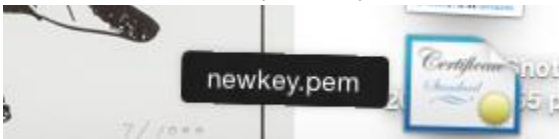


Part 1: Creating the Management Server

Use us-west-2 (US West (Oregon)) as your region.

1. Go to your ec2 dashboard and select 'Key Pairs' from the left sidebar.
2. Click 'create key pair' and name it 'newkey'. A file called 'newkey.pem' should have been downloaded; this is your keyfile.



3. Select 'Instances' from your left dashboard.
4. Click 'Launch instance'.
5. Select the top option (Amazon Linux 2 AMI (HVM), SSD Volume Type).
6. Click next until you reach 'Configure Security Group'.
7. Create a security group named 'my-task-1-security-group'.
8. You should already see a rule for SSH - TCP - 22 - Custom - 0.0.0.0/0.
9. Click 'Add rule'.
10. Select HTTP as the type. The remaining fields should be: 'TCP', '80', 'Custom', and '0.0.0.0/0, ::/0' respectively.
11. Click 'Review and Launch'.
12. Click 'Launch'.
13. Select "choose an existing key pair" and select "newkey" from your list of key pairs then tick the acknowledge box and select "Launch Instances"
14. Click 'View instances'.

Wait for the instance state to change to "running" with a green icon next to it.

Connecting via SSH

If you're working on a Windows PC, continue with the following section; if you're using a Mac, jump to 'Connecting to your EC2 via Terminal'

Connecting to your EC2 via PuTTY (for Windows users)

Visit the website to download PuTTY at
<https://www.chiark.greenend.org.uk/~sgtatham/putty/>

Click 'download it here':

PuTTY: a free SSH and Telnet client

[Home](#) | [FAQ](#) | [Feedback](#) | [Licence](#) | [Updates](#) | [Mirrors](#) | [Keys](#) | [Links](#) | [Team](#)
Download: [Stable](#) | [Snapshot](#) | [Docs](#) | [Changes](#) | [Wishlist](#)

PuTTY is a free implementation of SSH and Telnet for Windows and Unix platforms, along with an `xterm` terminal emulator. It is written and maintained primarily by [Simon Tatham](#).
The latest version is 0.70. [Download it here](#).

LEGAL WARNING: Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. We believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England, but we are not lawyers, and so if in doubt you should seek legal advice before downloading it. You may find useful information at cryptolaw.org, which collects information on cryptography in various countries.

Choose either 32-bit or 64-bit installation:

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

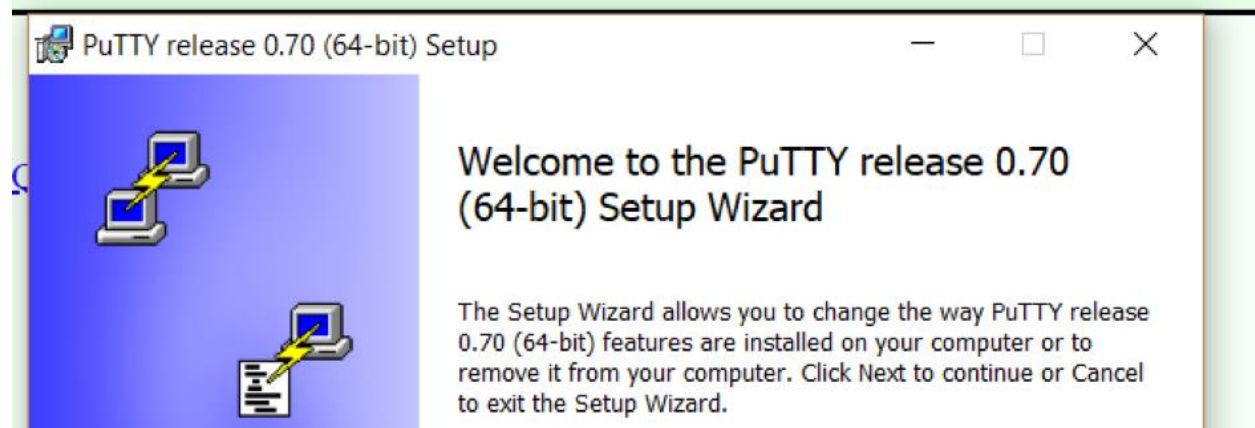
MSI ('Windows Installer')

32-bit:	putty-0.70-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.70-installer.msi	(or by FTP)	(signature)

Unix source archive


.tar.gz:	putty-0.70.tar.gz	(or by FTP)	(signature)
----------	-----------------------------------	-------------------------------	-------------------------------

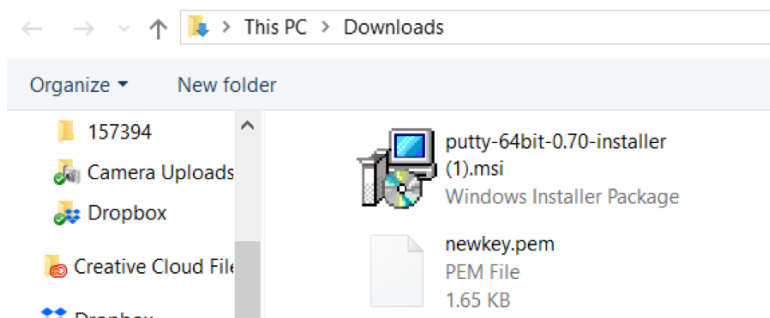
When the download is complete, run the installer and follow the wizard to complete the setup:



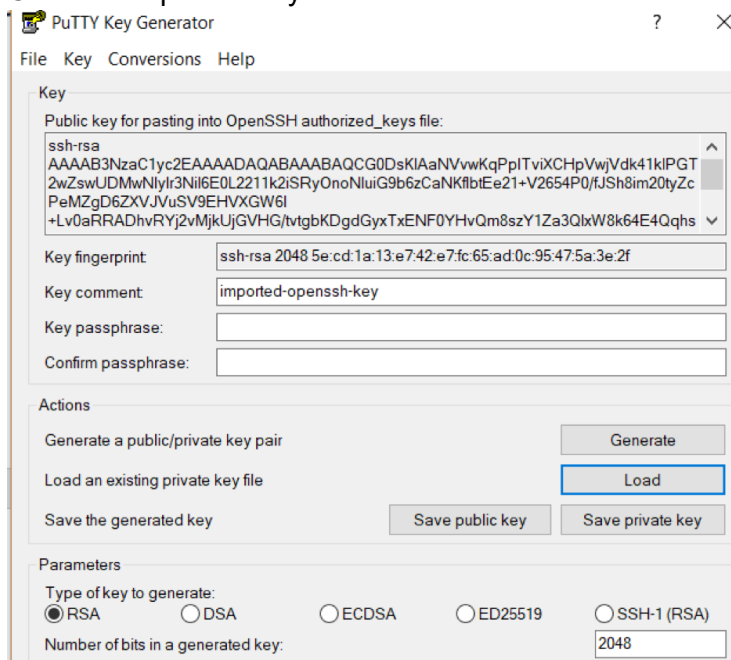
Click 'Start', search for 'PuTTYgen' and open it.

To load the 'newkey.pem' private key file, click 'Load' and browse to the key file's location. You will need to select 'View All Files' to see it:

 Load private key:



Click 'Save private key':

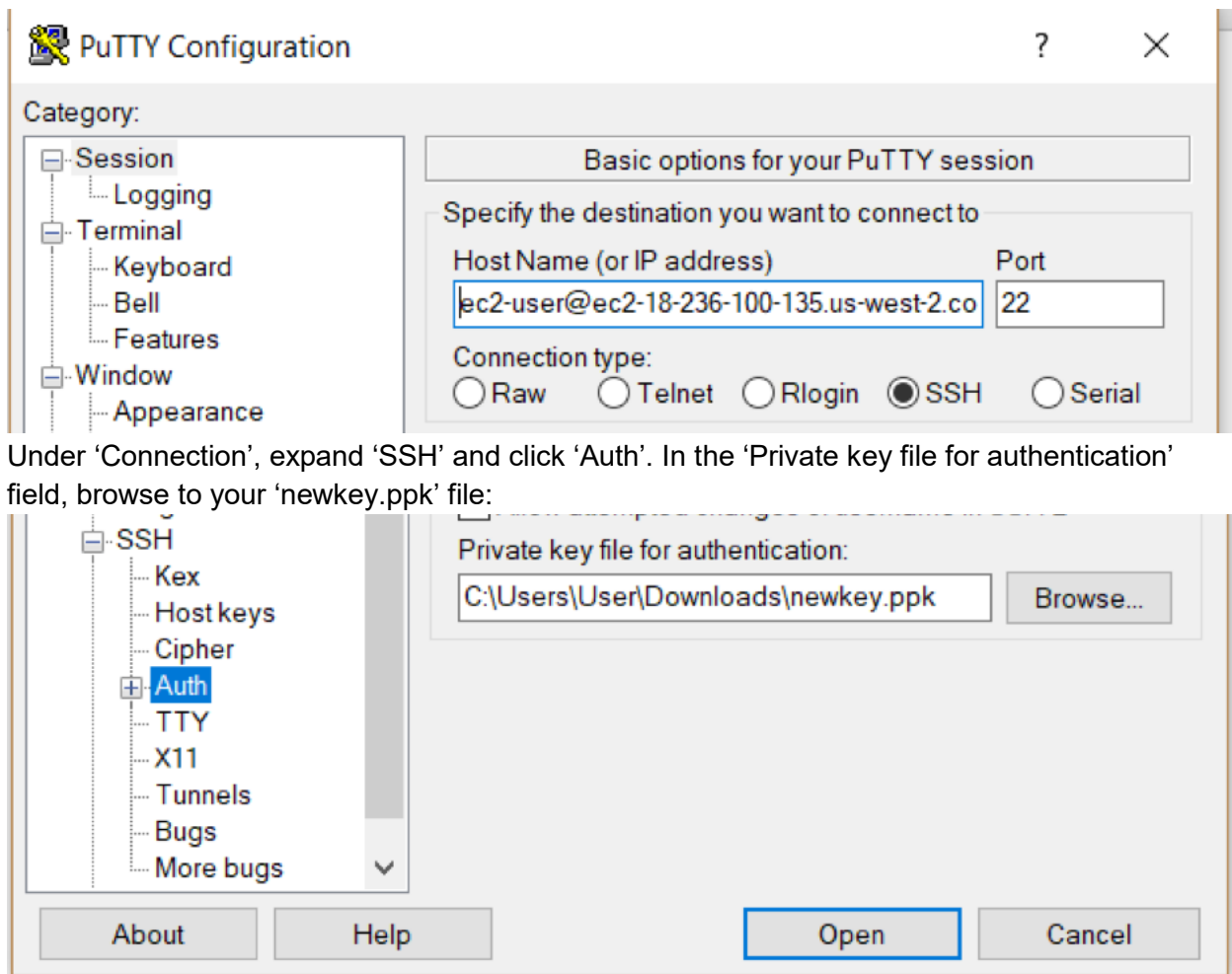


This creates a new file: 'newkey.ppk'. This is the file we use to connect to our EC2 instance:

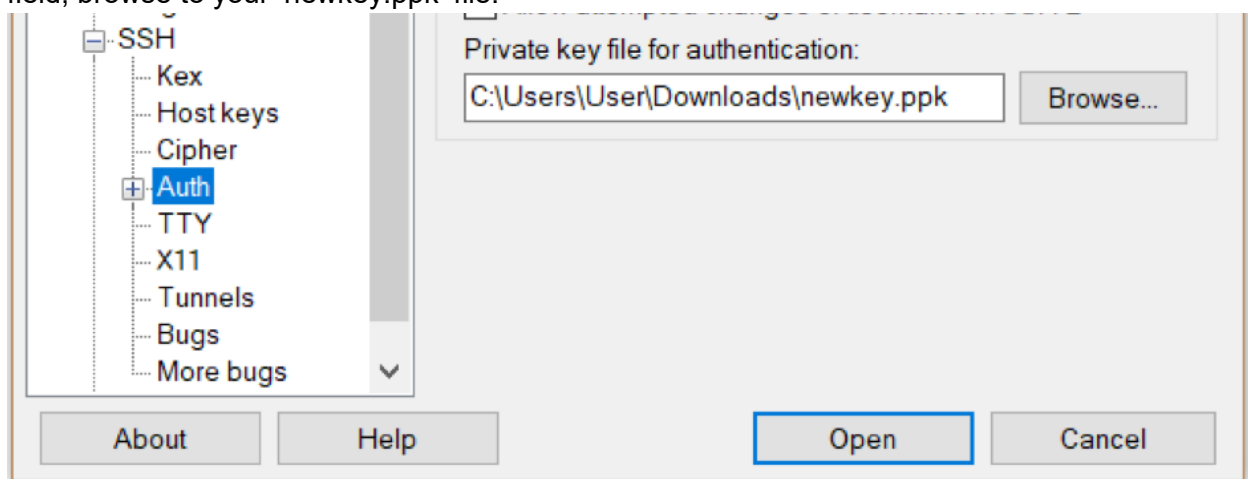
This PC > Downloads				
	Name	Date modified	Type	Size
	putty-64bit-0.70-installer (1).msi	22/08/2018 9:36 A	Windows Installer ...	2,978 KB
	newkey.ppk	21/08/2018 12:17	PuTTY Private Key ...	2 KB

Click Start and open PuTTY.

In the Host Name field, enter `ec2-user@[your EC2 instance's public DNS]`:



Under 'Connection', expand 'SSH' and click 'Auth'. In the 'Private key file for authentication' field, browse to your 'newkey.ppk' file:



Click 'Open' and you'll be connected to your EC2:

```
ec2-user@ip-172-31-29-250:~  
Using username "ec2-user".  
Authenticating with public key "imported-openssh-key"  
Last login: Tue Aug 21 08:18:47 2018 from 110.173.190.52  
  
 _ _ _ _ _  
 _ _ _ _ _ / Amazon Linux 2 AMI  
 _ _ _ _ _  
  
https://aws.amazon.com/amazon-linux-2/  
2 package(s) needed for security, out of 3 available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-29-250 ~]$
```

Continue to 'Running the Script for Task 1'.

Connecting to your EC2 via Terminal (for Mac users)

Open a new Terminal window (from Applications/Utilities/Terminal).

Go back to the AWS console, click on your new EC2 instance and click the 'Connect' button.

Note the first command in the window that appears. Type this command into the Terminal window:

3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 newkey.pem
```

Note: You may need to provide the path to the key file, for example

```
chmod 400 Downloads/newkey.pem
```

Next, enter the 'Example' command from the same window. Again, if necessary, include the path to the key so that

```
ssh -i "newkey.pem" ec2-user@ec2-54-218-234-45.us-west-2.compute.amazonaws.com
```

Would become:

```
ssh -i "Downloads/newkey.pem" ec2-user@ec2-54-218-234-45.us-west-2.compute.amazonaws.com
```

The Terminal prompt will ask you if you want to continue connecting; enter **yes**.

Running the LAMP and WordPress Install Script

Open task_1.sh in a plain text editor

In the console window, enter **nano task_1.sh**

Copy the script from the plain text editor and paste it into the console window (right-click on Windows, or Command-V on Mac)

Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

To run the script, enter **bash task_1.sh**

When the installation is complete, you should see a series of success statements:

```
Success: WordPress downloaded.
Success: Generated 'wp-config.php' file.
Success: Database created.
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100    13  100    13    0     0    13      0  0:00:01 --:--:--  0:00:01 3250
Success: WordPress installed successfully.
```

Leave this window open as we need it for Part 2

We now need to confirm that our web server and WordPress are both working correctly. To do this:

1. Go back into your AWS dashboard, select "Instances" from your left dashboard, copy your instance's IPv4 Public IP and paste it into your browser.
2. You should see the WordPress default theme. Navigate to `[your-public-ip]/wp-admin` to reach the login screen
3. Login using username "`group_1`" and password "`alpha wario salamander`" to reach the WordPress admin dashboard

Part 2: Creating Multiple EC2s and Automating Error Alerts

All of the following takes place on the server created in part 1. If you disconnected from the server, use Terminal or PuTTY to reconnect before continuing.

Moving the access key to the management server

1. Open newkey.pem (the key pair file downloaded in part 1) in notepad or another plain text editor
2. Copy the entire contents of the file from the plain text editor
3. In the console:

```
cd /var/www/html
```


Enter

```
nano newkey.pem
```
4. Paste the contents of the file into the console window. If you don't see all of it, the window may be too small - in that case, hold the up arrow to go to the top of the file.
5. Check that the file begins with
-----BEGIN RSA PRIVATE KEY-----
And ends with
-----END RSA PRIVATE KEY-----
6. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

Installing and Configuring the Required Software

1. Enter

```
nano installAnsible.sh
```
2. Open installAnsible.sh in a plain text editor and paste it into the console window
3. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

4. This script installs Ansible and other required software, and takes you straight to Ansible's config file. Enter `./installAnsible.sh` to run it.
5. When the script has finished and the config file is open, press the down arrow until your cursor is below the line '[defaults]'
6. Copy the following line and paste into ansible.cfg:
`private_key_file = /var/www/html/newkey.pem`
7. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

Moving the necessary files to the management server

Setting the necessary permissions

1. Enter `sudo chmod -R 777 /var/www/html`
2. Enter `sudo chmod 400 newkey.pem`

Creating the playbook for launching a new EC2

1. Enter `nano launchEC2.yml`
2. Open launchEC2.yml in a plain text editor and paste it into nano
3. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

Creating the playbook for checking that each EC2 is live

1. Enter `nano CheckInstances.yml`
2. Open CheckInstances.yml in a plain text editor and paste it into nano
3. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

Creating the script for checking that each EC2 is live

1. Enter `nano checkurl.sh`
2. Open checkurl.sh in a plain text editor and paste it into nano
3. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

Creating the playbook for sending an email notification

1. Enter `nano email.yml`

2. Open email.yml in a plain text editor and paste it into nano
3. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close

Creating the script for setting environment variables

Enter `nano keysAndEmail.sh`

Open keysAndEmail.sh in a plain text editor and paste it into nano

1. Add your email next to 'webmasterEmail='. Don't put a space after the equals sign.

Note re: email addresses

Our playbook sends an email on port 25, which may not get through some firewalls. If you're not sure whether your email server allows emails on port 25, use a gmail address as this has been successfully tested. The first email you receive will likely go to the spam folder, so make sure to check that after everything else is done. In a future implementation with more resources, we would use a secure mail server such as AWS WorkMail or AWS SES.

2. Return to the AWS dashboard in your browser
3. Click your name in the top right then click 'My Security Credentials'; select 'Continue to Security Credentials' on the popup
4. Expand 'Access Keys' and select 'Create new access key', then 'Download key file'
5. Open the key file you downloaded in a plain text editor. Keep this file open for use in the next stage
6. Copy and paste the access key ID and secret key from the key file into nano:

```
export webmasterEmail=webmaster@gmail.com #<-- Add your email here
export AWS_ACCESS_KEY_ID=AKIAIG50DIK7AJNN5ZDQ #<-- Paste your access key ID
export AWS_SECRET_KEY=5hL77+AR0yLoUtq4007yjcEWqwi32Vi7et1txX2R #<-- Paste your secret key
```
7. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close
8. Enter `sudo chmod 777 keysAndEmail.sh` so that it can be read, written to and executed

Set Environment Variables

This step is required to create the first new EC2 but will be automated using keysAndEmail.sh later on

1. In the console, type `export AWS_ACCESS_KEY_ID=` and paste your access key ID after the equals sign (don't put a space). Press Enter
2. Type `export AWS_SECRET_ACCESS_KEY=` and paste your secret access key. Press Enter

Creating the new server

1. Enter `ansible-playbook launchEC2.yml`

The playbook will take several minutes to run. At the end you should see this:

```
PLAY RECAP *****
54.70.116.131      : ok=20    changed=15    unreachable=0    failed=0
localhost         : ok=5     changed=3     unreachable=0    failed=0

[ec2-user@ip-172-31-25-142 html]$
```

Setting up notifications

Creating a broken link to test the email notification

Our finished system notifies us if the web server returns a 400 or 500 error, and checks each link on the homepage to ensure that it isn't broken. To test the latter, we need to add a broken link to the WordPress homepage.

1. Return to the AWS dashboard in your browser
2. Select 'Services' → 'EC2' → 'Running Instances'. You should see a new instance with a more recent launch time (scroll to the right to check)
3. Copy the newer instance's IPv4 Public IP and paste it into a new tab. Add /wp-admin after the IP address and press Enter
4. Log in with username 'group_1' and password 'alpha wario salamander' (Ignore the quotation marks but include the spaces in the password)
5. On the sidebar, Click 'Pages' → 'Sample Page' and scroll to the bottom of the sample page. Click on the linked text 'your dashboard' at the bottom, click the pencil and change '/wp-admin' to '/broken-link'. Press Enter to confirm and click 'Update' on the right panel
6. On the left sidebar, Click 'Appearance' → 'Customize' → 'Homepage Settings' then select 'a static page'. Select 'Sample Page' from the dropdown menu and click 'Publish' at the top of the screen. To confirm that the homepage has updated, navigate to the same

public IP as before (without '/wp-admin' or any other suffix). You should be able to scroll down and see the sample page

Setting a cronjob to test the status

Return to your console window

1. Enter `sudo nano /etc/crontab`
2. At the bottom of the file, below all the comments, paste the following *as a single line*:
`0,5,10,15,20,25,30,35,40,45,50,55 * * * * root .
/var/www/html/./keysAndEmail.sh >> /var/log/cron%`
3. Press Ctrl-O to save, Enter to confirm and Ctrl-X to close
4. Enter `sudo service httpd restart` to restart the web server

From now on, the keysAndEmail.sh script should run every five minutes. Everything works like this: keysAndEmail.sh runs CheckInstances.yml → CheckInstances.yml runs checkurl.sh for each EC2 in our group → checkurl.sh checks for 400 and 500 errors, and also broken links on the homepage → if an error is discovered, checkurl.sh runs email.yml → email.yml sends a notification email to the webmaster

To verify that everything is up and running, enter `sudo tail -f -n 100 /var/log/cron`

At each five-minute interval, you should see a playbook run. Because we set up a broken link on the homepage, you should receive a notification email soon after the playbook runs (and will continue to do so every five minutes unless you edit /etc/crontab, shut down the EC2 or change the broken link on the sample page back to /wp-admin).

Optional: Testing notification of a 404 error

To test for a 404 error, we need to manually pass a nonexistent page's URL to checkurl.sh. To do this, enter `bash checkurl.sh <new instance's public IP>/test.html` and check your email - including the spam folder

Note re: instance states

CheckInstances.yml only works when all the instances with the tag type:clientsites are running. If Ansible finds an EC2 in state 'stopped' or 'terminated' it will still add that EC2 to the list because it can still see that the tag is there. However, because stopped and terminated EC2s have no public IP, the playbook has nothing to pass to checkurl.sh and it won't run.

Terminated instances stay in the console until AWS clears them up automatically - [users cannot delete them after they're terminated](#). However, we can manually delete the type:clientsites tag after termination.