# Host your Application in the Azure Cloud with XAMPP and Bitnami

Vikram Vaswani

v1.0

# Table of Contents

# Introduction

If you're a PHP developer building a public-facing Web application, there are a number of good reasons why the cloud should be on your radar. It's highly scalable, allowing you to quickly scale up if you application turns out to be a hit. It's cost-efficient, because you only pay for the resources - bandwidth, CPU cycles, memory - you use. And it's secure, because cloud providers have invested a great deal of time and thought into ring-fencing applications and user data.

However, if you're new to the cloud or do most of your development locally, getting your PHP application from your local XAMPP box to the cloud can be a bit challenging. That's where this tutorial comes in. Over the next few pages, I'll walk you, step by step, through the process of deploying a PHP/MySQL application running on your local XAMPP server, to a cloud server running LAMP packaged by Bitnami. Keep reading!

# What You Will Need

Before we begin, a few quick assumptions. This tutorial assumes that you have a XAMPP installation with a working PHP/MariaDB application. It also assumes that you're familiar with the MariaDB command-line client and that you have a working knowledge of transferring files between servers using FTP.

> If you don't have a custom PHP/MySQL application at hand, use the example application included with this tutorial: it's a simple to-do list, created with Twitter Bootstrap and PHP. You can download it from here.

Now, if you're new to the cloud, you might be wondering what Azure and Bitnami are. Very briefly, Azure is Microsoft's cloud platform, which allows you to easily create Windows (and Linux) virtual servers in the cloud. Bitnami provides pre-packaged server images for these cloud servers, so that you can become productive with them the moment they come online. In short, Azure provides the cloud infrastructure, and Bitnami provides the server images and software.

For this tutorial, I'll be using LAMP packaged by Bitnami, which is Linux-based and bundles PHP, MariaDB and Apache, together with key applications and components like phpMyAdmin, SQLite, Memcache, OpenSSL, APC and cURL. LAMP packaged by Bitnami also includes a number of common PHP frameworks, including the Zend Framework, Symfony, CodeIgniter, CakePHP, Smarty and Laravel.

To deploy your application to the Azure cloud with LAMP packaged by Bitnami, here are the steps you'll follow:

- Register with Microsoft Azure
- Register with Bitnami
- Connect your Azure and Bitnami accounts
- Provision an Azure cloud server with the LAMP packaged by Bitnami
- Validate the cloud server
- Secure the cloud server

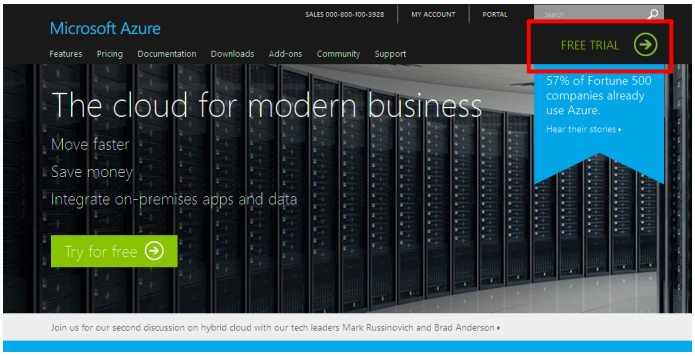
- Deploy and test your application on the cloud server

The next sections will walk you through these steps in detail.

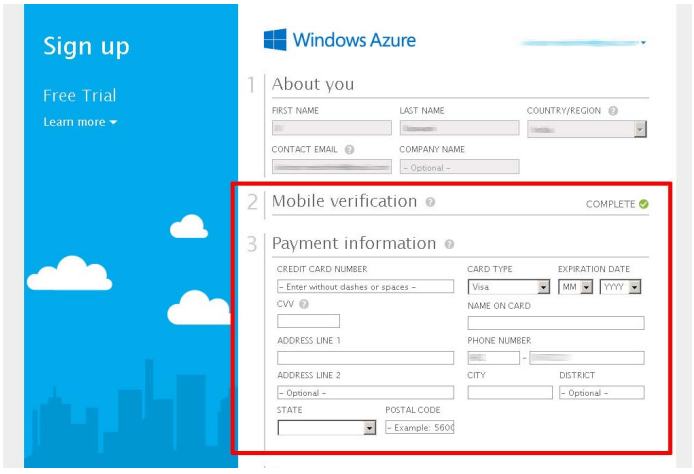
# Step 1: Register with Microsoft Azure

> At the end of this step, you will have signed up for the Microsoft Azure free trial.

Begin by creating an Azure account, by browsing to http://azure.microsoft.com and choosing the "Free Trial" option for a one month free trial. You will need an existing Microsoft account to log in and sign up for the free trial; if you don't have one, you can create one here (remember to keep track of your account username and password, because you'll need them in the next step).




Once you've signed in, sign up for the Azure free trial by providing some basic personal information and your mobile phone number. Azure will send a verification code to your mobile number, which you'll need to enter into the registration form. Once that's done, proceed to the next stage by entering your credit card information.

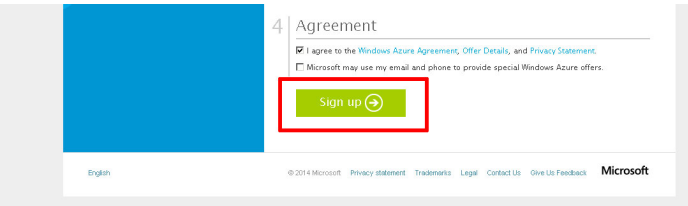



If you're wondering why you need to provide credit card information for a free trial, or if you're worried about being billed for services, relax. By default, Azure trial accounts are configured with a spending limit of $0, which means that your card will never be billed unless you remove or modify the spending limit. Azure needs your credit card information for security purposes, to avoid service misuse and to confirm your identity.

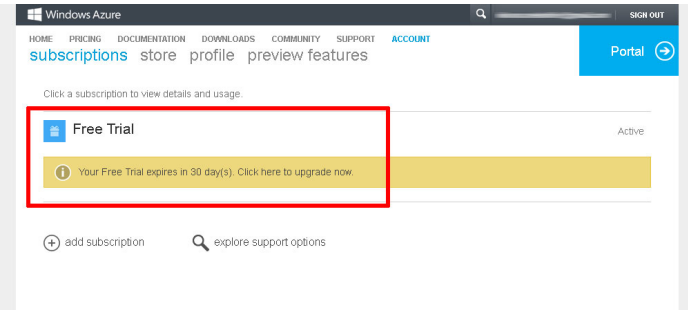
**IMPORTANT**

When your spending limit is set to $0, Azure will automatically deactivate your cloud servers so that you don't incur any charges if your usage exceeds your free quota. Read more about the Azure spending limit.

Once your payment information is entered, review Azure's terms of service, free trial details and privacy statement, and indicate your agreement by ticking the box. Then, hit the big green "Sign up" button.




The Azure account registration machine will churn away for a minute or so, and you will then be redirected to your Azure account management page, which allows you to manage your subscriptions, edit your profile and get support. You should see that your free trial is now active in the subscription list.




# Step 2: Register with Bitnami

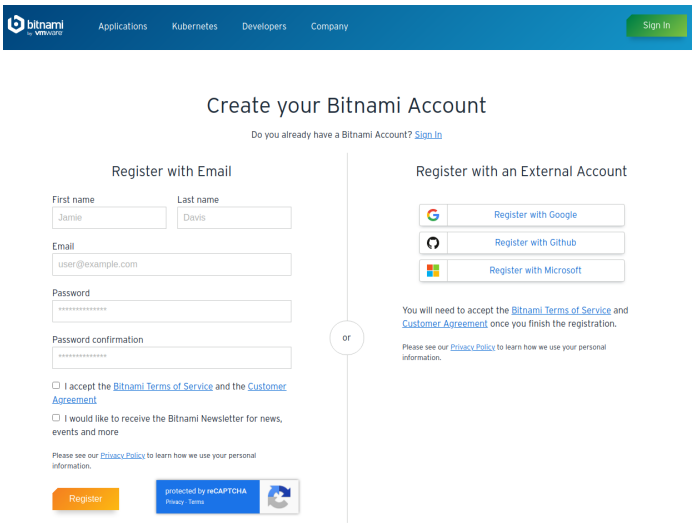
At the end of this step, you will have created a Bitnami account.

The next step is to create a Bitnami account, so that you can launch a cloud server with LAMP packaged by Bitnami image. If you have a Google, Microsoft or Github account, you can use your credentials from those services with Oauth to create your Bitnami account.

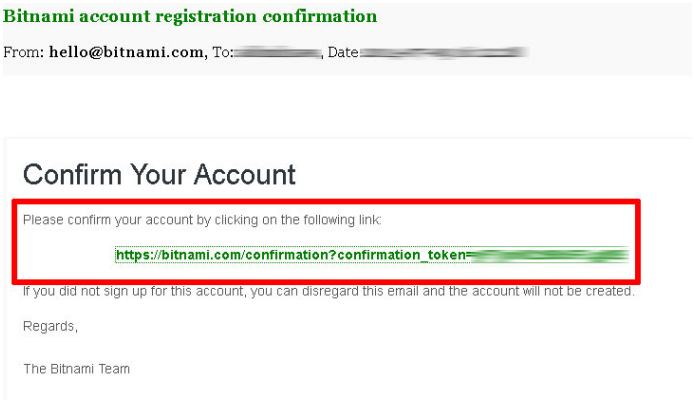
If you don't have accounts with those services (or you don't want to use them), you can use your email address and password to create a Bitnami account, as described below:

- Head to the Bitnami sign-up page.
- Enter your name and email address.
- Choose a password.
- Review and agree to the Bitnami terms of service.

Then, use the "Sign up" button to create your account.




Bitnami will send you an email with a verification link which you'll need to click or browse to, to activate your account. This will also sign you in to your Bitnami account.



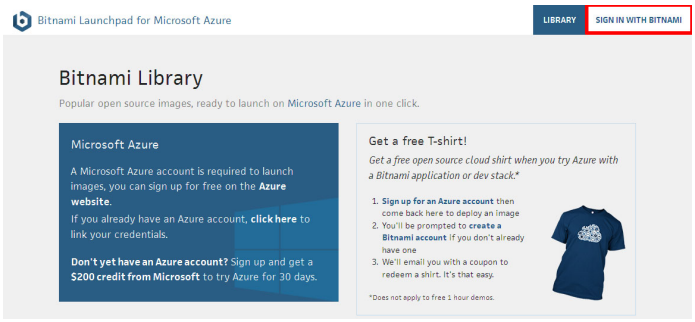

# Step 3: Connect your Azure and Bitnami Accounts

> At the end of this step, your Bitnami Launchpad for Azure will be configured and you will be ready to provision a cloud server.

The easiest way to set up your Azure cloud server with LAMP packaged by Bitnami is via the Bitnami Launchpad for Azure, which gives you a simple control panel to provision, start, stop and check status of your Azure cloud servers. However, to use it, you must first connect your Azure and Bitnami accounts.

To do this:

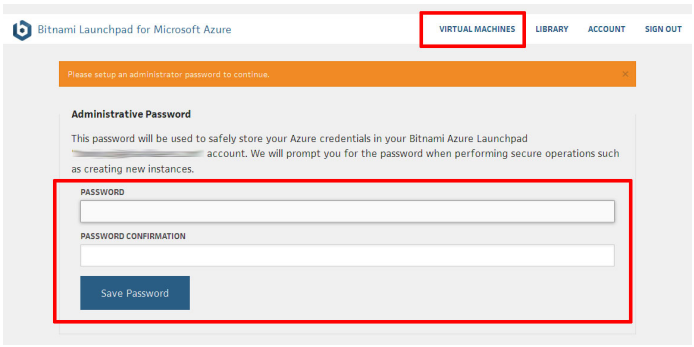
- Log in to your Bitnami account if you're not already logged in
- Browse to https://azure.bitnami.com/
- Select the "Sign in with Bitnami" link in the top right corner.




The Launchpad will recognize your Bitnami credentials and automatically sign you in.

The next step is to set up an administrative password for the Bitnami Launchpad and connect your Azure account with your Bitnami account.

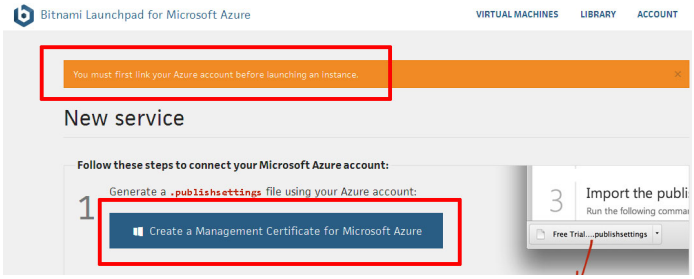
Select "Virtual Machines" in the Launchpad menu and - since this is your first time - you'll be prompted to enter an administrative password.

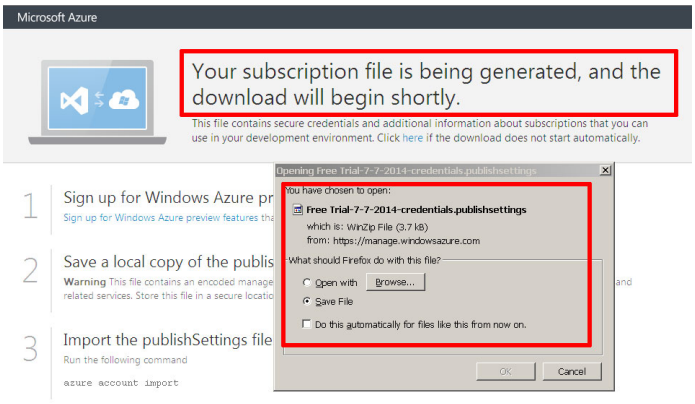



The administrative password offers an additional level of protection against misuse: you'll need to enter it when performing certain operations, such as creating new Azure server instances. Again, make sure you note it down for future reference.

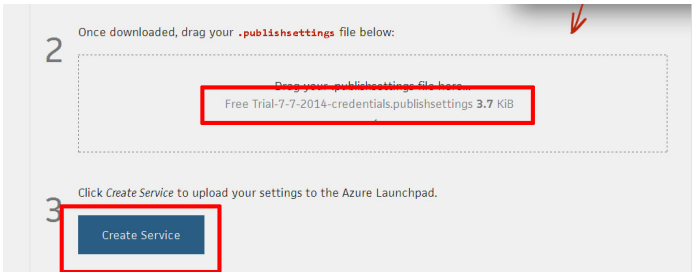
Next, you'll be asked to connect your Azure account to your Bitnami account, by transferring a management certificate generated by Azure into your Launchpad account. Although this sounds complex, the Launchpad makes this a simple two-step process.

- Select the "Create a Management Certificate for Microsoft Azure" button. So long as you're still logged in to Azure, this will generate a .publishsettings file which you'll be prompted to download through your browser.






- Once the file is downloaded, drag it from your desktop to the Bitnami Launchpad page and then select the "Create Service" option to upload the management certificate to the Launchpad.



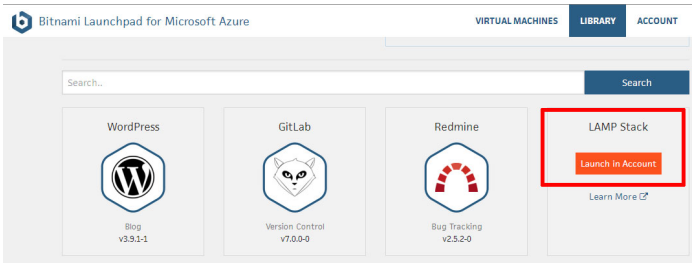

Your Azure and Bitnami accounts will now be connected.

# Step 4: Provision an Azure Cloud Server

> At the end of this step, your Azure cloud server will be running and you will be able to access it through your Web browser.

To provision your Azure cloud server:

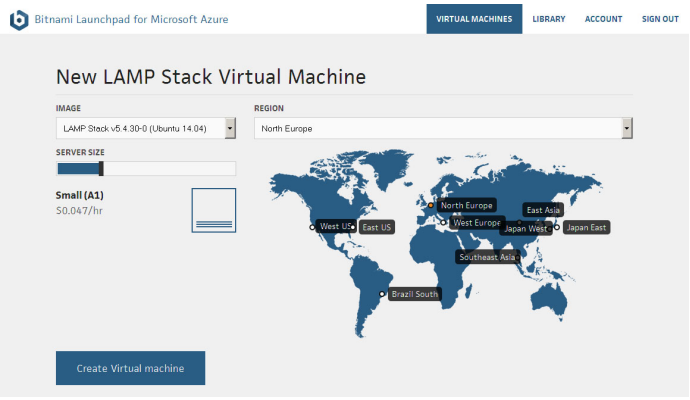
- Select "Library" in the Launchpad menu
- Look through the list of applications and images available in Bitnami until you find LAMP Stack
- Select it and click "Launch in account".




- Define the region and size for your Azure server. Choose from an "Extra Small" server, which uses shared virtual cores to an "Extra Large" server, which has 8 dedicated virtual cores, depending on the needs of

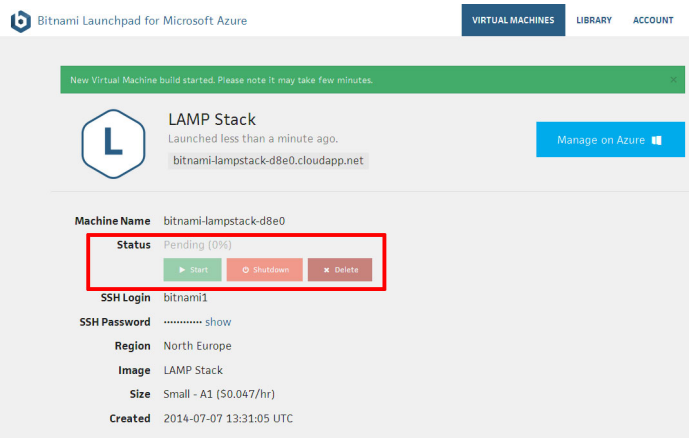
your application. For more information, refer to Azure's virtual machines and pricing sheet.

**TIP** "Extra Small" servers work just fine for most PHP application development tasks.

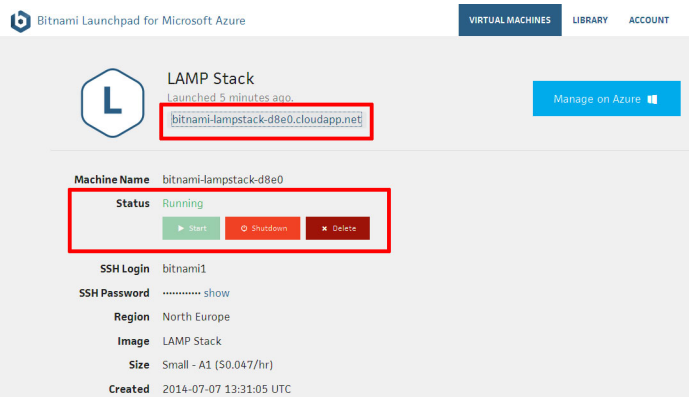



- Confirm your selection by hitting the "Create Virtual machine" button at the end of the page.

The Bitnami Launchpad will now assign the new server a host name and begin spinning it up. The process usually takes a few minutes: a status indicator on the page provides a progress update.

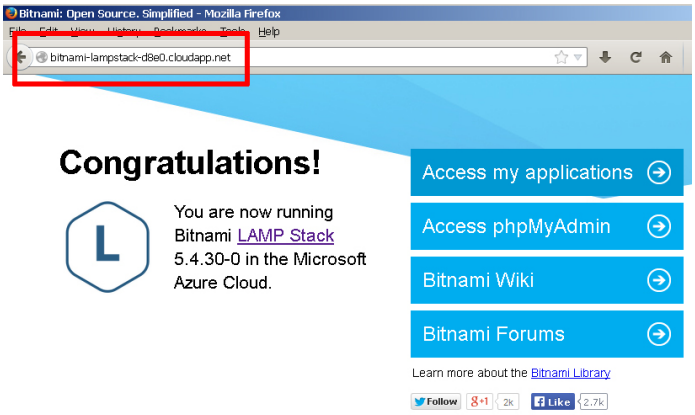



Once the cloud server has been provisioned, the status indicator will show that it's "running", and the host name specified in the Bitnami Launchpad will become an active link.




At this point, you should be able to browse to the cloud server, either by clicking the link in the Bitnami Launchpad or entering the cloud server host name directly into your browser's address bar. You should see a

welcome page like the one below (just so you know, it's served up by Apache, which is part of LAMP packaged by Bitnami).

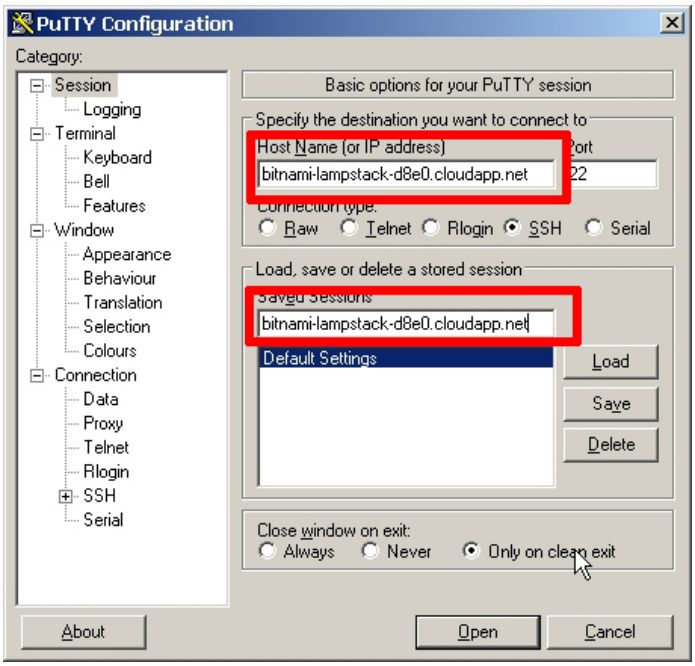



# Step 5: Test PHP and MariaDB

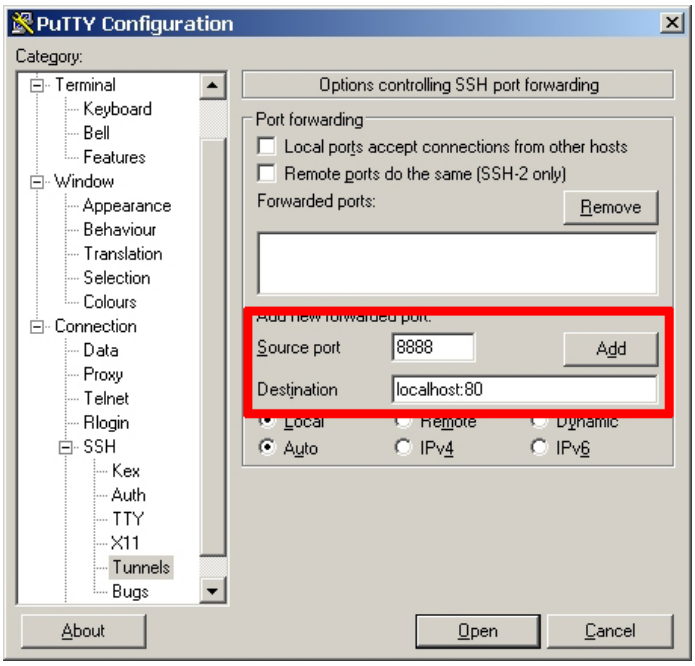
> At the end of this step, you will have logged in to your cloud server and verified that PHP, MariaDB and phpMyAdmin are working correctly.

You can now connect to the cloud server and test PHP to make sure it's working correctly and has all the extensions you need. The easiest way to do this is with PuTTY, a free SSH client for Windows and UNIX platforms.

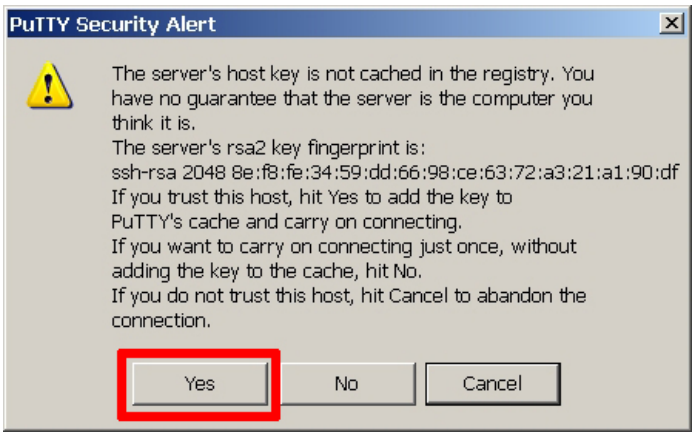
- Download the PuTTY ZIP archive from its website.
- Extract the contents to a folder on your desktop.
- Double-click the *putty.exe* file to bring up the PuTTY configuration window.
- Enter the host name of your cloud server into the "Host Name (or IP address)" field, as well as into the "Saved Sessions" field.
- Click "Save" to save the new session so you can reuse it later.



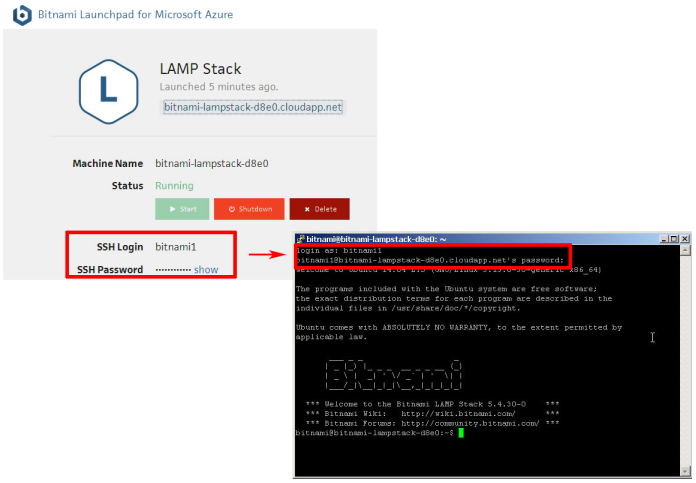

- In the "Connection _ SSH _ Tunnels" section, create a secure tunnel for the phpMyAdmin application by forwarding source port "8888" to destination port "localhost:80".
- Click the "Add" button to add the secure tunnel configuration to the session.




- Go back to the "Session" section and save your changes by clicking the "Save" button.
- Click the "Open" button to open an SSH session to the server.
- PuTTY will first ask you to confirm the server's host key and add it to the cache. Go ahead and click "Yes" to this request.



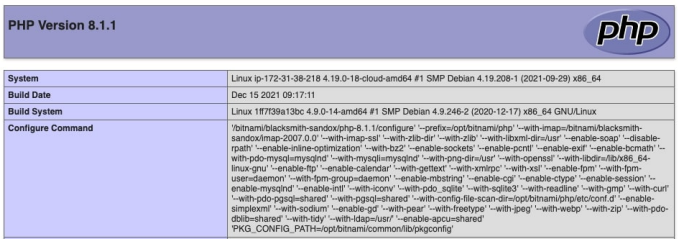

- Enter your SSH username and password when prompted. This information is available on the Bitnami detail page for your cloud server. You should then be logged in to your cloud server.




By default, LAMP packaged by Bitnami includes running Apache and MariaDB servers, and all the packages that come with the stack are located in the */opt/bitnami* directory. Most importantly, there's also a *phpinfo.php* file in the */opt/bitnami/docs* directory, which you can copy to the Apache web server root to verify PHP's capabilities.

```
shell> cd /opt/bitnami
shell> cp docs/phpinfo.php apache2/htdocs/
```

Once the file has been copied, browse to *http:///[your-cloud-server-hostname]/phpinfo.php* and you should see the output of the *phpinfo()* command.



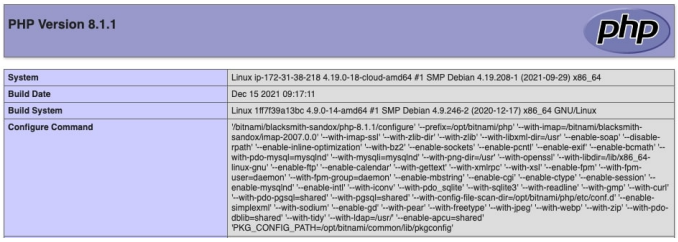

With this, you know that your PHP installation is configured and working correctly.

You can also check that MariaDB is working: simply launch the MariaDB command-line client by typing 'mysql' at the shell prompt.

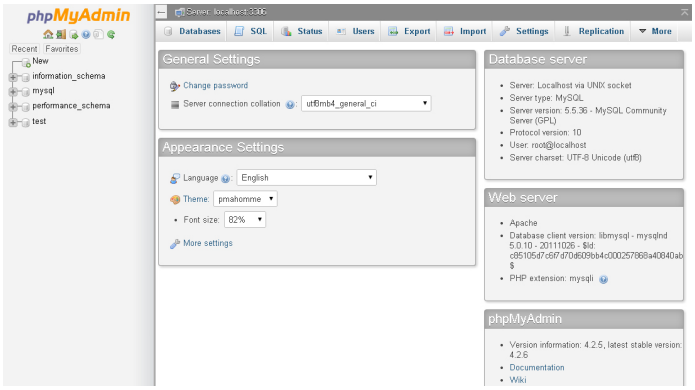
```
shell> mysql
```

The client should start up and connect to the local MariaDB server, displaying a welcome message as shown below.

You should also be able to access phpMyAdmin through the secure SSH tunnel you created, by browsing to http://127.0.0.1:8888/phpmyadmin.




To log in, use username 'root' and default password 'bitnami'.




In case you'd like to troubleshoot errors or modify the configuration for Apache, PHP or MariaDB - for example, adjusting the maximum upload file size in PHP or [changing the path to the MariaDB data directory] - here are the locations for key configuration and log files in LAMP packaged by Bitnami:

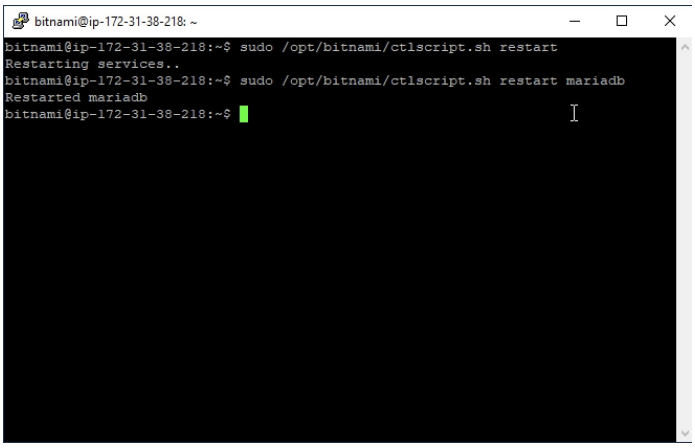
| | Configuration file(s) | Log file(s) |
|---|---|---|
| Apache | */opt/bitnami/apache2/conf/httpd.conf* | */opt/bitnami/apache2/logs/error_log* |
| PHP | */opt/bitnami/php/etc/php.ini* | - |
| MariaDB | */opt/bitnami/mariadb/conf/my.cnf* | */opt/bitnami/mariadb/logs/mysqld.log* |

Usually, you'll need to restart your server(s) for your changes to take effect. LAMP packaged by Bitnami includes a control script that lets you easily stop, start and restart Apache, MariaDB and PHP. The script is located at */opt/bitnami/ctlscript.sh*. Call it without any arguments to restart all services:

```
shell> sudo /opt/bitnami/ctlscript.sh restart
```

Or use it to restart a specific service only by passing the service name as argument - for example 'mysql':

```
shell> sudo /opt/bitnami/ctlscript.sh restart mariadb
```




# Step 6: Configure Server Security

At the end of this step, you will have updated the default account passwords and configured key-based SSH authentication to make your cloud server more secure.

Although this is an optional step, it's best practice to secure your server by changing default passwords and configuring SSH key-based access. For example, the default MySQL installation uses 'bitnami' as the root password, so this should be changed right away, by using the command shown below (replace 'guessme' with your new password):

```
shell> mysqladmin -u root password guessme
```

You can also change the password of the default 'bitnami1' account, as shown below:

```
shell> passwd
```

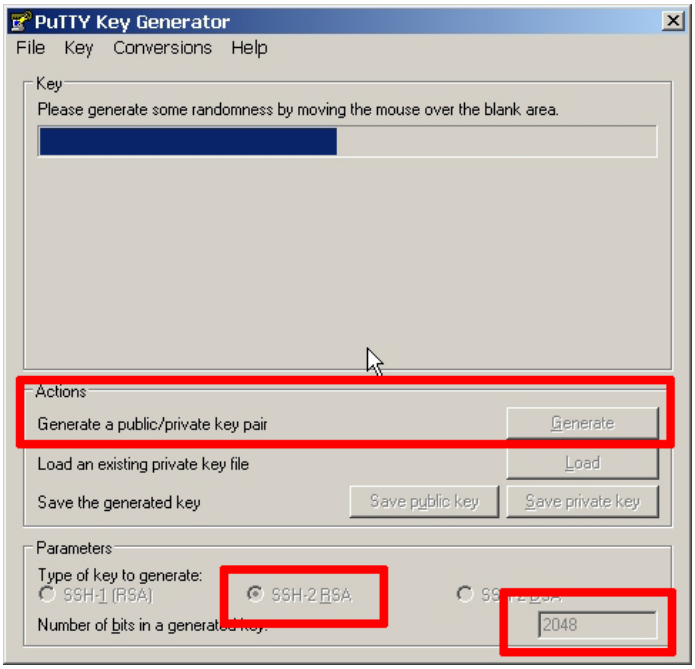
You will be prompted to enter the current password, then a new password.

To further secure the server, it's a good idea to replace SSH's password-based authentication with key-based authentication. This ensures that only users with access to the correct key can log in to the server.

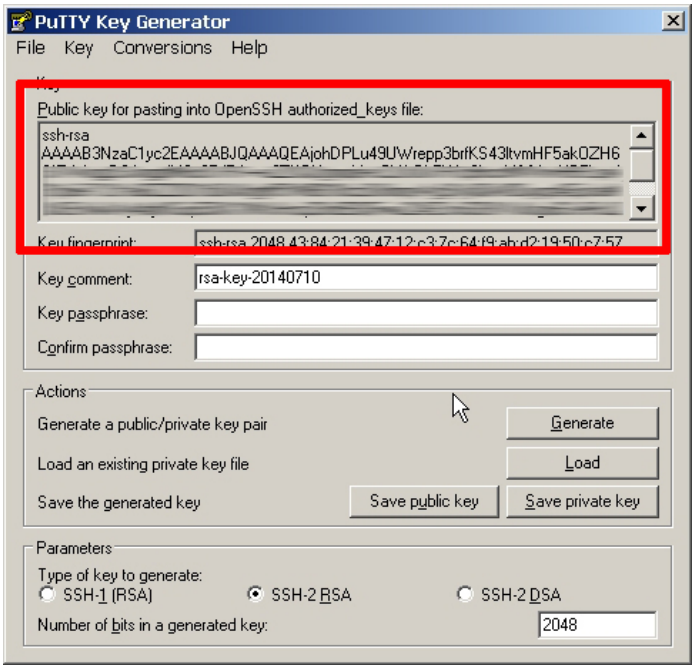
To do this:

- Launch PuTTY's key generation tool *puttygen.exe*.

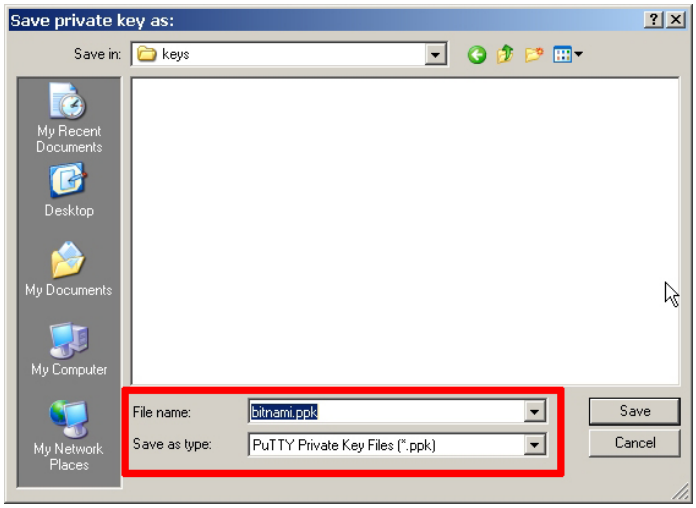
- Select "SSH-2 RSA" as the key type.
- Set the number of bits to 2048.
- Click the "Generate" button to start the process of generating a public/private key pair.
- Move your mouse around to generate some randomness for the key.




PuTTY will generate the keys and display the public key, as in the screenshot below.



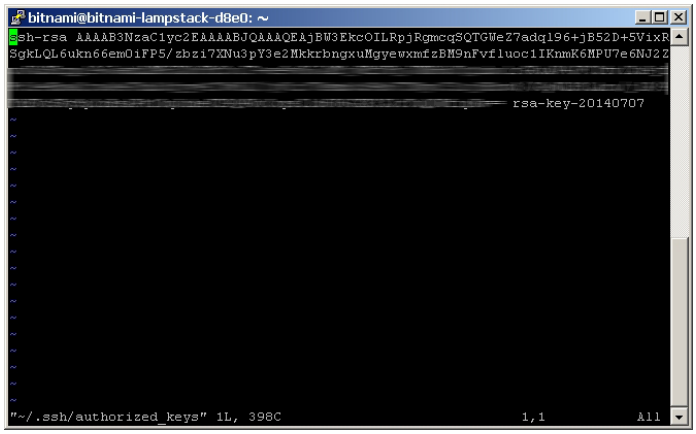

- Select "Save private key" and select a file name and location for the private key file - in this example, *bitnami.ppk*.
- For added security, enter a passphrase. Leave it blank for faster, passphrase-less authentication.



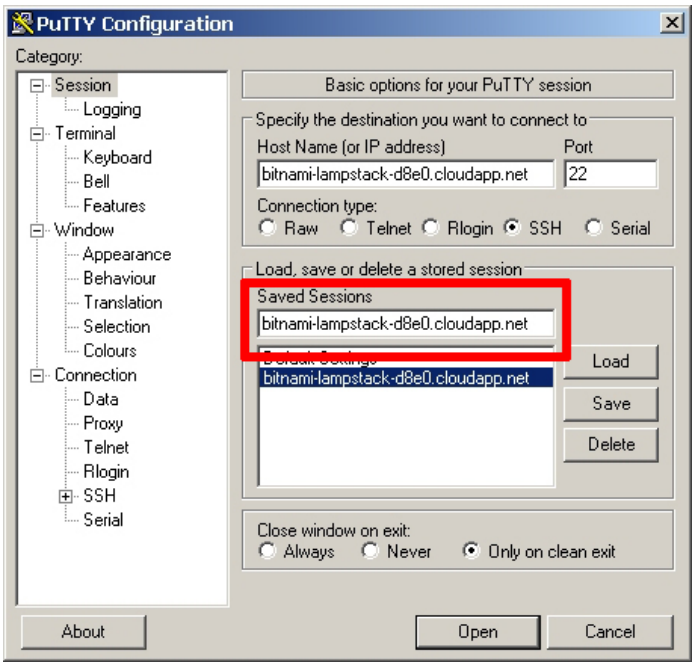

- On the cloud server, edit the *~/.ssh/authorized_keys* file and copy the public key from the PuTTY key generator into it. Ensure that the public key is pasted on a single line.

```
shell> cd ~
shell> vi ~/.ssh/authorized_keys
```

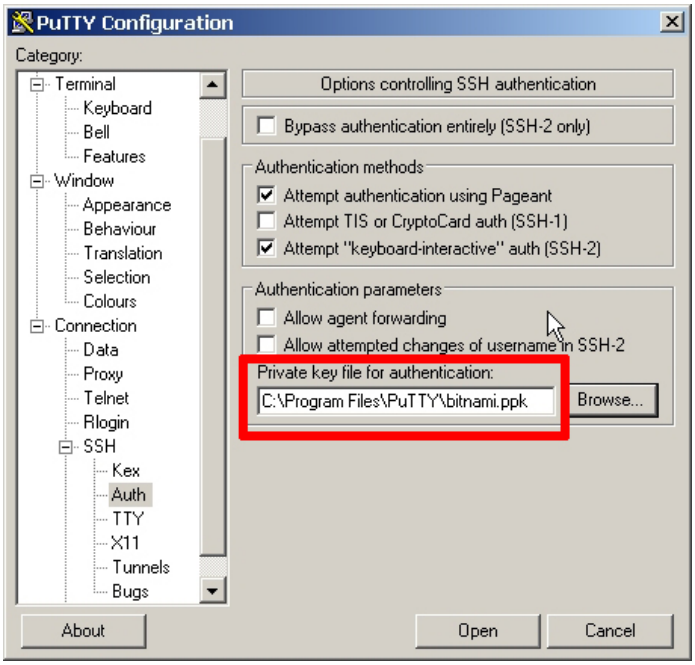
- Save the *~/.ssh/authorized_keys* file. Here's what the file should look like:



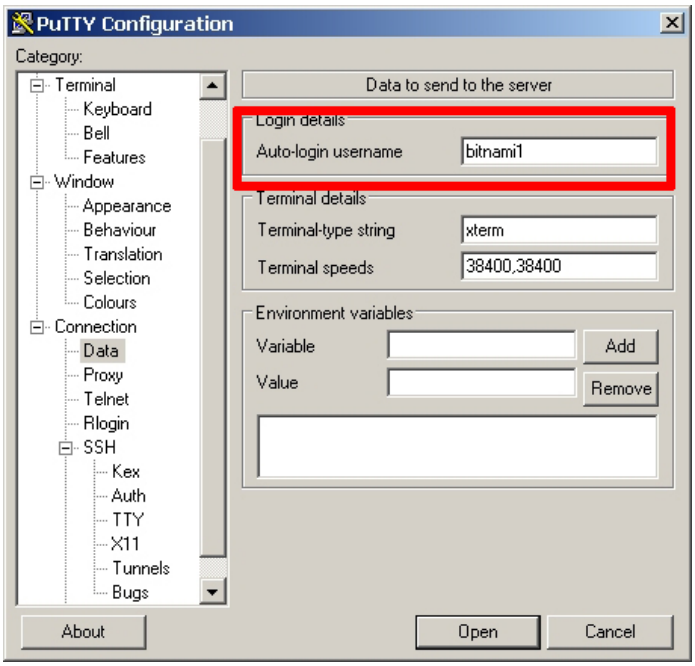

- Log out of the cloud server.
- On your desktop, start PuTTY with the *putty.exe* file.
- In the PuTTY configuration window, select the session from the "Saved Sessions" list.
- Click "Load" to load the session.




- In the "Connection _ SSH _ Auth" section, select the private key file you saved a few steps ago.

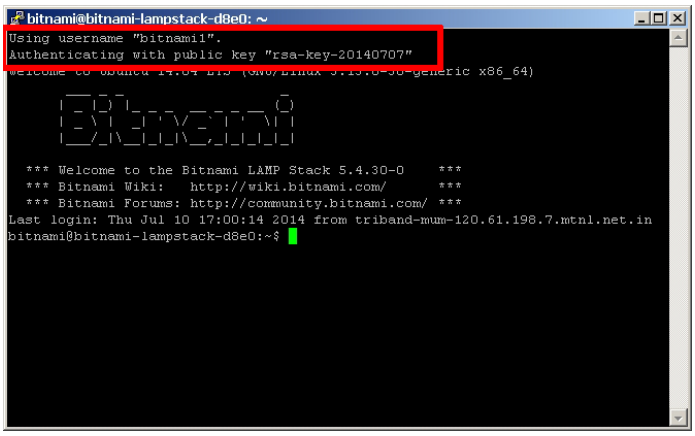



- In the "Connection _ Data" section, enter the username into the "Auto-login username" field.




- Go back to the "Session" section and save your changes by clicking the "Save" button.

Now, open a new connection to the cloud server by clicking "Open". This time, PuTTY should use the private key to automatically log you in to the cloud server, without requiring you to enter a password.

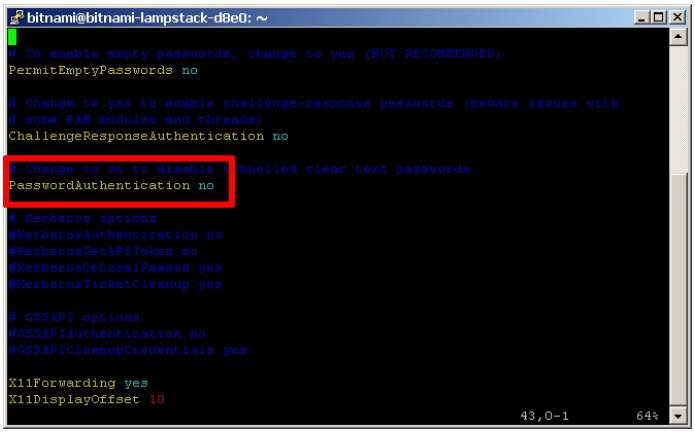



Assuming it all worked, the final step is to turn off password-based authentication.

- Edit the */etc/ssh/sshd_config* file on the cloud server.

```
shell> vi /etc/ssh/sshd_config
```

- Find the line containing 'PasswordAuthentication yes' and change it to 'PasswordAuthentication no'.




- Restart the SSH daemon

```
shell> sudo service ssh restart
```

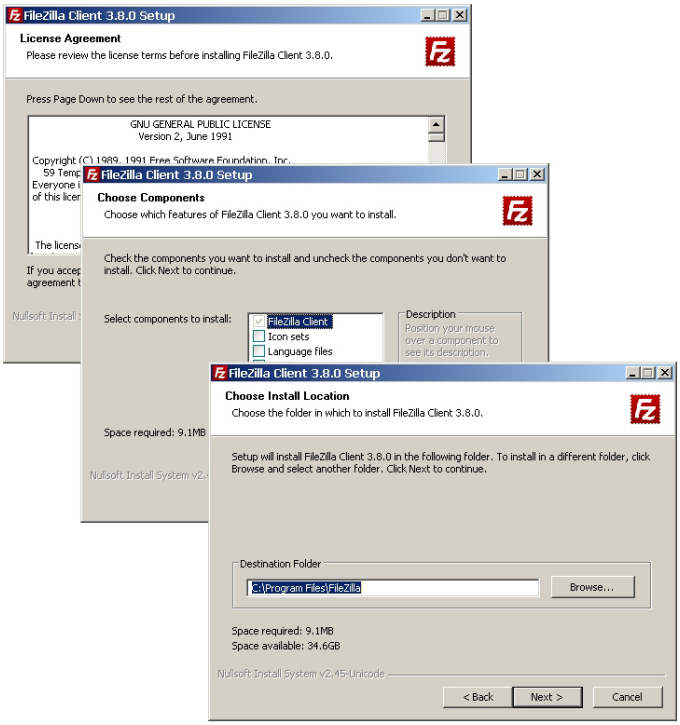
At this point, your private SSH key is the only way of gaining access to your cloud server - so keep it safe!

# Step 7: Deploy the XAMPP Application to the Cloud Server

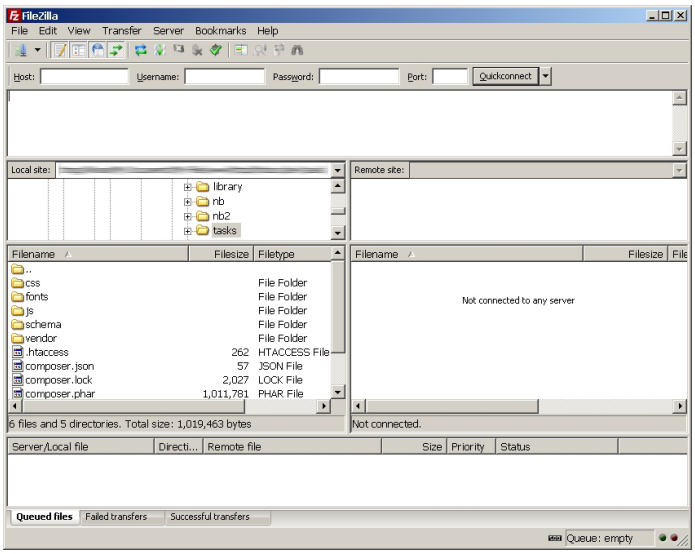
> At the end of this step, your PHP/MariaDB application will be running in the cloud.

Your cloud server is now provisioned, secured and has a functional PHP/MariaDB environment. All that's left is for you to transfer your application code from your local XAMPP environment to your cloud server and set up the database.

The easiest way to transfer files to the server is with FTP or SFTP. Although you can use any FTP/SFTP client, I like FileZilla, a cross-platform, open source and feature-rich client. Download it from the FileZilla website and install it using the automated installer - it's a quick process, only requiring you to agree to the license, choose the components (the default selection is usually fine) and specify the installation directory.

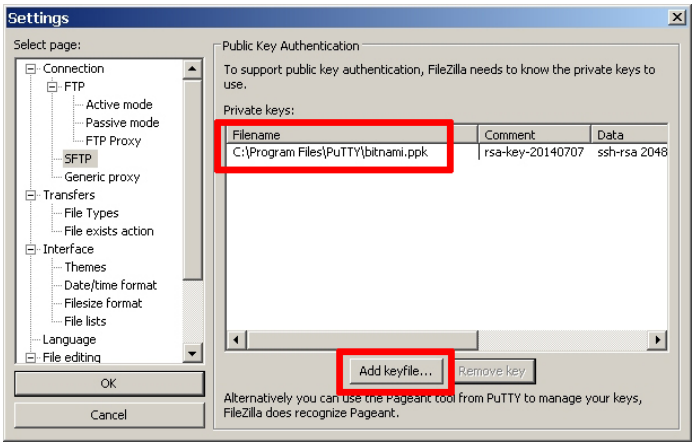



Once FileZilla is installed, launch it and you'll arrive at the main split-screen interface, one side for your local directories and the other for remote directories.

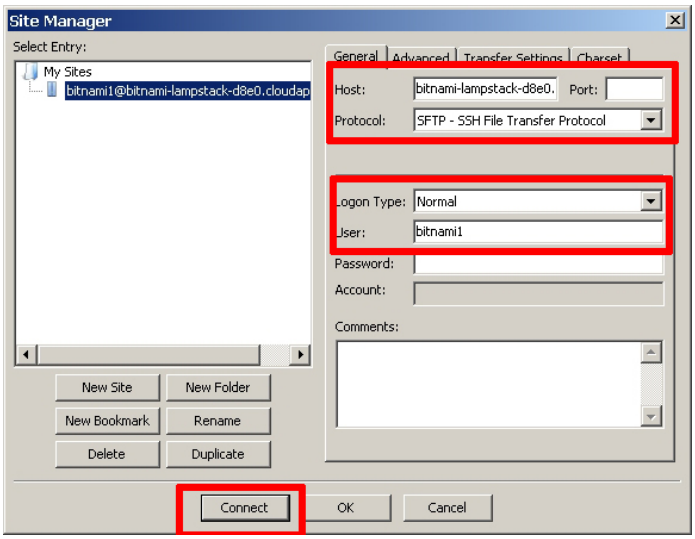



To connect to the cloud server and deploy your application, follow these steps:

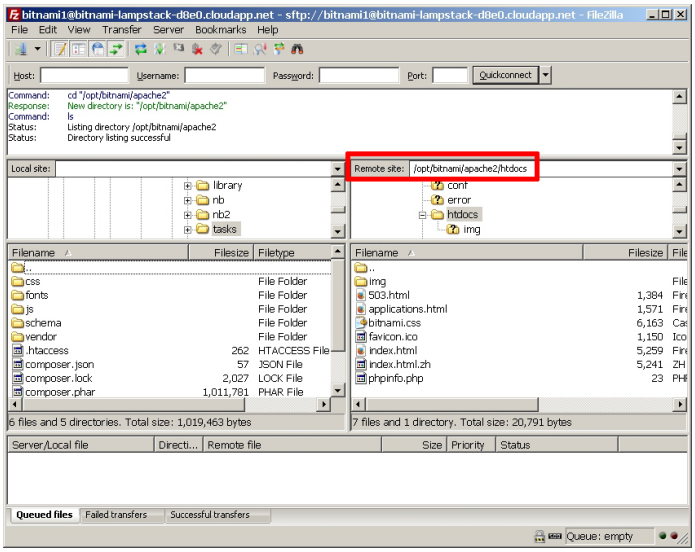
- Use the "Edit _ Settings" command to bring up FileZilla's configuration settings.
- Within the "Connection _ SFTP" section, use the "Add keyfile" command to select the private key file for your server. FileZilla will use this private key to log in to the cloud server.




- Use the "File _ Site Manager _ New Site" command to bring up the FileZilla Site Manager, where you can set up a connection to your cloud server.
- Enter your server host name or IP address and user name.
- Select "SFTP" as the protocol and "Normal" as the logon type.




- Use the "Connect" button to connect to the cloud server and begin an SFTP session.
- On the remote server side of the window, change to the */opt/bitnami/apache2/htdocs* directory
- On the local server side of the window, change to the directory containing your application code.
- Upload your XAMPP application code to the remote directory by dragging and dropping the files from the local server to the cloud server (you can back up the original contents of the directory if you wish, by downloading them first).

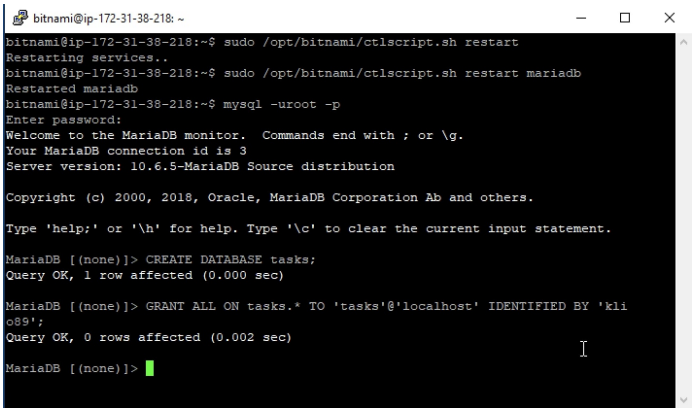



- Once the files are transferred, log in to the server console using PuTTY.
- Create a database for the application using the MariaDB command-line client (you can use phpMyAdmin if you prefer a graphical interface). For example, since the application is a to-do list, let's call the database 'tasks'.

```
mysql> CREATE DATABASE tasks;
```

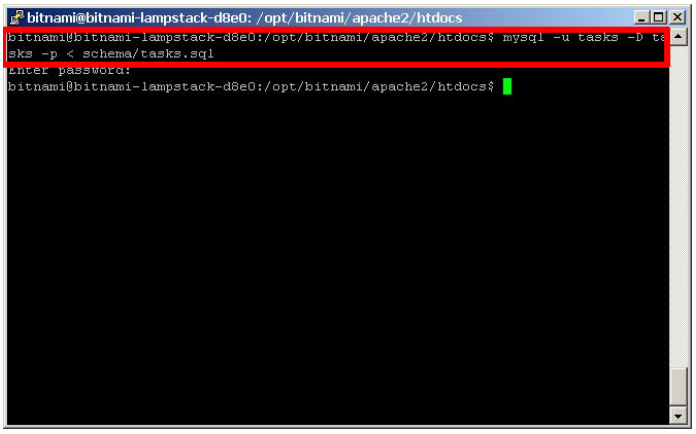
- Follow best practices and create a separate MariaDB user with privileges to access only this database.

```
mysql> GRANT ALL ON tasks.* TO 'tasks'@'localhost' IDENTIFIED BY 'klio89';
```

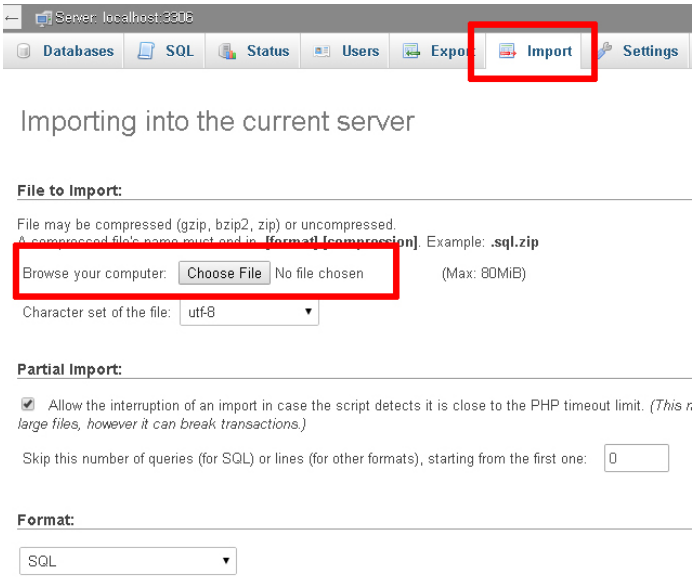



- If required, update database credentials in your application. Then, install the application schema in the new database (assuming you already uploaded it with the application code). For example, you can use the following command with the MariaDB command-line client:

```
shell> mysql -u tasks -D tasks -p < schema/tasks.sql
```

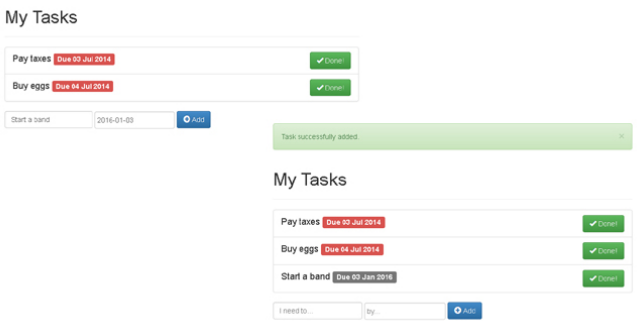



If you're logged in to phpMyAdmin, you can also import the database schema from your local XAMPP system. To do this, select the "Import" tab of the phpMyAdmin dashboard, select the file containing the schema, and click "Go" to have the tables created in your selected database.




You can also learn more about using phpMyAdmin to back up and restore databases.

Browse to your cloud server's host name and your application should be active. Here are a few screenshots of the example to-do list application running on the cloud server.



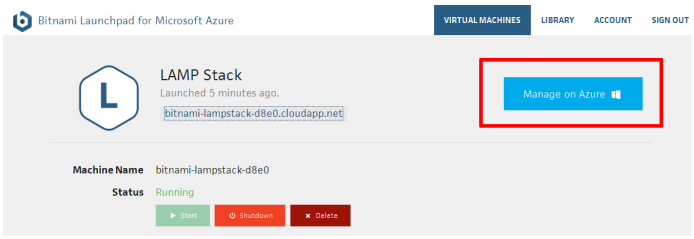

Congratulations! You've successfully deployed your XAMPP application in the cloud.

# Understand Azure's Dashboard and Monitoring Tools

To help you get the most out of your cloud server, Azure makes a number of administrative and monitoring tools available. These help you keep track of your application's performance in the cloud and optimize your server and software configuration as needed.

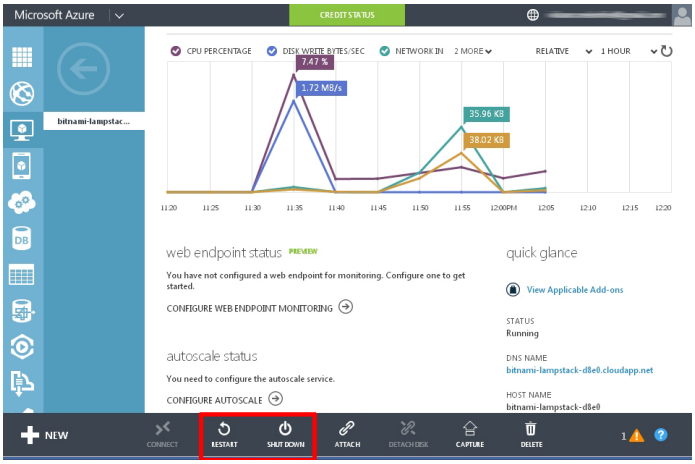
To see these tools in action:

- Browse to the Bitnami Launchpad for Azure and sign in if required using your Bitnami account.
- Select the "Virtual Machines" menu item.
- Select your cloud server from the resulting list.
- On the server detail page, click the "Manage on Azure" button. You will be redirected to the Azure management console.




- Sign in to Azure if required using your existing Microsoft account.

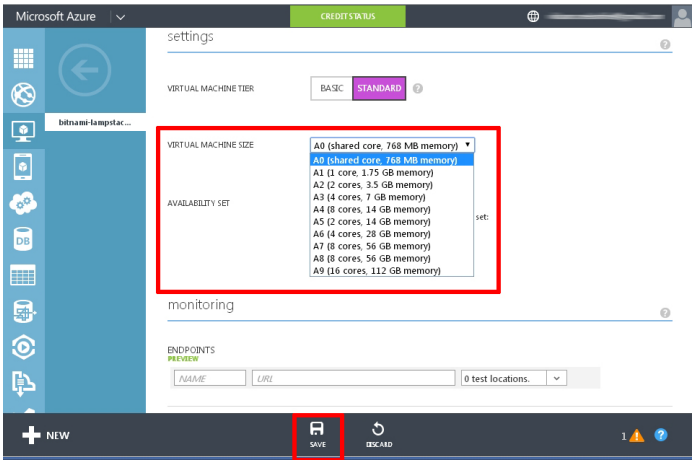
Once you do this, you'll arrive at the Azure dashboard, which lists all your virtual machines and cloud services. Select the cloud server you created with the Bitnami Launchpad to arrive at the dashboard, which provides an overview of CPU usage, disk activity and network traffic over the last hour, 24 hours or 7 days. You'll notice that the toolbar below the activity monitor provides controls to restart or shut down the cloud server.




You can also use the Azure dashboard to resize your virtual server. To do this:

- Select the "Configure" tab of the dashboard.

- Select a new "Virtual Machine Size" on the resulting page.

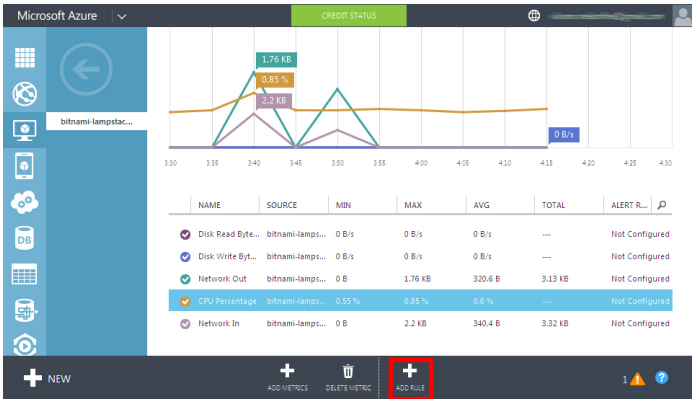



- Click the "Save" command in the toolbar at the bottom of the page.

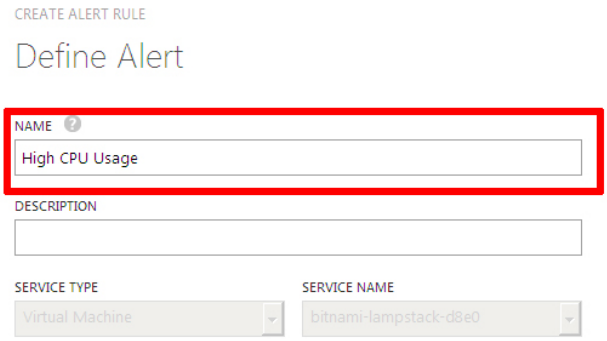
Azure will resize and restart your cloud server with the new configuration.

You can also define alerts, so that you're automatically informed if any of the above metrics crosses a particular threshold. To do this:

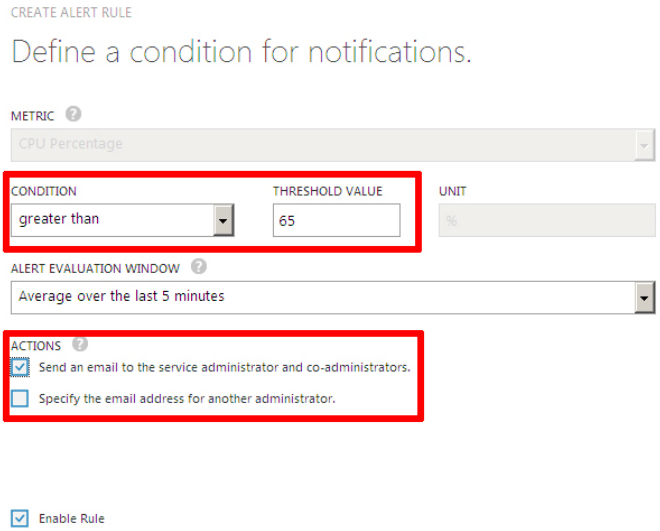
- Select the "Monitor" tab of the dashboard.
- Select a metric on the resulting page.
- Click the "Add Rule" command in the toolbar at the bottom of the screen.




- Specify a name for the new alert.




- Define the alert threshold for the selected metric, the evaluation window and the email address to be notified when the alert is triggered.



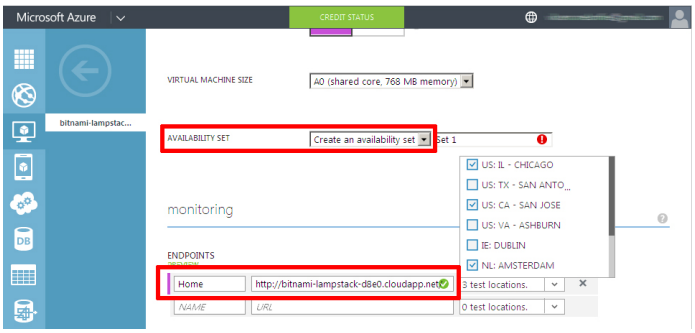

When the above rule is submitted, Azure will automatically send you an email when your cloud server's CPU usage exceeds 65%.

You can also monitor the uptime of your XAMPP application, by configuring an HTTP endpoint that is automatically checked from up to three different geographical locations. To do this:

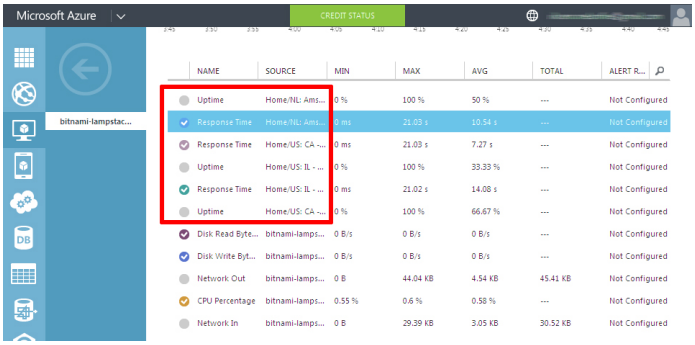
- Select the "Configure" tab of your virtual machine dashboard.
- Create a new availability set and give it a name.

| | |
|---|---|
| **NOTE** | An availability set is a group of Azure cloud servers, designed to provide redundancy if one goes down. In the event of a server fault or maintenance event, Azure will use the servers in the availability set to ensure continuous service. Read more about availability sets. |

- Set the application's index page URL (or any other URL you wish) as a monitoring. endpoint.
- Save your changes.




Azure will restart your cloud server and configure the URL you specified as a monitoring endpoint. This will also add new metrics for response time and uptime to your list of available metrics, which you can plot in your dashboard or configure alerts against.




# Improve Application Performance

Web application performance problems are hard to debug at the best of times, and more so when your server is in the cloud and running a pre-packaged stack. The responsiveness of your application at any given moment depends on numerous factors: server type, network bandwidth, cloud provider load, database load, caching system in use, application code structure, query structure and various other variables.

**IMPORTANT**

LAMP packaged by Bitnami already uses the Apache Event MPM and PHP-FPM for reduced memory usage and an increase in the number of simultaneous requests that the server can handle (more information). It also comes with the mod_pagespeed Apache module activated to rewrite pages on the fly and improve latency.

If you're finding that your PHP/MariaDB application's performance is not up to scratch, here are a few general tips you can consider:

- LAMP packaged by Bitnami includes APCu, a popular PHP bytecode cache. Usually, when a PHP script is executed, the PHP compiler converts the script to opcodes and then executes the opcodes. APC provides a framework for opcode caching, thereby speeding up PHP applications without needing any code changes. Make sure your APC cache has enough memory and a long TTL. Read more about APCu and how to use APC with PHP and Bitnami.

- LAMP packaged by Bitnami also includes the PHP memcache extension. Memcache is a high-performance, distributed memory object caching system. Consider using memcache to store frequently-

accessed fragments of data in memory as arrays, thereby reducing the load on your MariaDB database server. Read more about memcache in PHP.

- Turn on MariaDB's slow query log and set MariaDB's 'long_query_time' variable to a low number. This lets you track which of your queries are performing inefficiently and adjust them, either structurally or by applying table indexes as needed, to improve performance. You can use tools like mysqldumpslow or mysql-slow-query-log-visualizer to parse and analyze the slow query logs generated.
- If your application is database-heavy, you'll gain performance by giving the MariaDB server more memory. You may use the MariaDB Optimization and Tuning guides, to identify which server parameters need tuning, and incrementally make changes to your server's cache and buffers to improve performance. For example, if your tables are all MyISAM, disable InnoDB in your *my.cnf* file to save further memory.
- Unload Apache modules which you don't need to save memory, and adjust the log level to errors only.
- Minify your JavaScript code, and consider using a CDN for static content like images.

Good luck, and happy coding!

# Useful Links

- Microsoft Azure
- Bitnami Launchpad for Azure
- LAMP packaged by Bitnami
- LAMP packaged by Bitnami Documentation
- PuTTY
- FileZilla
- Example Project (.zip)

# About the author

Vikram Vaswani is the founder of Melonfire, an open source software consultancy firm, and the author of seven books on PHP, MySQL and XML development. Read more about him at http://vikram-vaswani.in/.