvzmjdmjs.eu-west-1.elasticbeanstalk.com** (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID

Automatically report details of possible security incidents to Google. Privacy policy

This server could not prove that it is **sample-env.kqvzmjdmjs.eu-west-1.elasticbeanstalk.com**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection. Learn more.

Proceed to sample-env.kqvzmjdmjs.eu-west-1.elasticbeanstalk.com (unsafe)

This is the main page!

How are you doing?

Figure 19, Even though this web site uses HTTPS, Google Chrome will advise the user to not visit your website, since the certificate hasn't been signed by a trusted certificate authority.

Note: One should of course not use certificates created by web services, such as http://www.selfsignedcertificate.com. The web service that created the certificate will know the private encryption key, so if you use this certificate, that web service will be able to decrypt the traffic to/from your web application. Furthermore, the `.key` and `.cert` files were downloaded over HTTP, so anyone that can monitor the routers the response was sent through can read them as well.

When using Elastic Beanstalk, there exists two different ways to enable HTTPS:

1. Install the certificate on your EC2 instance (your server).
2. If you use a load balancer, install the certificate on the load balancer.

The easiest way is to install the certificate is on your load balancer. This procedure is also the same no matter which programming language/framework your web applications in implemented in. However, the usage of a load balancer usually exceeds the usage limits for the free tier, so we recommend you to go with the first approach.

Way 1: Installing the certificate on your EC2 instance

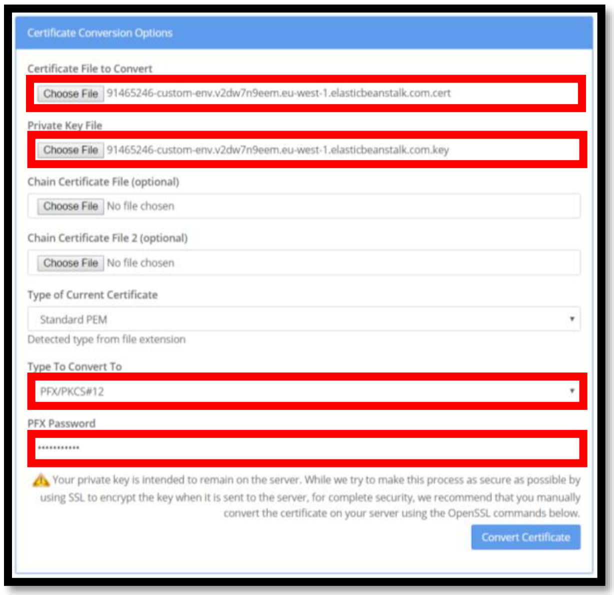
This procedure differs from framework to framework, but here we describe how you can install it for ASP.NET applications.

On your EC2 instance, a program called Internet Information Services (IIS) will run and receive all incoming HTTP/HTTPS request. When you upload a new version of your ASP.NET application to Elastic Beanstalk, Elastic Beanstalk will simply install your ASP.NET application on the IIS program on your EC2 instance.

The IIS program can run multiple ASP.NET applications at the same time, and each time it receives an incoming HTTP request, it will simply forward it to the right ASP.NET application. However, if it receives an incoming HTTPS request, it first needs to decrypt the request to know which ASP.NET application to forward it to. This result in two consequences:

1. Your certificate needs to be installed on the IIS program (not your ASP.NET application).
2. Your ASP.NET application will handle HTTP and HTTPS request the same way (the IIS program will decrypt HTTPS requests into HTTP request for you).

When you install your certificate on your IIS program, it needs to be provided in `.pfx` format. You can use the internet service https://www.sslshopper.com/ssl-converter.html to convert it into this format, as shown in Figure 20 below. A `.pfx` file contains the information from both the `.cert` file and the `.key` file, and it is also encrypted using a password.


Certificate Conversion Options

Certificate File to Convert: Choose File 91465246-custom-env.v2dw7n9eem.eu-west-1.elasticbeanstalk.com.cert

Private Key File: Choose File 91465246-custom-env.v2dw7n9eem.eu-west-1.elasticbeanstalk.com.key

Chain Certificate File (optional): Choose File No file chosen

Chain Certificate File 2 (optional): Choose File No file chosen

Type of Current Certificate: Standard PEM

Detected type from file extension

Type To Convert To: PFX/PKCS#12

PFX Password: ••••••••••••

Your private key is intended to remain on the server. While we try to make this process as secure as possible by using SSL to encrypt the key when it is sent to the server, for complete security, we recommend that you manually convert the certificate on your server using the OpenSSL commands below.

Figure 20, You can convert between different certificate formats at https://www.sslshopper.com/ssl-converter.html.

If you open the right network port on your EC2 instance and have a key pair you can use to connect to it, you can remotely control the IIS program running on your EC2 instance from another Windows computer. This way, you can install your certificate using a graphical user interface.

However, Elastic Beanstalk has a feature called `.ebextensions`. This feature allows you to configure your EC2 instance each time you upload a new version of your ASP.NET application to it by writing configuration commands in a `.config` file in your project. To use it, simply create a folder named `.ebextensions` in your project folder and put a file with the extension `.config` in it. More information about `.ebextensions` can be found at http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html.

To use `.ebextensions` to install your certificate on the IIS program on your EC2 instance, you need to:

1. Create the `.ebextension` folder in your project root folder.
2. Put your `.pfx` file in your `.ebextension` folder. Name it `certificate.pfx`.
3. Put the following `install-certificate.ps1` file in your `.ebextension` folder:

```
import-module webadministration

# Change to your password for the certificate.pfx file.
$password = "YOUR PASSWORD"

$certfile = ".ebextensions\certificate.pfx"

mv $certfile "C:\certification.pfx"

# Cleanup existing binding.
if ( Get-WebBinding "Default Web Site" -Port 443 ) {
         Echo "Removing WebBinding"
         Remove-WebBinding -Name "Default Web Site" -BindingInformation *:443:
}
if ( Get-Item -path IIS:\SslBindings\0.0.0.0!443 ) {
         Echo "Deregistering WebBinding from IIS"
         Remove-Item -path IIS:\SslBindings\0.0.0.0!443
}

# Install certificate.
Echo "Installing cert..."
$securepwd = ConvertTo-SecureString -String $password -Force -AsPlainText
$cert = Import-PfxCertificate -FilePath "C:\certification.pfx" cert:\localMachine\my -Password $securepwd

# Create site binding.
Echo "Creating and registering WebBinding"
New-WebBinding -Name "Default Web Site" -IP "*" -Port 443 -Protocol https
New-Item -path IIS:\SslBindings\0.0.0.0!443 -value $cert -Force

## (optional) Remove the HTTP binding - uncomment the following line to unbind port 80.
# Remove-WebBinding -Name "Default Web Site" -BindingInformation *:80:

# Update firewall.
netsh advfirewall firewall add rule name="Open port 443" protocol=TCP localport=443 action=allow dir=OUT
```

   In this file, you also need to change `$password` on line 4 to the password for your `.pfx` file.
4. Put the following `.config` file in your `.ebextension` folder:

```
container_commands:
  00_install_ssl:
    command: powershell -NoProfile -ExecutionPolicy Bypass -file .ebextensions/install-certificate.ps1
```

When you are done, your folder structure should look something like the one shown in Figure 21 below.

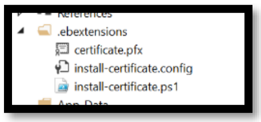
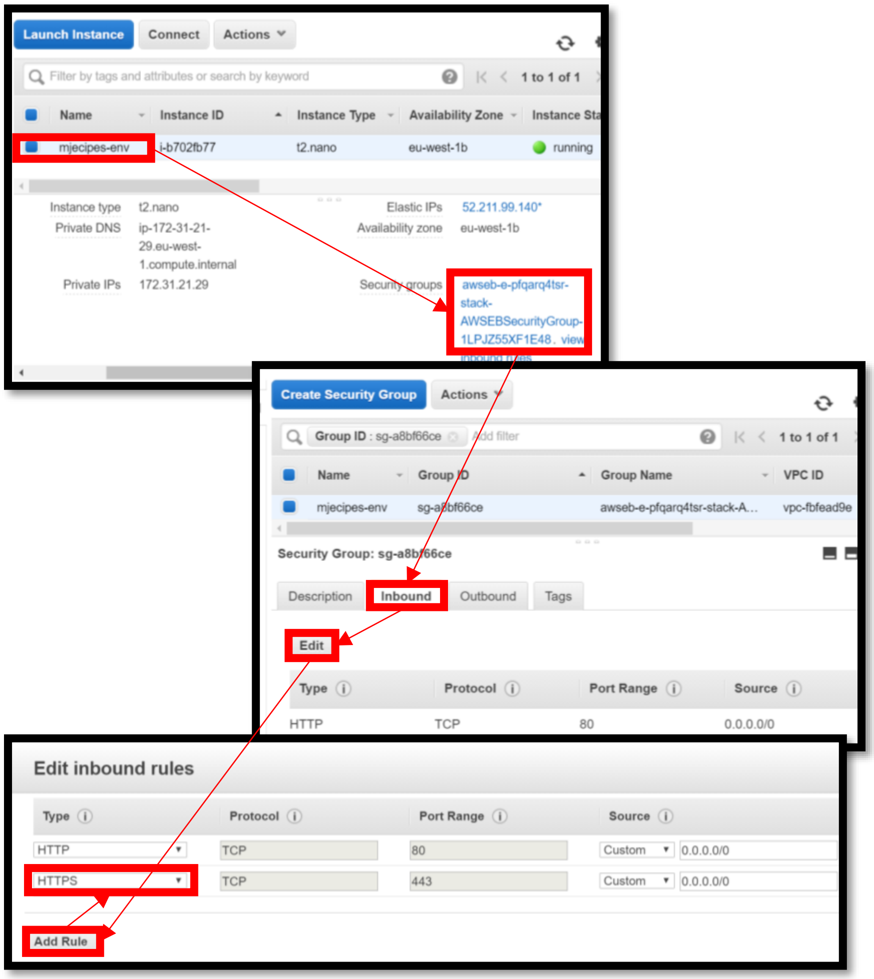


Figure 21, The files you need to add to your ASP.NET application to install your certificate.

Next time you upload a new ZIP file containing the build of your ASP.NET application, the certificate will be installed on the IIS program running on your EC2 instance. However, you cannot send HTTPS requests to your web application yet, since the port 443 is disabled on your EC2. You first need to enable it. This is done in a way similar to how you enabled your own computer to connect to your RDS instance; see Figure 22 below.



Name	Instance ID	Instance Type	Availability Zone	Instance Sta
mjecipes-env	i-b702fb77	t2.nano	eu-west-1b	running

Instance type t2.nano
Private DNS ip-172-31-21-29.eu-west-1.compute.internal
Private IPs 172.31.21.29
Elastic IPs 52.211.99.140*
Availability zone eu-west-1b
Security groups awseb-e-pfqarq4tsr-stack-AWSEBSecurityGroup-1LPJZ55XF1E48. view inbound rules

Group ID : sg-a8bf66ce

Name	Group ID	Group Name	VPC ID
mjecipes-env	sg-a8bf66ce	awseb-e-pfqarq4tsr-stack-A...	vpc-fbfead9e

Security Group: sg-a8bf66ce

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0

Edit inbound rules

Type	Protocol	Port Range	Source
HTTP	TCP	80	Custom 0.0.0.0/0
HTTPS	TCP	443	Custom 0.0.0.0/0

Figure 22, Enabling HTTPS on your EC2 instance.