# Amazon EC2 Website Setup Instructions

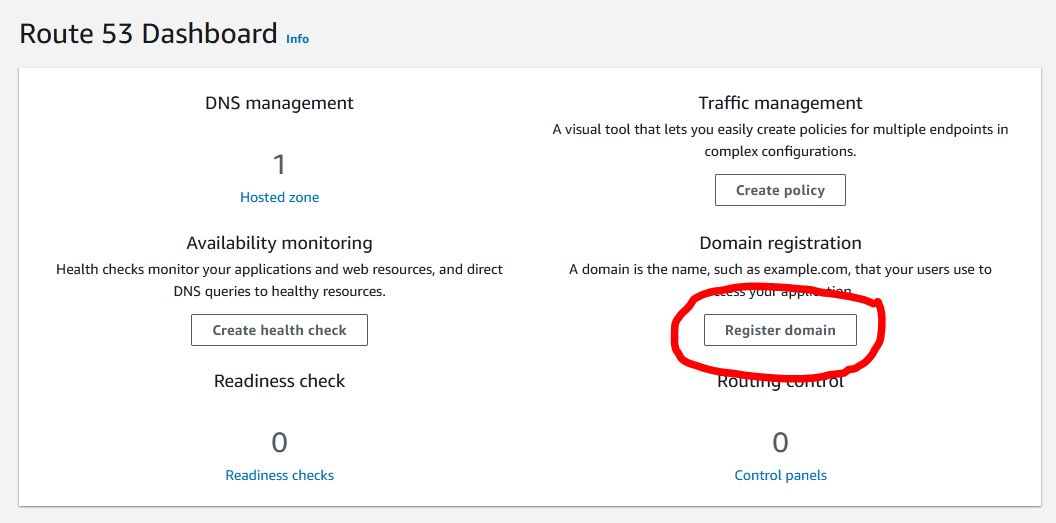
4 April 2023, Chris Bartlett

This document shows how to get a domain name and set up a basic website.

## Get a domain name

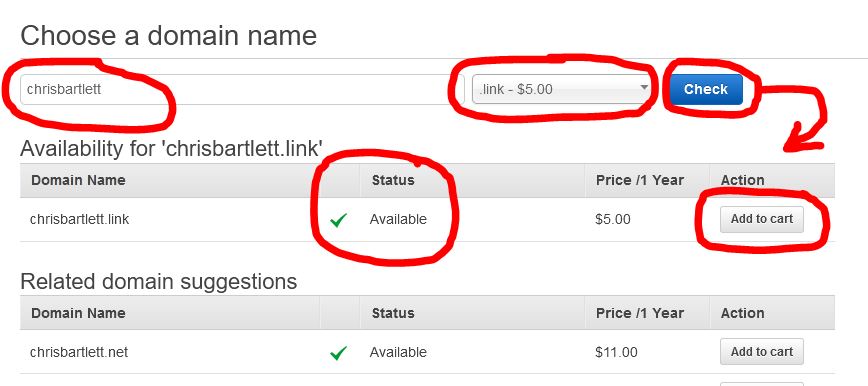
In the past we have used free domains, but the service we used now allows only one free domain. We need to be able to create a number of domains (actually subdomains) for the website projects we work on. It is possible to get a domain for US$5 (annual) from Amazon, which is what I show here.

Go to the AWS Route 53 service and click Register Domain:



For this course I require you to have your name in your domain, so that support and marking etc., is manageable. If you already have a domain and want to use that, please discuss with me.

Enter your name (lowercase, no spaces) and choose the .link domain (this is the cheapest, at US$5). Click the “Check” button to make sure the domain is available. Modify your name if it isn’t (e.g., add a middle initial and check again). Add the domain name to the cart.



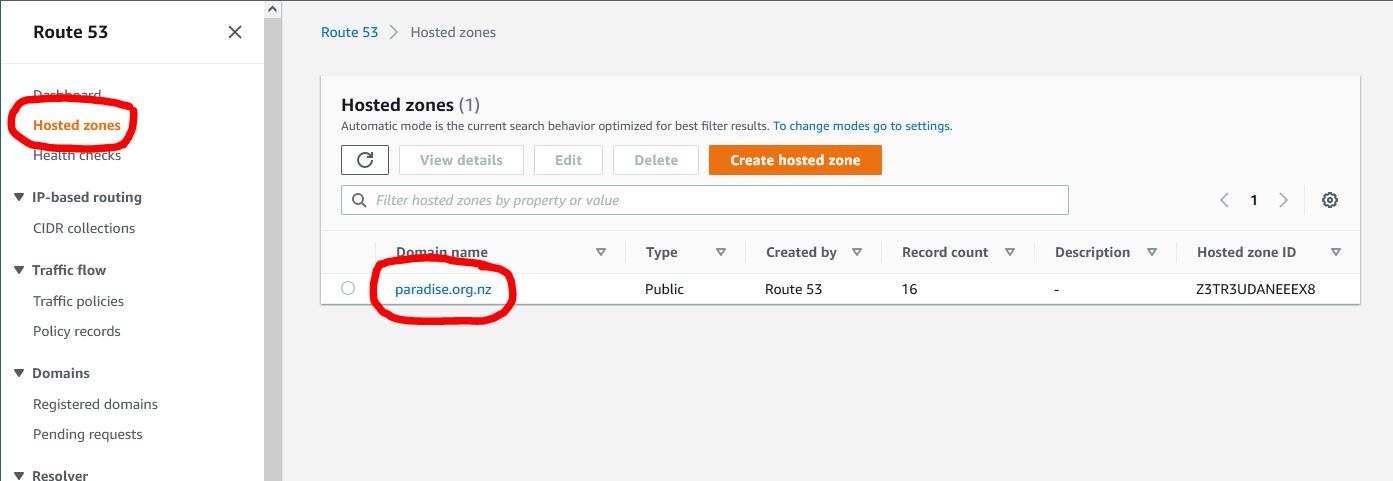
Click the Continue button (scroll down to find it) and then fill out the registrant contact details with your information.

Continue to the final screen (Verify & Purchase). You might want to choose “Disable” for the “Do you want to automatically renew your domain?” question.

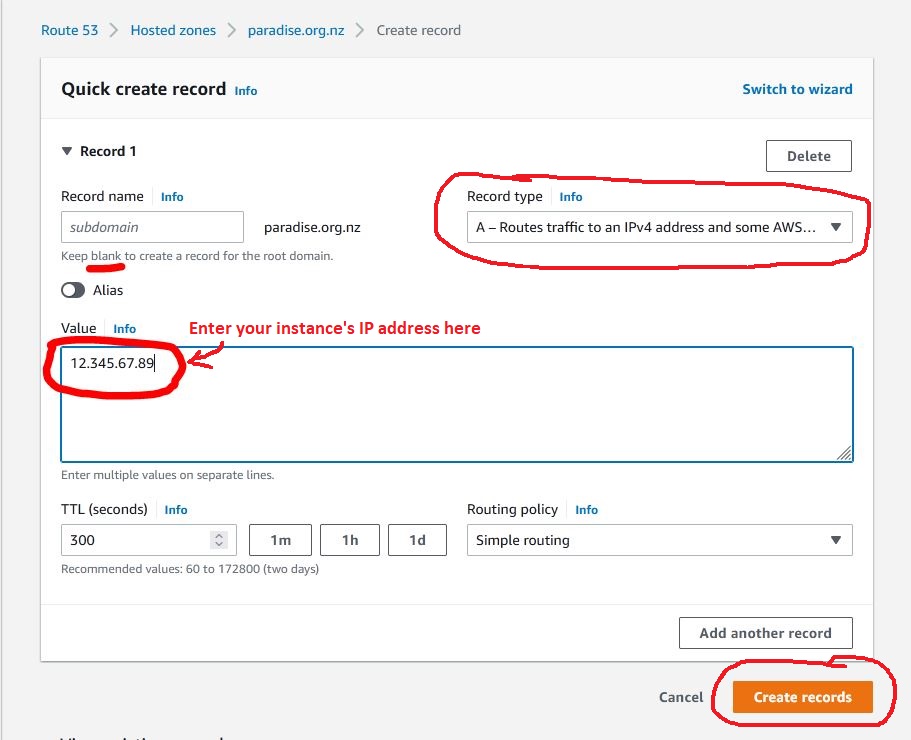
Complete the order.

## Point your domain at your instance

You have to link your new domain name with your instance’s IP address. Go to the Route53 service and click Hosted zones in the left-side menu. Then click your domain name:

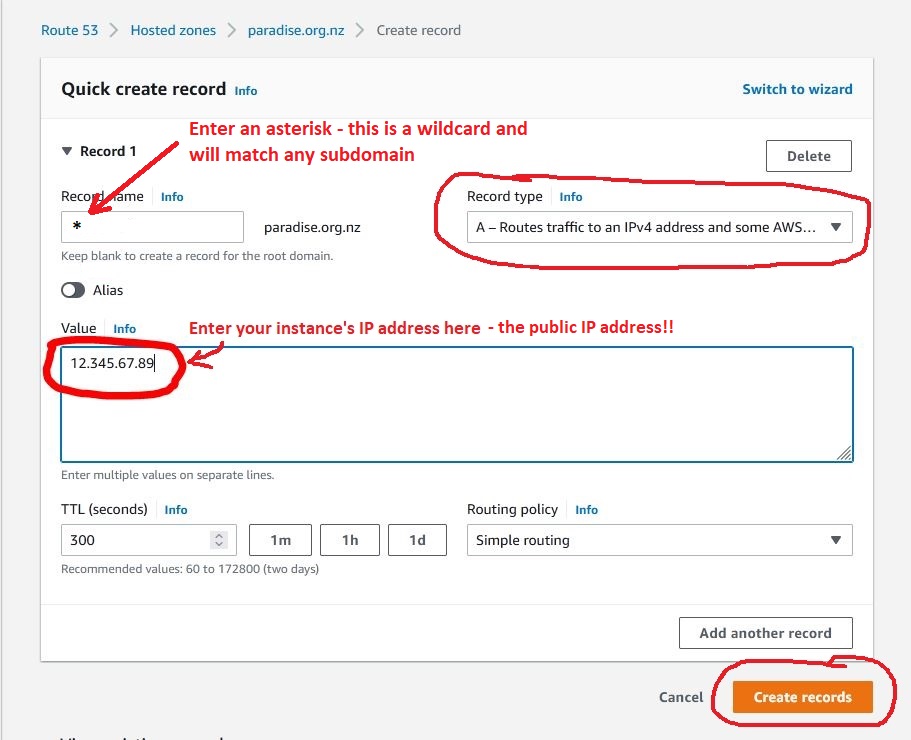


Then click the “Create record” button. On the next screen enter your IP address and click Create Records.



## Create a wildcard DNS record

The Domain Name System cannot guess that your subdomains (e.g., test.hemibrown.org) are hosted where your main domain points in the DNS. You need a wildcard A record in the DNS if you want any subdomain to point at your instance. Click “Create record” again, but this time enter \* (the asterisk character) in the Record Name field, as shown in the following screenshot. Click “Create Records” when you are done.



## Configure the webserver (Apache) to serve a website

### Introduction: suggested naming conventions

Because you are going to set up several websites, you should make use of subdomains for each one. For example, if your main domain name is hemibrown.link, then you should set up each website *with its own subdomain*; e.g., you might have three websites with the following subdomains:

test-website.hemibrown.link  
bootstrap-portfolio.hemibrown.link  
wordpress-portfolio.hemibrown.link

Each website should be in its own directory in the /var/www directory. It is advisable to name those directories to match the domain names in some way, to avoid confusion. For example, the websites for the domains above could be in these directories:

/var/www/test-website/  
/var/www/bootstrap-portfolio/  
/var/www/wordpress-portfolio/

When you create the Apache configuration files for each website (see below) you should also name them to match the website subdomains. For example, the websites for the domains above could be configured in these files:

/etc/apache2/sites-available/test-website.conf  
/etc/apache2/sites-available/bootstrap-portfolio.conf  
/etc/apache2/sites-available/wordpress-portfolio.conf

There is nothing magic about this suggested naming convention (names of files and directories do not *have* to match), but you will get very confused if they don’t!

### Configure a website

**NOTE: Where it says “your-project-name” below, substitute something you choose, following the suggested naming conventions above. Do not use spaces.**

Connect to your instance via SSH (e.g., using PuTTY).

Create a configuration file for your website, based on the default configuration file. The next command is one line:

sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/**your-project-name**.conf

Edit the website’s configuration file:

sudo nano /etc/apache2/sites-available/**your-project-name**.conf

1. Change this line from:

#ServerName [www.example.com](http://www.example.com)

To this (using your own website domain or a subdomain of it) – make sure you remove the comment character (the hash) at the start and don’t use www at the start (your domain name is not configured to use www…):

ServerName your.website.domain

1. Tell the webserver where your web files are - change this line from:

DocumentRoot /var/www/html

To this:

DocumentRoot /var/www/**your-project-name**

Exit, saving changes

Ctrl-x (answer y, then press enter)

Make a directory that you will put the website’s files into

sudo mkdir /var/www/**your-project-name**

Give the ubuntu user access to that directory

sudo chown ubuntu /var/www/**your-project-name**

Enable the website

sudo a2ensite **your-project-name**

Reload the webserver to apply the changes

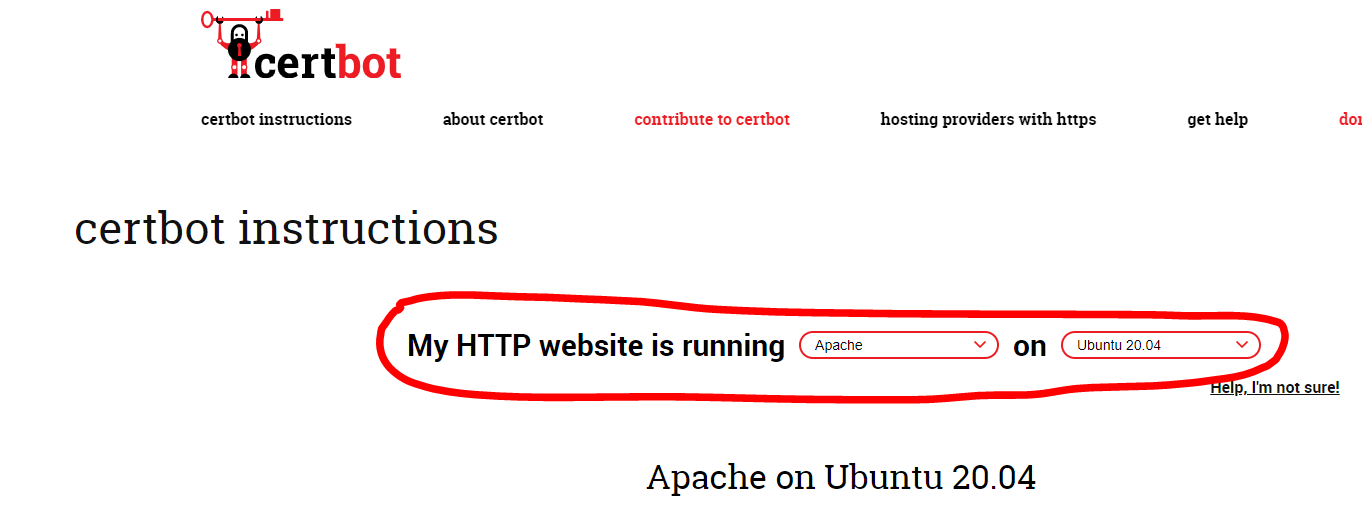
sudo systemctl reload apache2

## Create a test home page for the new web site

Create an HTML file called index.html (or get some web files, such as the case study) and upload it into the directory you created for the website above, using WinSCP (connect in the same basic way you did with PuTTY, with the ubuntu user name and the key file). Test it in your web browser.

## Set up secure serving of the website

If you have not already done this, install Certbot (you only need to do this the first time), following the instructions on the Certbot website: <https://certbot.eff.org/> Make sure you choose the right options to get the appropriate instructions, as in the image below (note that the instructions for Ubuntu 20 also work for 22, which you have installed):



Use Certbot to get a security certificate and install it (NOTE: do **not** use the certificate-only instructions). You will be required to answer a number of questions.

sudo certbot --apache

Now try it in your web browser by going to https://your.website.domain. Also check that HTTP traffic is switched to HTTPS: http://your.website.domain.