

Host your Application in the Amazon Cloud with XAMPP and Bitnami

Vikram Vaswani

v1.0

Table of Contents

Introduction	1
What You Will Need	1
Step 1: Register with Amazon Web Services.....	2
Step 2: Register with Bitnami	4
Step 3: Connect your AWS and Bitnami Accounts	5
Step 4: Provision an AWS Cloud Server	10
Step 5: Test PHP and MariaDB	14
Step 6: Deploy the XAMPP Application to the Cloud Server	19
Improve Application Performance	24
Useful Links.....	24
About the author.....	25

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 your 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/MySQL 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/MariaDB 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 Amazon Web Services and Bitnami are. Very briefly, [Amazon Web Services](#) is a cloud platform, which allows you to easily create Windows and Linux virtual servers online. 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, Amazon provides the cloud infrastructure, and Bitnami provides the server images and software. And since both Amazon and Bitnami have a free tier, you can run and manage a full-featured PHP server for free for 1 year.

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 Amazon cloud with LAMP packaged by Bitnami, here are the steps you'll follow:

- Register with Amazon Web Services (AWS)
- Register with Bitnami
- Connect your AWS and Bitnami accounts
- Provision an AWS cloud server with LAMP packaged by Bitnami

- Validate 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 Amazon Web Services

At the end of this step, you will have signed up for the Amazon Web Services free tier. If you already have an Amazon Web Services account, you may skip this step.

You will need an existing Amazon account to log in and sign up. To create it, follow these steps:

- Browse to <https://aws.amazon.com> and click the "Create an AWS account" button at the top of the page.
- In the resulting page, enter an email address, a password, and an AWS account name. Then, click "Continue" to start the registration process.

The screenshot shows the 'Create an AWS account' form. It has four input fields: 'Email address', 'Password', 'Confirm password', and 'AWS account name'. Below these is a large yellow 'Continue' button. At the bottom left is a link to 'Sign in to an existing AWS account'. The footer contains copyright information and links to 'Privacy Policy' and 'Terms of Use'.

Create an AWS account

Email address

Password

Confirm password

AWS account name ⓘ

Continue

Sign in to an existing AWS account

© 2018 Amazon Web Services, Inc. or its affiliates.
All rights reserved.
[Privacy Policy](#) | [Terms of Use](#)

- Once you've signed in to Amazon, sign up for AWS by selecting the account type and providing some basic contact information and your mobile phone number.

Contact Information

All fields are required.

Please select the account type and complete the fields below with your contact details.

Account type Professional Personal

Full name
[Text input field]

Company name
[Text input field]

Phone number
[Text input field]

Country/Region
United States

Address
Street, P.O. Box, Company Name, etc.
Apartment, suite, unit, building, floor, etc.

- Once that's done, proceed to the next stage by entering your credit card information. Click the "Secure Submit" button to continue with the account creation.

Payment Information

Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the AWS Free Tier Limits. Review frequently asked questions for more information.

Credit/Debit card number
[Text input field]

Expiration date
01 2018

Cardholder's name
[Text input field]

Billing address
 Use my contact address
 Use a new address

Secure Submit

If you're worried about how much you'll be billed for services, relax. When you first sign up for AWS, you get automatic access to the [AWS Free Tier](#), which entitles you to 12 months of free usage up to certain limits. This includes 750 hours per month of free usage of [Amazon EC2](#) micro servers, which are just right for development or low-traffic website hosting. So long as your usage falls within the limits of the free tier, your credit card will never be billed. However, Amazon still needs your credit card information for security purposes, to avoid service misuse and to confirm your identity.

IMPORTANT

You should fully understand [the limits of the AWS free tier](#) to avoid being unduly charged for service usage.

- Amazon will now verify your identity, by making an automated call to your mobile phone number and prompting you to enter the PIN number displayed on the screen.
- Once your identity is verified, choose the "Basic" support plan (also free) and confirm your account.

NOTE

At this point, make sure that you have subscribed a plan, even if you decide to register for the free tier or "Basic" support plan.

The AWS account registration machine will churn away for a minute or so, and you will then be redirected to a welcome page, which includes a link to the AWS management console. You should also receive an account confirmation email, which tells you that your account is good to go.

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.

The screenshot shows the Bitnami registration interface. At the top, there's a navigation bar with links for Applications, Kubernetes, Developers, and Company, along with a Sign In button. Below the navigation is a title 'Create your Bitnami Account' and a link to 'Sign In'. The main form is divided into two sections: 'Register with Email' on the left and 'Register with an External Account' on the right. The 'Email' section includes fields for First name (Jamie), Last name (Davis), Email (user@example.com), and Password. The 'External Account' section includes buttons for Register with Google, Register with GitHub, and Register with Microsoft. A central 'or' button indicates that users can choose either method. Below the 'Email' section, there are terms of service and privacy policy links, and checkboxes for accepting terms and receiving newsletters. At the bottom, there's a 'Register' button and a reCAPTCHA field.

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.

Bitnami account registration confirmation

From: hello@bitnami.com, To: [REDACTED], Date: [REDACTED]

Confirm Your Account

Please confirm your account by clicking on the following link:

[https://bitnami.com/confirmation?confirmation_token=\[REDACTED\]](https://bitnami.com/confirmation?confirmation_token=[REDACTED])

If you did not sign up for this account, you can disregard this email and the account will not be created.

Regards,

The Bitnami Team

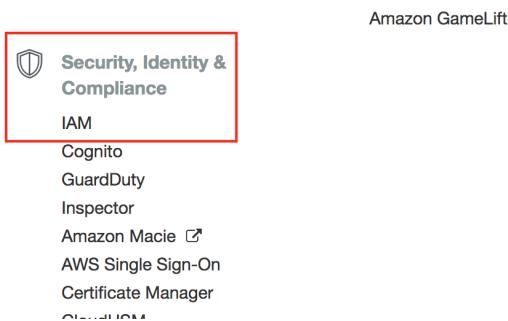
Step 3: Connect your AWS and Bitnami Accounts

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

The easiest way to set up your AWS cloud server with LAMP packaged by Bitnami is via [Bitnami Launchpad for AWS Cloud](#), which gives you a simple control panel to provision, start, stop and check status of your AWS cloud servers. However, to use it, you must first connect your AWS and Bitnami accounts, by obtaining security credentials for your AWS account and saving those credentials in your Bitnami Launchpad account.

Once your AWS account has been activated, the next step is to create an AWS Identity and Access Management (IAM) user and generate an AWS Access Key ID and Secret Access Key. You will need this to connect your AWS account with Bitnami. To do this:

- Log in to the [AWS Console](#).
- In the AWS services menu, scroll down until you see the "Security, Identity & Compliance" section. Select the IAM service.



- Select the "Policies" section in the left navigation bar and click the "Get Started" button.
- Click the "Create Policy" button.

The screenshot shows the AWS IAM Policies page. On the left, there's a navigation bar with links like Dashboard, Groups, Users, Roles, Policies, Identity providers, Account settings, Credential report, and Encryption keys. The 'Policies' link is highlighted. At the top, there's a search bar labeled 'Search IAM' and a 'Create policy' button. Below the search bar is a filter section with dropdowns for 'Policy type', 'Policy name', 'Type', 'Attachments', and 'Description'. There are also 'Show' and 'Filter' buttons.

- On the next page, select the JSON tab and enter the following content:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "sts:GetFederationToken", "ec2:*", "cloudwatch:GetMetricStatistics", "cloudformation:*" ],
      "Resource": [ "*" ]
    }
  ]
}
```

The screenshot shows the 'Create policy' wizard. It has two steps: 'Editor' (Step 1) and 'Review' (Step 2). Step 1 is currently active. A note at the top says: 'A policy defines the AWS permissions that can be assigned to a user, group, role, or resource. You can construct a policy using the visual editor or create a policy document using the JSON editor.' Below this is a 'Visual editor' tab and a 'JSON' tab, which is highlighted with a red box. The JSON code from the previous step is pasted into the editor area. There's also a 'Import managed policy' link.

The screenshot shows the 'Create policy' wizard, Step 2: 'Review'. It has two steps: 'Editor' (Step 1) and 'Review' (Step 2). Step 2 is active. At the top, there are 'Cancel' and 'Review policy' buttons, with 'Review policy' highlighted with a red box. Below this is a 'Name*' field containing 'BitnamiConsole' and a 'Description' field. Further down is a 'Summary' table showing policy details for CloudFormation, CloudWatch, EC2, and STS services. At the bottom are 'Cancel', 'Previous', and 'Create policy' buttons, with 'Create policy' highlighted with a red box.

- Click "Review Policy" to proceed.
- Set the policy name to "BitnamiConsole" and click "Create Policy" to save the new policy.

The screenshot shows the 'Create policy' wizard, Step 2: 'Review'. It has two steps: 'Editor' (Step 1) and 'Review' (Step 2). Step 2 is active. At the top, there are 'Cancel' and 'Review policy' buttons. Below this is a 'Name*' field containing 'BitnamiConsole' and a 'Description' field. Further down is a 'Summary' table showing policy details for CloudFormation, CloudWatch, EC2, and STS services. At the bottom are 'Cancel', 'Previous', and 'Create policy' buttons, with 'Create policy' highlighted with a red box.

- Select the "Users" section in the left navigation bar and click the "Add user" button.

The screenshot shows the AWS IAM 'Users' section. At the top, there are buttons for 'Add user' and 'Delete users'. Below them is a search bar labeled 'Find users by username or access key' and a dropdown menu for 'User name'. On the left sidebar, 'Users' is selected. Other options like 'Dashboard', 'Groups', 'Roles', 'Policies', 'Identity providers', 'Account settings', and 'Credential report' are also listed. A 'Encryption keys' section is at the bottom.

- On the "Details" page, enter a user name for use with Bitnami. Ensure that the "Programmatic access" checkbox in the "Select AWS access type" section is selected. Click the "Next: Permissions" button to proceed.

The screenshot shows the 'Add user' wizard. Step 1: Set user details. It asks for a user name ('bitnami') and provides an option to add another user. Step 2: Select AWS access type. It shows the 'Programmatic access' checkbox checked, which enables an access key ID and secret access key for AWS API, CLI, SDK, and other development tools. Step 3: Next: Permissions. A 'Next: Permissions' button is highlighted.

- On the "Permissions" page, select the option to "Attach existing policies directly". From the list of policies, find the new "BitnamiConsole" policy. Select it and click the "Next: Review" button.

The screenshot shows the 'Add user' wizard. Step 2: Set permissions for 'bitnami'. It offers three options: 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly'. The third option is highlighted with a red box. Below it, a list of policies is shown, with 'BitnamiConsole' selected.

Policy name	Type	Attachments	Description
BitnamiConsole	Customer managed	0	

- On the "Review" page, review the selected options and click the "Create user" button.

The screenshot shows the 'Add user' wizard. Step 3: Review. It displays the user details: 'User name' (bitnami) and 'AWS access type' (Programmatic access - with an access key). It also shows a 'Permissions summary' table with one entry: 'Managed policy' (BitnamiConsole). A 'Create user' button is highlighted.

Type	Name
Managed policy	BitnamiConsole

- A new user and corresponding key pair, consisting of an "Access Key ID" and "Secret Access Key", will be generated and displayed. The "Secret Access Key" value will not be displayed again, so it is important to

accurately note down the "Access Key ID" and "Secret Access Key" values displayed on the screen at this point.

Add user

Success
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.
Users with AWS Management Console access can sign-in at:

User	Access key ID	Secret access key
bitnami	[REDACTED]	[REDACTED] Show

Download .csv

Close

Your IAM user account and access keys are now ready for use.

- Note the Access Key ID and Secret Access Key.

You're now ready to connect AWS with Bitnami. To do this:

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

Bitnami Launchpad for AWS Cloud

Library Sign in with Bitnami

Bitnami Library

Popular Images, provided by Bitnami, ready to launch on AWS Cloud in one click.

What is AWS Cloud?
AWS Cloud enables developers to build, test and deploy applications on AWS's highly-scalable and reliable Infrastructure.

AWS Free Tier 0

All categories Search applications... Search

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

The next step is to set up an administrative password and connect your AWS cloud account with your Bitnami account. To do this:

- Select "Virtual Machines" in the Launchpad menu.

Since this is your first time, you'll be prompted to set up your Bitnami Vault password by entering an administrative password. Enter a hard-to-guess password.

Setup Your Bitnami Vault



Before continuing, we ask that you setup a password for your Bitnami Vault.

This password is independent from your AWS Cloud account and primary Bitnami account, and is used to secure access to sensitive information such as SSH keys or API credentials .

We can't recover it for you, so please be sure to write it down.

Vault password	<input type="password"/>
Vault password confirmation	<input type="password"/>
Save Password	

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

IMPORTANT

Your Bitnami Vault password is different from your Amazon Web Services password.

- Once your password has been accepted, you'll be redirected back to the Launchpad page. Select "Accounts _ Cloud Accounts" in the Launchpad menu.
- Click the "Add Cloud Account" button.

Bitnami Launchpad for AWS Cloud [Virtual Machines](#) [Library](#) [Support ▾](#)

Your Cloud Accounts

Bitnami Launchpad

Name	Virtual Machines	Options
1 hour demo	0	Launch VM

[Add Cloud Account](#)

- You'll be transferred to an authorization page, where you will need to enter the Access Key ID and Secure Access Key. Enter this information and click "Continue" to proceed. An authorization check will now be performed.

Cloud Account setup

Add AWS credentials

Authorization check

Finished

STEP 1 OF 3

Add AWS credentials

To launch Cloud Machines on your behalf we first need your AWS Credentials.

Name: My AWS Account

Access Key ID: ABCDEFGHJKLTEUEYUADAD

Secret Access Key:

These credentials allow us to perform critical background tasks like creating your virtual machines and configuring network access. These credentials are secured in the Bitnami Vault using a vault-specific password.

This [link](#) will take you to your credentials page and allow you to create them.

Please take a look at the information on [how to sign up for Amazon EC2 services](#) and [getting your credentials](#)

Continue

NOTE

If you are using a recently-created AWS account, you may need to wait until your account is verified by AWS before you can complete the process of connecting your AWS Cloud and Bitnami accounts.

Your AWS and Bitnami accounts will now be connected.

Cloud Account setup

Add AWS credentials

Authorization check

Finished

STEP 3 OF 3

You're all done!

You are now ready to launch a VM with the Bitnami Launchpad for AWS Cloud.

Launch an instance [Browse the Bitnami library to get started.](#) **Browse**

Step 4: Provision an AWS Cloud Server

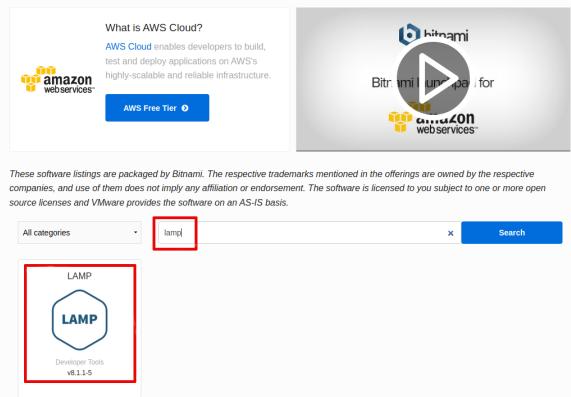
At the end of this step, your AWS cloud server will be running and you will be able to access it through your Web browser.

To provision your AWS cloud server:

- Select "Library" in the [Launchpad](#) menu.
- Look through the list of applications available in Bitnami until you find LAMP Stack. Select it and click "Launch". If required, enter your administrative password.

Bitnami Library

Popular images, provided by [Bitnami](#), ready to launch on [AWS Cloud](#) in one click.

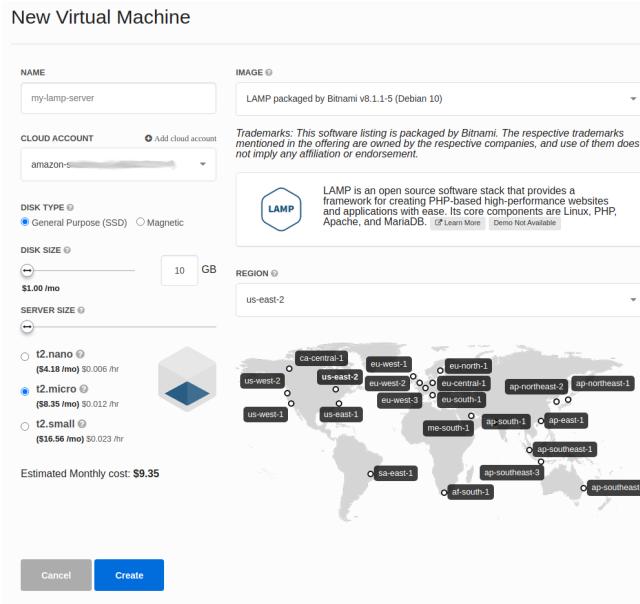


The screenshot shows the Bitnami Library interface. At the top, there's a banner for AWS Cloud with the Bitnami logo and an 'AWS Free Tier' button. Below the banner, a message states: "These software listings are packaged by Bitnami. The respective trademarks mentioned in the offerings are owned by the respective companies, and use of them does not imply any affiliation or endorsement. The software is licensed to you subject to one or more open source licenses and VMware provides the software on an AS-IS basis." A search bar has 'lamp' typed into it, with a red box highlighting the search term. Below the search bar, a list of packages is shown, with the first item, 'LAMP v8.1.1-5', highlighted by a red box.

- Define a name and domain name for your AWS server. The default server configuration is a "Micro" server, 1000 MB RAM and 10 GB EBS storage, which is eligible for the AWS free tier.
- Define a name, size and region for your cloud server. You can choose from a "micro" server, which uses a shared CPU to a "high CPU" server, which has 16 dedicated virtual cores. For more information, refer to the [AWS pricing sheet](#). The default server configuration is a "Micro" server, 1GB RAM and 10 GB EBS storage, which is eligible for the AWS free tier.

TIP

A "micro" server will work just fine for most PHP application development tasks.



The screenshot shows the 'New Virtual Machine' creation form. It includes fields for NAME (my-lamp-server), IMAGE (LAMP packaged by Bitnami v8.1.1-5 (Debian 10)), CLOUD ACCOUNT (amazon-something), DISK TYPE (General Purpose (SSD)), DISK SIZE (10 GB), SERVER SIZE (t2.micro selected), and REGION (us-east-2). The form also displays a map of AWS regions with various availability zones marked. At the bottom, there are 'Cancel' and 'Create' buttons.

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

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

bitnami-lampstack-cf-2b92

The screenshot shows the Bitnami Launchpad interface. At the top, a red box highlights the status bar message "Create virtual machine: Finishing setup 23%". Below this, the "Application Info" section displays a LAMP icon and a brief description of the software stack. The "Server Info" section shows the server configuration: T2.MICRO instance type, \$8.35/MO cost, 10 GB solid state storage, 1 GB memory, US-EAST-2 region, and an estimated monthly cost of \$9.35. A note indicates "CONNECTION NOT AVAILABLE" and provides download keys for PEM and PPK files.

Once the cloud server has been provisioned, the status indicator will show that it's "running", and the Bitnami Launchpad page will display the server details, including its IP address.

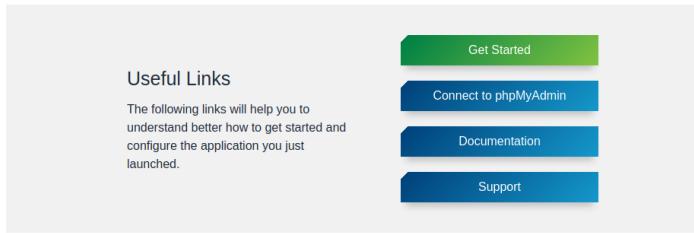
bitnami-lampstack-cf-2b92

The screenshot shows the Bitnami Launchpad interface after the virtual machine has been provisioned. The "Server Info" section now shows the status as "Running 10 minutes ago". The IP address of the server, 130.211.51.215, is highlighted with a red box. The rest of the server configuration remains the same as in the previous screenshot.

At this point, you should be able to browse to the cloud server, either by clicking the link in the Bitnami Launchpad (a new browser tab will open) or entering the cloud server IP address 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).

Congratulations!

You are now running **Bitnami LAMP 8.1.1** in the Cloud.



Once the server is provisioned, you need to gather the security credentials you will need to begin using it. To do this:

- Go back to the Bitnami Launchpad for AWS Cloud page and in the "Virtual Machines" section, select the running server. This will launch the server information page.
- From the server information page, download the **.ppk** file which contains the SSH access credentials you will need to connect to the server. Typically, this file is named using the format *bitnami-[google-project]-[nn].ppk*. If you're using Mac OS X or Linux, you should instead download the corresponding **.pem** file.

The screenshot shows the Bitnami Launchpad interface for a server named "bitnami-lampstack-cf-2b92".
Application Info:

- LAMP 8.1.1-5: A brief description of the LAMP stack.
- GO TO APPLICATION: A button to launch the application in a new window.
- CREDENTIALS: A section showing a password field and port information (PORTS: 80, 443).

Server Info:

- Running 10 minutes ago.
- IP: 130.211.51.215
- T2.MICRO: Instance type, \$8.35/MO (0.012/HR)
- SOLID STATE: 10 GB (\$1.00/MO)
- 1 GB: MEMORY
- US-EAST-2: REGION
- \$9.35: ESTIMATED MONTHLY COST
- DOWNLOAD KEY: .PEM, .PPK (the .PPK link is highlighted with a red box)
- Show SSH command

- By default, Bitnami Launchpad creates a user account named 'user' and an auto-generated password when a new server is provisioned. You will need this password when accessing Bitnami-supplied applications (including MySQL). Go back to the server information screen, look in the "Credentials" section of the "Application Info" panel, and display and make a note of the application password.

The Launchpad page also includes controls to reboot, shut down or delete the server.

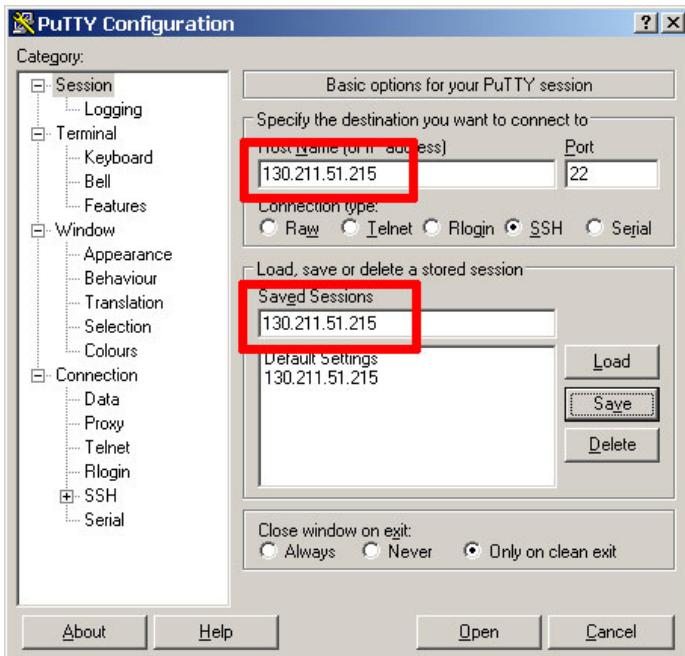
Step 5: Test PHP and MariaDB

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.

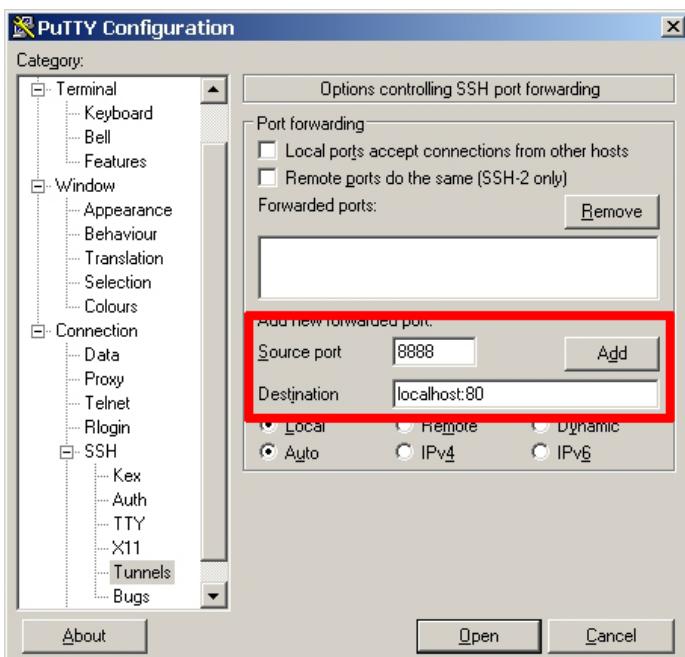
You can now connect to the cloud server and test PHP to make sure it's working correctly and has all the extensions you need. The easiest way to do this is with [PuTTY](#), a free SSH client for Windows and UNIX platforms.

- 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.

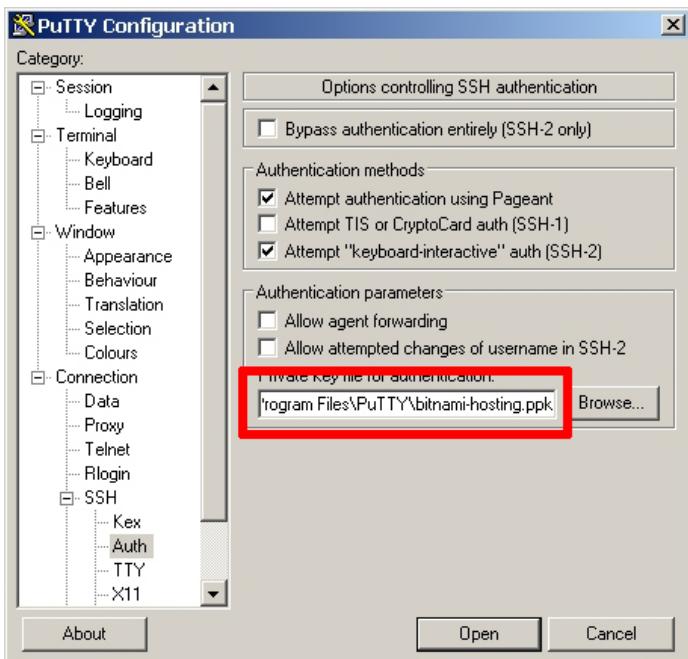
- Enter the host name of your cloud server into the "Host Name (or IP address)" field, as well as into the "Saved Sessions" field.
- Click "Save" to save the new session so you can reuse it later.



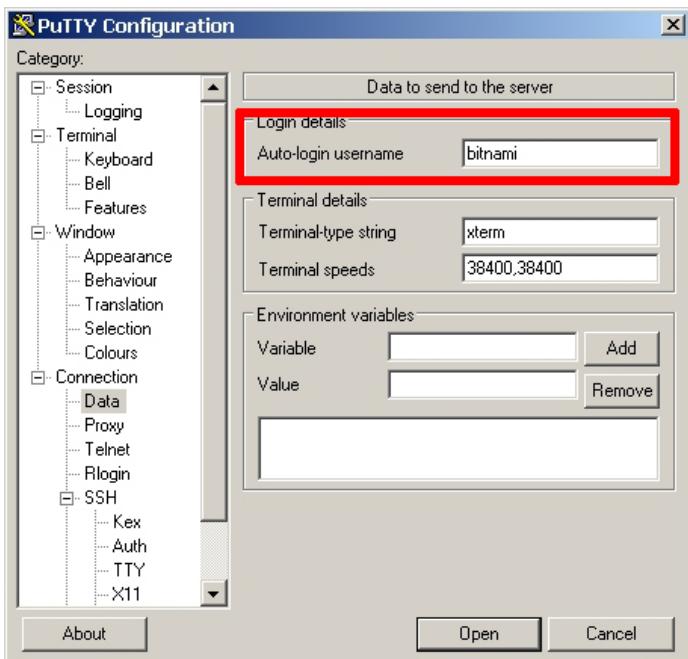
- In the "Connection _ SSH _ Tunnels" section, create a secure tunnel for the phpMyAdmin application by forwarding source port "8888" to destination port "localhost:80".
- Click the "Add" button to add the secure tunnel configuration to the session.



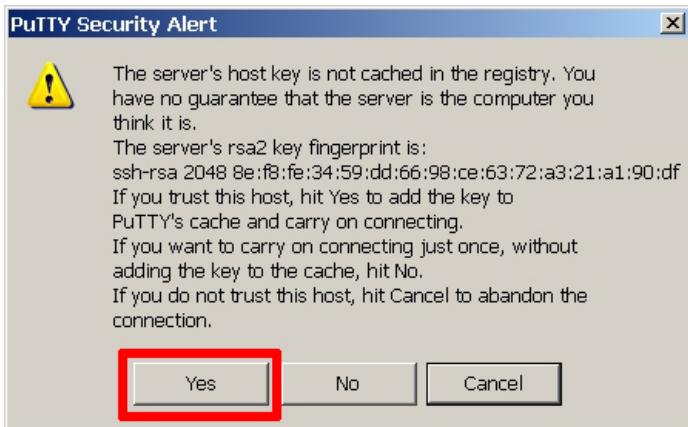
- In the "Connection _ SSH _ Auth" section, select the private key file (*.ppk) you saved in the previous step.



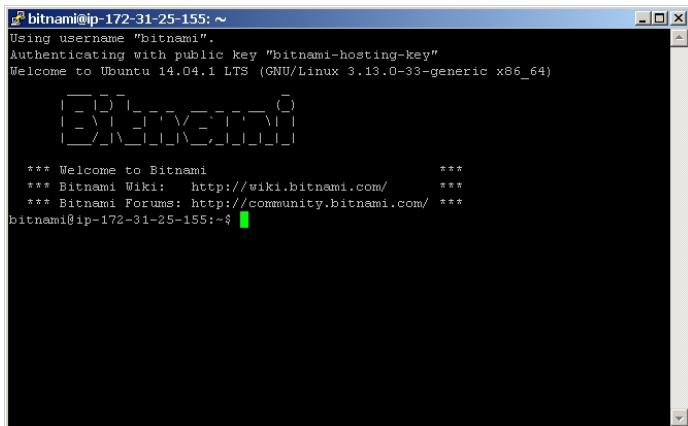
- In the "Connection _ Data" section, enter the username 'bitnami' into the "Auto-login username" field.



- 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.



You should now 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. Your first step should be to create a `phpinfo.php` file in the Apache web server root at `/opt/bitnami/apache2/htdocs` directory to verify PHP's capabilities.

```
shell> cd /opt/bitnami/apache2/htdocs
shell> echo "<?php phpinfo(); ?>" > phpinfo.php
```

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.

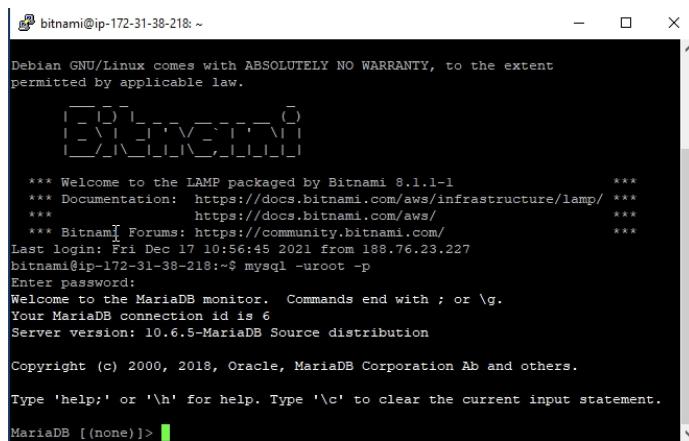
PHP Version 8.1.1	
System	Linux ip-172-31-218-4 4.19.0-18-cloud-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64
Build Date	Dec 15 2021 09:17:11
Build System	Linux 1f7759a13bc 4.9.0-14-amd64 #1 SMP Debian 4.9.246-2 (2020-12-17) x86_64 GNU/Linux
Configure Command	'/bin/sh' '--prefix=/opt/bitnami/php' '--with-imap=/binamiblacksmith-sandboximap-2007.0.0' '--with-imap-sasl' '--with-zlib' '--with-libxml-dir=/usr' '--enable-soap' '--disable-rpath' '--enable-inline-optimization' '--with-bz2' '--enable-sockets' '--enable-pcntl' '--enable-exif' '--enable-bcmath' '--enable-mbstring' '--enable-fpm' '--with-mcrypt' '--with-xmpc' '--with-xsl' '--enable-bom' '--with-fpm-user=demon' '--with-fpm-group=demon' '--enable-mbstring' '--enable-cgi' '--enable-ctype' '--enable-session' '--enable-mysqld' '--enable-mysqlnd' '--with-xml' '--with-pdo_sqlite' '--with-sqlite3' '--with-readline' '--with-gmp' '--with-curl' '--with-openssl' '--with-apache' '--with-zip' '--with-freetype' '--with-gdb' '--with-wsdl' '--with-pdo-odbc' '--with-pdo-mysql' '--enable-gd' '--with-png' '--with-jpeg' '--with-zip' '--with-pdo-dblib' '--shared' '--with-ldap' '--enable-apcu=shared' 'PKG_CONFIG_PATH=/opt/bitnami/common/lib/pkgconfig'

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

You can also check that MariaDB is working by launching the MariaDB command-line client at the shell prompt.

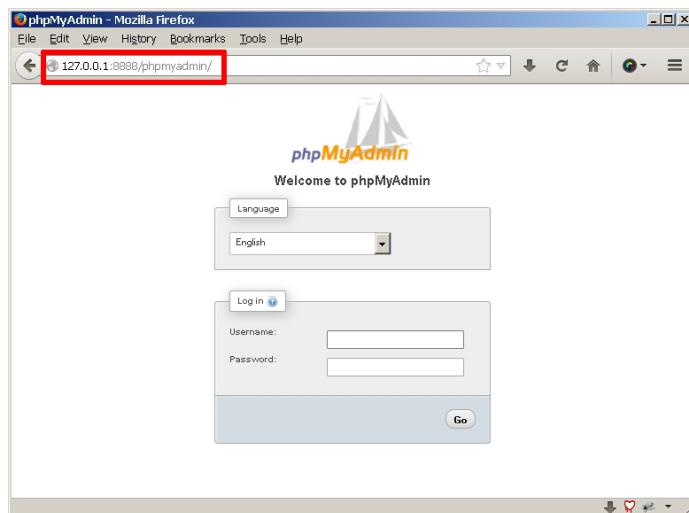
```
shell> mysql -u root -p
```

When prompted, enter the application password retrieved in the previous step. The client should start up and connect to the local MariaDB server, displaying a welcome message as shown below.



```
bitnami@ip-172-31-38-218:~  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
[ - ] [ - ] [ - ] [ - ] [ - ] [ - ]  
*** Welcome to the LAMP packaged by Bitnami 8.1.1-1 ***  
*** Documentation: https://docs.bitnami.com/aws/infrastructure/lamp/ ***  
*** https://docs.bitnami.com/aws/ ***  
*** Bitnami Forums: https://community.bitnami.com/ ***  
Last login: Fri Dec 17 10:56:45 2021 from 188.76.23.227  
bitnami@ip-172-31-38-218:~$ mysql -uroot -p  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 6  
Server version: 10.6.5-MariaDB Source distribution  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]>
```

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' with the application password from the previous step.



The screenshot shows the phpMyAdmin configuration interface. At the top, there's a navigation bar with links for Databases, SQL, Status, Users, Import, Settings, and More. Below the navigation, there are two main sections: "General Settings" and "Database server". The "General Settings" section includes a "Change password" link, a "Server connection collation" dropdown set to "utf8mb4_general_ci", and "Appearance Settings" with language and theme options. The "Database server" section provides detailed information about the MySQL server, including its type (MySQL), version (5.5.38), and connection details. On the left, a sidebar lists recent databases: "New", "information_schema", "mysql", "performance_schema", and "test".

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 'mariadb':

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

```
bitnami@ip-172-31-38-218:~$ sudo /opt/bitnami/ctlscript.sh restart
Restarting services..
bitnami@ip-172-31-38-218:~$ sudo /opt/bitnami/ctlscript.sh restart mariadb
Restarted mariadb
bitnami@ip-172-31-38-218:~$
```

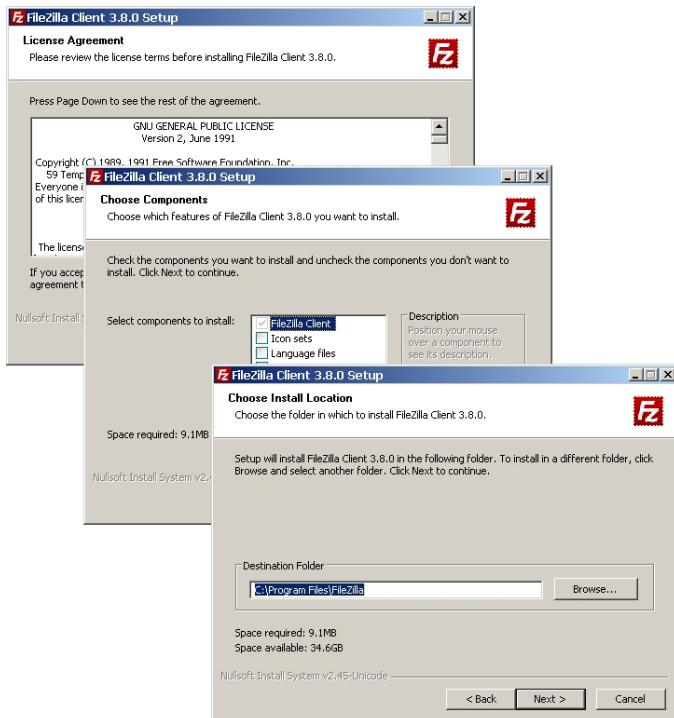
Step 6: Deploy the XAMPP Application to the Cloud Server

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

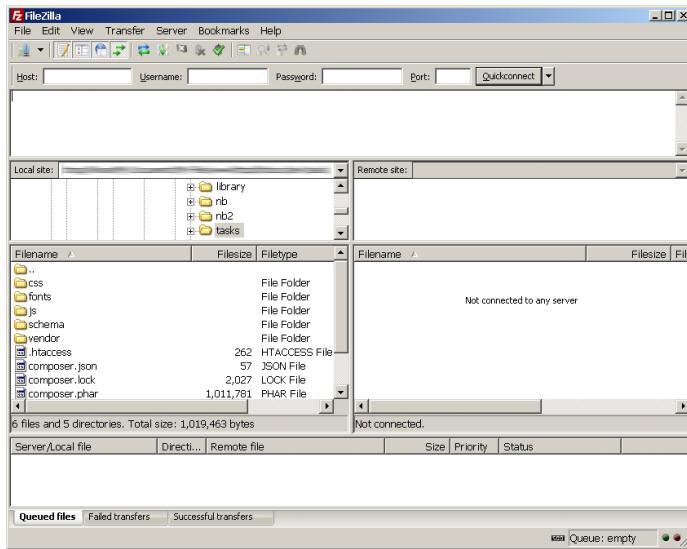
Your cloud server is now provisioned, secured and has a functional PHP/MariaDB environment. All that's left is for you to transfer your application code from your local XAMPP environment to your cloud server and set up the database.

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.

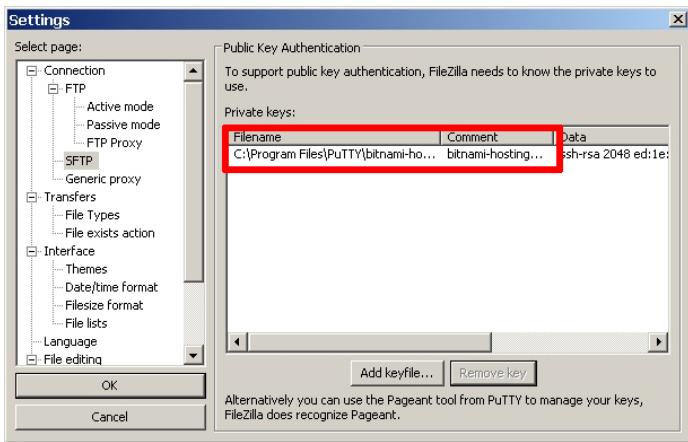


Once FileZilla is installed, launch it and you'll arrive at the main split-screen interface, one side for your local directories and the other for remote directories.

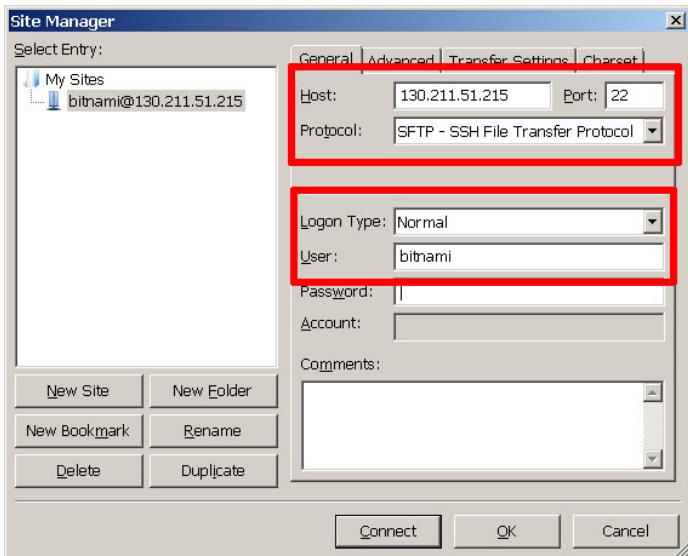


To connect to the cloud server and deploy your application, follow these steps:

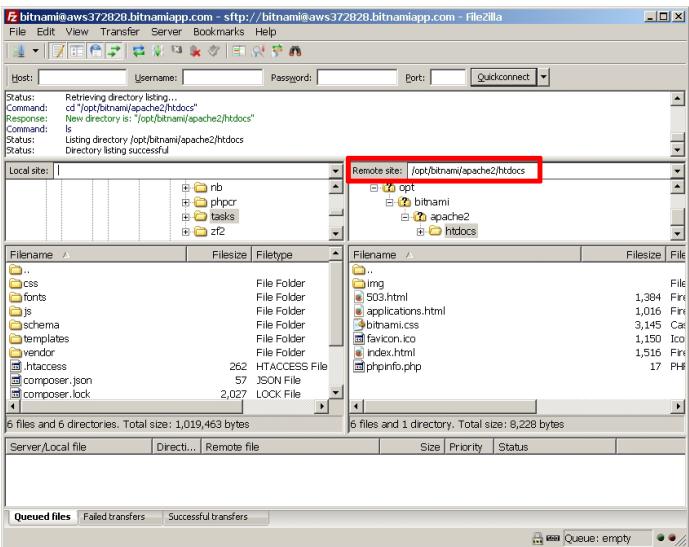
- Use the "Edit → Settings" command to bring up FileZilla's configuration settings.
- Within the "Connection → SFTP" section, use the "Add keyfile" command to select the private key file for your server. FileZilla will use this private key to log in to the cloud server.



- 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';
```

```
bitnami@ip-172-31-38-218:~$ sudo /opt/bitnami/ctlscript.sh restart
Restarting services...
bitnami@ip-172-31-38-218:~$ sudo /opt/bitnami/ctlscript.sh restart mariadb
Restarted mariadb
bitnami@ip-172-31-38-218:~$ mysql -uroot -P
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.6.5-MariaDB Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help,' or '\h' for help. Type '\c' to clear the current input statement.

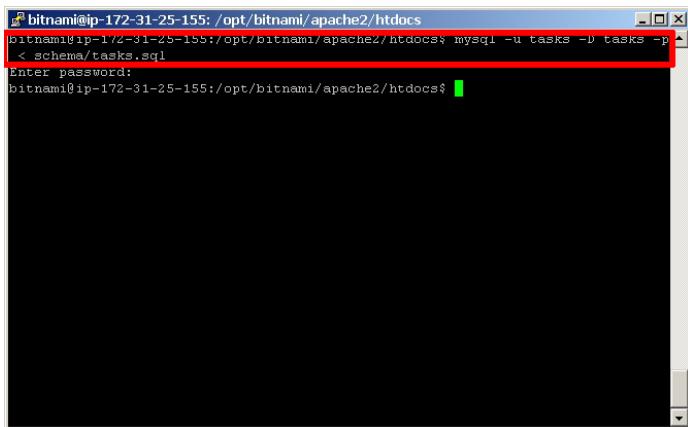
MariaDB [(none)]> CREATE DATABASE tasks;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL ON tasks.* TO 'tasks'@'localhost' IDENTIFIED BY 'klio89';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]>
```

- 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
```



```

bitnami@ip-172-31-25-155:/opt/bitnami/apache2/htdocs$ mysql -u tasks -p < schema/tasks.sql
Enter password:
bitnami@ip-172-31-25-155:/opt/bitnami/apache2/htdocs$ 
```

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.



Importing into the current server

File to Import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **[format] [compression]**. Example: **.sql.zip**

Browse your computer: No file chosen (Max: 80MB)

Character set of the file:

Partial Import:

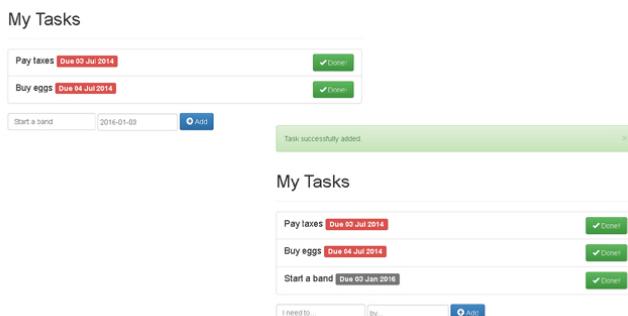
Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. (This is useful for large files, however it can break transactions.)

Skip this number of queries (for SQL) or lines (for other formats), starting from the first one:

Format:

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.



My Tasks

Pay taxes	Due 03 Jul 2014	<input checked="" type="checkbox"/> Done!
Buy eggs	Due 04 Jul 2014	<input checked="" type="checkbox"/> Done!
Start a band	Due 02 Jan 2016	<input checked="" type="checkbox"/> Done!

Task successfully added.

My Tasks

Pay taxes	Due 03 Jul 2014	<input checked="" type="checkbox"/> Done!
Buy eggs	Due 04 Jul 2014	<input checked="" type="checkbox"/> Done!
Start a band	Due 02 Jan 2016	<input checked="" type="checkbox"/> Done!

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

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

- [Amazon Web Services](#)

- Bitnami Launchpad for AWS Cloud
- 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/>.