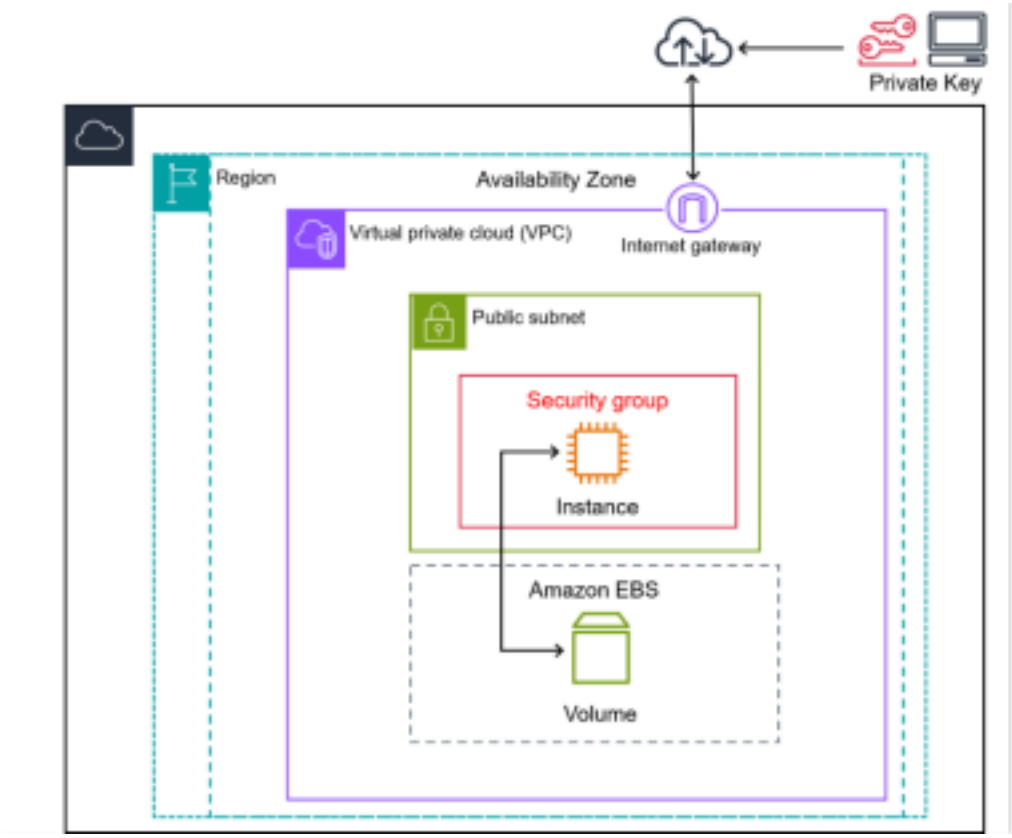
Amazon Elastic Compute Cloud: User Guide for Linux Instances

Table of Contents

What is Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.

The following diagram shows a basic architecture of an Amazon EC2 instance deployed within an Amazon Virtual Private Cloud (VPC). In this example, the EC2 instance is within an Availability Zone in the Region. The EC2 instance is secured with a security group, which is a virtual firewall that controls incoming and outgoing traffic. A private key is stored on the local computer and a public key is stored on the instance. Both keys are specified as a key pair to prove the identity of the user. In this scenario, the instance is backed by an Amazon EBS volume. The VPC communicates with the internet using an internet gateway. For more information about Amazon VPC, see the Amazon VPC User Guide.



Tip

This user guide provides information specific to running Linux-based instances on Amazon EC2. See the EC2 User Guide for Windows Instances for information to help you run Windows-based instances on EC2.

Amazon EC2 supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1.

If you are looking for technical guidance about Amazon EC2, try AWS re:Post.

For more information about cloud computing, see What is cloud computing?

Topics

Features of Amazon EC2

Amazon EC2 provides the following high-level features:

Instances

Virtual servers.

Amazon Machine Images (AMIs)

Preconfigured templates for your instances that package the components you need for your server (including the operating system and additional software).

Instance types

Various configurations of CPU, memory, storage, networking capacity, and graphics hardware for your instances.

Key pairs

Secure login information for your instances. AWS stores the public key and you store the private key in a secure place.

Instance store volumes

Storage volumes for temporary data that is deleted when you stop, hibernate, or terminate your instance.

Amazon EBS volumes

Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS).

Regions, Availability Zones, Local Zones, AWS Outposts, and Wavelength Zones

Multiple physical locations for your resources, such as instances and Amazon EBS volumes. **Security groups**

A virtual firewall that allows you to specify the protocols, ports, and source IP ranges that can reach your instances, and the destination IP ranges to which your instances can connect.

Elastic IP addresses

Static IPv4 addresses for dynamic cloud computing.

Tags

Metadata that you can create and assign to your Amazon EC2 resources.

Virtual private clouds (VPCs)

Virtual networks you can create that are logically isolated from the rest of the AWS Cloud. You can optionally connect these virtual networks to your own network.

For details about all of the features of Amazon EC2, see Amazon EC2 features.

For options to run your website on AWS, see Web Hosting.

Get started with Amazon EC2

The following topics can help you get started with Amazon EC2. After you set up to use EC2, you can walk through Tutorial: Get started with Amazon EC2 Linux instances to launch, connect to, and

clean up an instance. The remaining topics point to more information about the high-level features of EC2.

Set up and use an EC2 instance

- Set up to use Amazon EC2
- Tutorial: Get started with Amazon EC2 Linux instances
- Connect to your Linux instance
- Transfer files

Learn the basics of Amazon EC2

- Instances and AMIs
- Regions and Zones
- Instance types
- Tags

Read about networking and security

- Key pairs
- Security groups

- Elastic IP addresses
- Virtual private clouds

Review your storage options

- Amazon EBS
- Instance store

Walk through a Linux tutorial

- Remotely Run Commands on an EC2 Instance with AWS Systems Manager • Install LAMP on Amazon Linux 2
- Configure SSL/TLS on Amazon Linux 2
- Host a WordPress blog

- Troubleshoot EC2 instances
- AWS re:Post

Related services

You can provision Amazon EC2 resources, such as instances and volumes, directly using Amazon EC2. In addition, you can provision EC2 resources using other AWS services, such as the following:

- Amazon EC2 Auto Scaling

  Helps ensure you have the correct number of Amazon EC2 instances available to handle the load for your application.

- AWS CloudFormation

  Helps you model and set up your AWS resources using templates.

- AWS Elastic Beanstalk

  Deploy and manage applications in the AWS Cloud without having to understand the underlying infrastructure.

- AWS OpsWorks

  Automate how servers are configured, deployed, and managed across your Amazon EC2 instances using Chef and Puppet.

- EC2 Image Builder

  Automate the creation, management, and deployment of customized, secure, and up-to-date server images.

- AWS Launch Wizard

  Size, configure, and deploy AWS resources for third-party applications without having to manually identify and provision individual AWS resources.

Additional related services

- Amazon Lightsail

  To build websites or web applications, you can deploy and manage basic cloud resources using Amazon Lightsail. To compare the features of Amazon EC2 and Lightsail for your use case, see Amazon Lightsail or Amazon EC2.

- Elastic Load Balancing

  Automatically distribute incoming application traffic across multiple instances. •

Amazon Relational Database Service (Amazon RDS)

  Set up, operate, and scale a managed relational database in the cloud. Although you can set up a database on an EC2 instance, Amazon RDS offers the advantage of handling your database management tasks, such as patching the software, backing up, and storing the backups.

- Amazon Elastic Container Service (Amazon ECS)

  Deploy, manage, and scale containerized applications on a cluster of EC2 instances.

- Amazon Elastic Kubernetes Service (Amazon EKS)

  Run your Kubernetes applications on AWS.

- Amazon CloudWatch

  Monitor your instances and Amazon EBS volumes.

- Amazon GuardDuty

Detect potentially unauthorized or malicious use of your EC2 instances.

- AWS Backup

  Automate backing up your Amazon EC2 instances and the Amazon EBS volumes attached to them.

Access Amazon EC2

You can create and manage your Amazon EC2 instances using the following interfaces:

Amazon EC2 console

A simple web interface to create and manage Amazon EC2 instances and resources. If you've signed up for an AWS account, you can access the Amazon EC2 console by signing into the AWS Management Console and selecting **EC2** from the console home page.

AWS Command Line Interface

Enables you to interact with AWS services using commands in your command-line shell. It is supported on Windows, Mac, and Linux. For more information about the AWS CLI , see AWS Command Line Interface User Guide. You can find the Amazon EC2 commands in the AWS CLI Command Reference.

AWS Tools for PowerShell

A set of PowerShell modules that are built on the functionality exposed by the AWS SDK for .NET. The Tools for PowerShell enable you to script operations on your AWS resources from the PowerShell command line. To get started, see the AWS Tools for Windows PowerShell User Guide. You can find the cmdlets for Amazon EC2, in the AWS Tools for PowerShell Cmdlet Reference.

AWS CloudFormation

Amazon EC2 supports creating resources using AWS CloudFormation. You create a template, in JSON or YAML format, that describes your AWS resources, and AWS CloudFormation provisions and configures those resources for you. You can reuse your CloudFormation templates to provision the same resources multiple times, whether in the same Region and account or in multiple Regions and accounts. For more information about supported resource types and properties for Amazon EC2, see EC2 resource type reference in the *AWS CloudFormation User Guide*.

Query API

Amazon EC2 provides a Query API. These requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named Action. For more information about the API actions for Amazon EC2, see Actions in the *Amazon EC2 API Reference*. **AWS SDKs**

If you prefer to build applications using language-specific APIs instead of submitting a request over HTTP or HTTPS, AWS provides libraries, sample code, tutorials, and other resources for software developers. These libraries provide basic functions that automate tasks such as cryptographically signing your requests, retrying requests, and handling error responses, making it easier for you to get started. For more information, see Tools to Build on AWS.

Pricing for Amazon EC2

Amazon EC2 provides the following pricing options:

Free Tier

You can get started with Amazon EC2 for free. To explore the Free Tier options, see AWS Free Tier.

On-Demand Instances

Pay for the instances that you use by the second, with a minimum of 60 seconds, with no long term commitments or upfront payments.

Savings Plans

You can reduce your Amazon EC2 costs by making a commitment to a consistent amount of usage, in USD per hour, for a term of 1 or 3 years.

Reserved Instances

You can reduce your Amazon EC2 costs by making a commitment to a specific instance configuration, including instance type and Region, for a term of 1 or 3 years. **Spot Instances**

Request unused EC2 instances, which can reduce your Amazon EC2 costs significantly. **Dedicated Hosts**

Reduce costs by using a physical EC2 server that is fully dedicated for your use, either On Demand or as part of a Savings Plan. You can use your existing server-bound software licenses and get help meeting compliance requirements.

On-Demand Capacity Reservations

Reserve compute capacity for your EC2 instances in a specific Availability Zone for any duration of time.

Per-second billing

Removes the cost of unused minutes and seconds from your bill.

For a complete list of charges and prices for Amazon EC2 and more information about the purchase models, see Amazon EC2 pricing.

Estimates, billing, and cost optimization

To create estimates for your AWS use cases, use the AWS Pricing Calculator. Estimates, billing, and cost optimization 8

To see your bill, go to the **Billing and Cost Management Dashboard** in the AWS Billing and Cost Management console. Your bill contains links to usage reports that provide details about your bill. To learn more about AWS account billing, see AWS Billing and Cost Management User Guide.

If you have questions concerning AWS billing, accounts, and events, contact AWS Support.

To calculate the cost of a sample provisioned environment, see Cloud Economics Center. When calculating the cost of a provisioned environment, remember to include incidental costs such as snapshot storage for EBS volumes.

You can optimize the cost, security, and performance of your AWS environment using AWS Trusted Advisor.

Set up to use Amazon EC2

Complete the tasks in this section to get set up for launching an Amazon EC2 instance for the first time:

1. Sign up for an AWS account
2. Create an administrative user
3. Create a key pair
4. Create a security group

When you are finished, you will be ready for the Amazon EC2 Getting started tutorial.

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create an administrative user

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User*

Guide.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create an administrative user

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to an administrative user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the administrative user

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Create a key pair

AWS uses public-key cryptography to secure the login information for your instance. A Linux instance has no password; you use a key pair to log in to your instance securely. You specify the name of the key pair when you launch your instance, then provide the private key when you log in using SSH.

If you haven't created a key pair already, you can create one by using the Amazon EC2 console. Note that if you plan to launch instances in multiple AWS Regions, you'll need to create a key pair in each Region. For more information about Regions, see Regions and Zones.

To create your key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2.

In the navigation pane, choose **Key Pairs**.

3. Choose **Create key pair**.

4. For **Name**, enter a descriptive name for the key pair. Amazon EC2 associates the public key with the name that you specify as the key name. A key name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

5. For **Key pair type**, choose either **RSA** or **ED25519**. Note that **ED25519** keys are not supported for Windows instances.

6. For **Private key file format**, choose the format in which to save the private key. To save the private key in a format that can be used with OpenSSH, choose **pem**. To save the private key in a format that can be used with PuTTY, choose **ppk**.

7. Choose **Create key pair**.

8. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is determined by the file format you chose. Save the private key file in a safe place.

> **Important**
>
> This is the only chance for you to save the private key file.

9. If you plan to use an SSH client on a macOS or Linux computer to connect to your Linux instance, use the following command to set the permissions of your private key file so that only you can read it.

 chmod 400 *key-pair-name*.pem

If you do not set these permissions, then you cannot connect to your instance using this key pair. For more information, see Error: Unprotected private key file.

For more information, see Amazon EC2 key pairs and Linux instances.

Create a security group

Security groups act as a firewall for associated instances, controlling both inbound and outbound traffic at the instance level. You must add rules to a security group that enable you to connect to your instance from your IP address using SSH. You can also add rules that allow inbound and

outbound HTTP and HTTPS access from anywhere.

Note that if you plan to launch instances in multiple AWS Regions, you'll need to create a security group in each Region. For more information about Regions, see Regions and Zones.

Prerequisites

You'll need the public IPv4 address of your local computer. The security group editor in the Amazon EC2 console can automatically detect the public IPv4 address for you. Alternatively, you can use the search phrase "what is my IP address" in an internet browser, or use the following service: Check IP. If you are connecting through an Internet service provider (ISP) or from behind a firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

You can create a custom security group using one of the following methods. New console

To create a security group with least privilege

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2. From the top navigation bar, select an AWS Region for the security group. Security groups are specific to a Region, so you should select the same Region in which you created your key pair.
3. In the left navigation pane, choose **Security Groups**.
4. Choose **Create security group**.

 5. For **Basic details**, do the following:

a. Enter a name for the new security group and a description. Use a name that is easy for you to remember, such as your user name, followed by _SG_, plus the Region name. For example, *me*_SG_*uswest2*.

b. In the **VPC** list, select your default VPC for the Region.

6. For **Inbound rules**, create rules that allow specific traffic to reach your instance. For example, use the following rules for a web server that accepts HTTP and HTTPS traffic. For more examples, see Security group rules for different use cases.

a. Choose **Add rule**. For **Type**, choose **HTTP**. For **Source**, choose **Anywhere**. b.

Choose **Add rule**. For **Type**, choose **HTTPS**. For **Source**, choose **Anywhere**. c.

Choose **Add rule**. For **Type**, choose **SSH**. For **Source**, do one of the following:

- Choose **My IP** to automatically add the public IPv4 address of your local computer.
- Choose **Custom** and specify the public IPv4 address of your computer or network

in CIDR notation. To specify an individual IP address in CIDR notation, add the routing suffix /32, for example, 203.0.113.25/32. If your company or your router allocates addresses from a range, specify the entire range, such as 203.0.113.0/24.

Warning

For security reasons, do not choose **Anywhere** for **Source** with a rule for SSH. This would allow access to your instance from all IP addresses on the internet. This is acceptable for a short time in a test environment, but it is unsafe for production environments.

7. For **Outbound rules**, keep the default rule, which allows all outbound traffic. 8.

Choose **Create security group**.

Old console

To create a security group with least privilege

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2. In the left navigation pane, choose **Security Groups**.

3. Choose **Create Security Group**.

4. Enter a name for the new security group and a description. Use a name that is easy for you to remember, such as your user name, followed by _SG_, plus the Region name. For example, *me*_SG_*uswest2*.

5. In the **VPC** list, select your default VPC for the Region.

6. On the **Inbound rules** tab, create the following rules (choose **Add rule** for each new rule):

- Choose **HTTP** from the **Type** list, and make sure that **Source** is set to **Anywhere** (0.0.0.0/0).
- Choose **HTTPS** from the **Type** list, and make sure that **Source** is set to **Anywhere** (0.0.0.0/0).
- Choose **SSH** from the **Type** list. In the **Source** box, choose **My IP** to automatically populate the field with the public IPv4 address of your local computer. Alternatively, choose **Custom** and specify the public IPv4 address of your computer or network in CIDR notation. To specify an individual IP address in CIDR notation, add the routing suffix /32, for example, 203.0.113.25/32. If your company allocates addresses from a range, specify the entire

range, such as 203.0.113.0/24.

Warning

For security reasons, do not allow SSH access from all IP addresses to your instance. This is acceptable for a short time in a test environment, but it is unsafe for production environments.

7. On the **Outbound rules** tab, keep the default rule, which allows all outbound traffic. 8. Choose **Create security group**.

AWS CLI

When you use the AWS CLI to create a security group, an outbound rule that allows all outbound traffic is automatically added to the security group. An inbound rule isn't automatically added; you'll need to add it.

In this procedure, you'll combine the create-security-group and authorize-security-group ingress AWS CLI commands to create the security group and add the inbound rule that allows

the specified inbound traffic. An alternative to the following procedure is to run the commands separately, first creating a security group, and then adding an inbound rule to the security group.

To create a security group and add an inbound rule to the security group

Use the create-security-group and authorize-security-group-ingress AWS CLI commands as follows:

```
aws ec2 authorize-security-group-ingress \
--region us-west-2 \
--group-id $(aws ec2 create-security-group \
--group-name myname_SG_uswest2 \
--description "Security group description" \
--vpc-id vpc-12345678 \
--output text \
--region us-west-2) \
--ip-permissions \

IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges='[{CidrIp=0.0.0.0/0,Description="HTTP   from anywhere"}]' \

IpProtocol=tcp,FromPort=443,ToPort=443,IpRanges='[{CidrIp=0.0.0.0/0,Description="HTTPS   from anywhere"}]' \
```

```
    IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges='[{CidrIp=172.31.0.0/16,Description="SSH from private network"}]'
  \

    IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges='[{CidrIp=203.0.113.25/32,Description="SSH from public IP"}]'
```

For:

- --region – Specify the Region in which to create the inbound rules.
- --group-id – Specify the create-security-group command and the following parameters to create the security group:
  - --group-name – Specify a name for the new security group. Use a name that is easy for you to remember, such as your user name, followed by _SG_, plus the Region name. For example, myname_SG_uswest2.
  - --description – Specify a description that will help you know what traffic the security group allows.

  - --vpc-id – Specify your default VPC for the Region.
  - --output – Specify text as the output format for the command.
  - --region – Specify the Region in which to create the security group. It should be the same Region that you specified for the inbound rules.
- --ip-permissions – Specify the inbound rules to add to the security group. The rules in this example are for a web server that accepts HTTP and HTTPS traffic from anywhere, and that accepts SSH traffic from a private network (if your company or your router allocates addresses from a range) and a specified public IP address (such as the public IPv4 address of your computer or network in CIDR notation).

> **Warning**
>
> For security reasons, do not specify 0.0.0.0/0 for CidrIp with a rule for SSH. This would allow access to your instance from all IP addresses on the internet. This is acceptable for a short time in a test environment, but it is unsafe for production environments.

PowerShell

When you use the AWS Tools for Windows PowerShell to create a security group, an outbound rule that allows all outbound traffic is automatically added to the security group. An inbound rule isn't automatically added; you'll need to add it.

In this procedure, you'll combine the New-EC2SecurityGroup and Grant EC2SecurityGroupIngress AWS Tools for Windows PowerShell commands to create the security group and add the inbound rule that allows the specified inbound traffic. An alternative to the following procedure is to run the commands separately, first creating a security group, and then adding an inbound rule to the security group.

To create a security group

Use the New-EC2SecurityGroup and Grant-EC2SecurityGroupIngress AWS Tools for Windows PowerShell commands as follows.

```
Import-Module AWS.Tools.EC2
New-EC2SecurityGroup -GroupName myname_SG_uswest2 -Description 'Security group   description'
-VpcId vpc-12345678 -Region us-west-2 | `
 Grant-EC2SecurityGroupIngress `
```



```
-GroupName $_ `
-Region us-west-2 `
-IpPermission @(
(New-Object -TypeName Amazon.EC2.Model.IpPermission -Property @{   IpProtocol = 'tcp';
FromPort = 80;
ToPort = 80;
Ipv4Ranges = @(@{CidrIp = '0.0.0.0/0'; Description = 'HTTP from   anywhere'})
}),
(New-Object -TypeName Amazon.EC2.Model.IpPermission -Property @{   IpProtocol = 'tcp';
FromPort = 443;
ToPort = 443;
Ipv4Ranges = @(@{CidrIp = '0.0.0.0/0'; Description = 'HTTPS from   anywhere'})
}),
(New-Object -TypeName Amazon.EC2.Model.IpPermission -Property @{   IpProtocol = 'tcp';
FromPort = 3389;
ToPort = 3389;
Ipv4Ranges = @(
@{CidrIp = '172.31.0.0/16'; Description = 'RDP from private   network'},
@{CidrIp = '203.0.113.25/32'; Description = 'RDP from public   IP'}
)
})
)
```

For the security group:

- -GroupName – Specify a name for the new security group. Use a name that is easy for you to remember, such as your user name, followed by _SG_, plus the Region name. For example,

myname_SG_uswest2.

- -Description – Specify a description that will help you know what traffic the security group allows.
- -VpcId – Specify your default VPC for the Region.
- -Region – Specify the Region in which to create the security group.

For the inbound rules:

• -GroupName – Specify $_ to reference the security group you're creating. • -Region – Specify the Region in which to create the inbound rules. It should be the same Region that you specified for the security group.

• -IpPermission – Specify the inbound rules to add to the security group. The rules in this example are for a web server that accepts HTTP and HTTPS traffic from anywhere, and that accepts RDP traffic from a private network (if your company or your router allocates addresses from a range) and a specified public IP address (such as the public IPv4 address of your computer or network in CIDR notation).

Warning

For security reasons, do not specify 0.0.0.0/0 for CidrIp with a rule for RDP. This would allow access to your instance from all IP addresses on the internet. This is acceptable for a short time in a test environment, but it is unsafe for production environments.

For more information, see Amazon EC2 security groups for Linux instances. Create a security

group 19

Tutorial: Get started with

Amazon EC2 Linux instances

Use this tutorial to get started with Amazon Elastic Compute Cloud (Amazon EC2). You'll learn how to launch, connect to, and use a Linux instance. An *instance* is a virtual server in the AWS Cloud. With Amazon EC2, you can set up and configure the operating system and applications that run on your instance.

When you sign up for AWS, you can get started with Amazon EC2 using the AWS Free Tier. If you created your AWS account less than 12 months ago, and have not already exceeded the Free Tier benefits for Amazon EC2, it won't cost you anything to complete this tutorial because we help you select options that are within the Free Tier benefits. Otherwise, you'll incur the standard Amazon EC2 usage fees from the time that you launch the instance until you terminate the instance (which is the final task of this tutorial), even if it remains idle.

Related tutorials

- If you'd prefer to launch a Windows instance, see this tutorial in the *Amazon EC2 User Guide for Windows Instances*: Get started with Amazon EC2 Windows instances.
- If you'd prefer to use the command line, see this tutorial in the *AWS Command Line Interface User Guide*: Using Amazon EC2 through the AWS CLI.

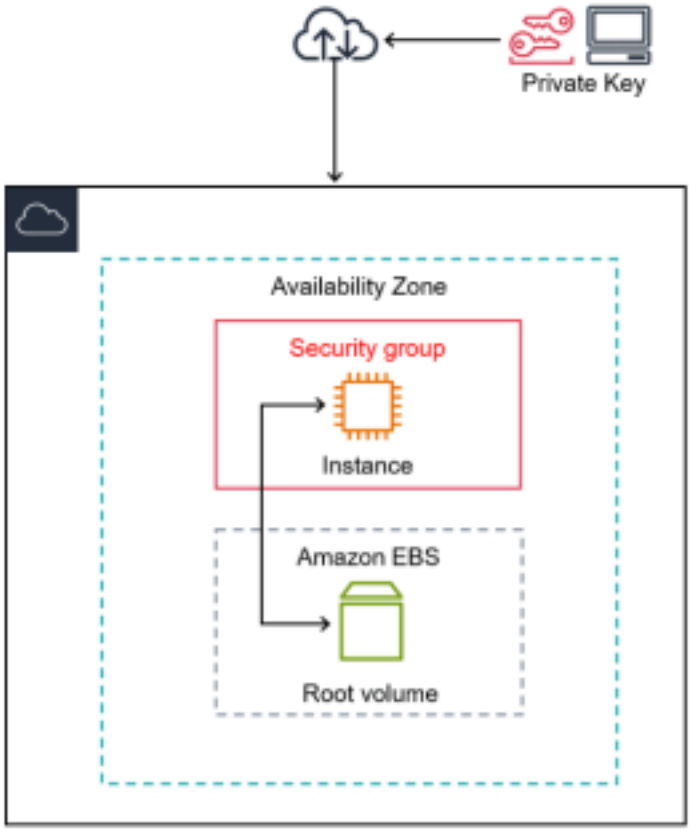
Contents

Overview

The instance launched in this tutorial is an Amazon EBS-backed instance (meaning that the root volume is an EBS volume). You can either specify the Availability Zone in which your instance runs,

or let Amazon EC2 select an Availability Zone for you. Availability Zones are multiple, isolated locations within each AWS Region. You can think of an Availability Zone as an isolated data center.

When you launch your instance, you secure it by specifying a key pair (to prove your identity) and a security group (which acts as a virtual firewall to control ingoing and outgoing traffic). When you connect to your instance, you must provide the private key of the key pair that you specified when you launched your instance.



Prerequisites

Before you begin, be sure that you've completed the steps in Set up to use Amazon EC2.

Step 1: Launch an instance

You can launch a Linux instance using the AWS Management Console as described in the following procedure. This tutorial is intended to help you quickly launch your first instance, so it doesn't cover all possible options. For information about advanced options, see Launch an instance using the new launch instance wizard. For information about other ways to launch your instance, see Launch your instance.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2. From the EC2 console dashboard, in the **Launch instance** box, choose **Launch instance**. 3. Under **Name and tags**, for **Name**, enter a descriptive name for your instance. 4. Under **Application and OS Images (Amazon Machine Image)**, do the following:

a. Choose **Quick Start**, and then choose Amazon Linux. This is the operating system (OS) for your instance.

b. From **Amazon Machine Image (AMI)**, select an HVM version of Amazon Linux 2. Notice that these AMIs are marked **Free Tier eligible**. An *Amazon Machine Image (AMI)* is a basic configuration that serves as a template for your instance.

> **Note**
>
> AL2023 is the successor to Amazon Linux 2. For more information, see Launching AL2023 using the Amazon EC2 console.

5. Under **Instance type**, from the **Instance type** list, you can select the hardware configuration for your instance. Choose the t2.micro instance type, which is selected by default. The t2.micro instance type is eligible for the Free Tier. In Regions where t2.micro is unavailable, you can use a t3.micro instance under the Free Tier. For more information, see AWS Free Tier.
6. Under **Key pair (login)**, for **Key pair name**, choose the key pair that you created when getting set up.

> **Warning**
>
> Do not choose **Proceed without a key pair (Not recommended)**. If you launch your instance without a key pair, then you can't connect to it.

7. Next to **Network settings**, choose **Edit**. For **Security group name**, you'll see that the wizard created and selected a security group for you. You can use this security group, or alternatively you can select the security group that you created when getting set up using the following steps:

a. Choose **Select existing security group**.

b. From **Common security groups**, choose your security group from the list of existing security groups.

8. Keep the default selections for the other configuration settings for your instance.

9. Review a summary of your instance configuration in the **Summary** panel, and when you're ready, choose **Launch instance**.
10. A confirmation page lets you know that your instance is launching. Choose **View all instances** to close the confirmation page and return to the console.
11. On the **Instances** screen, you can view the status of the launch. It takes a short time for an instance to launch. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running and it receives a public DNS name. If the **Public IPv4 DNS** column is hidden, choose the settings icon ( ) in the top-right corner, toggle on **Public IPv4 DNS**, and choose **Confirm**.
12. It can take a few minutes for the instance to be ready for you to connect to it. Check that your instance has passed its status checks; you can view this information in the **Status check** column.

Step 2: Connect to your instance

There are several ways to connect to your Linux instance. For more information, see Connect to your Linux instance.

Important

You can't connect to your instance unless you launched it with a key pair for which you have the .pem file and you launched it with a security group that allows SSH access from your computer. If you can't connect to your instance, see Troubleshoot connecting to your instance for assistance.

Step 3: Track your Free Tier usage

You can use Amazon EC2 without incurring charges if you've been an AWS customer for less than 12 months and you stay within the Free Tier usage limits. It's important to track your Free Tier

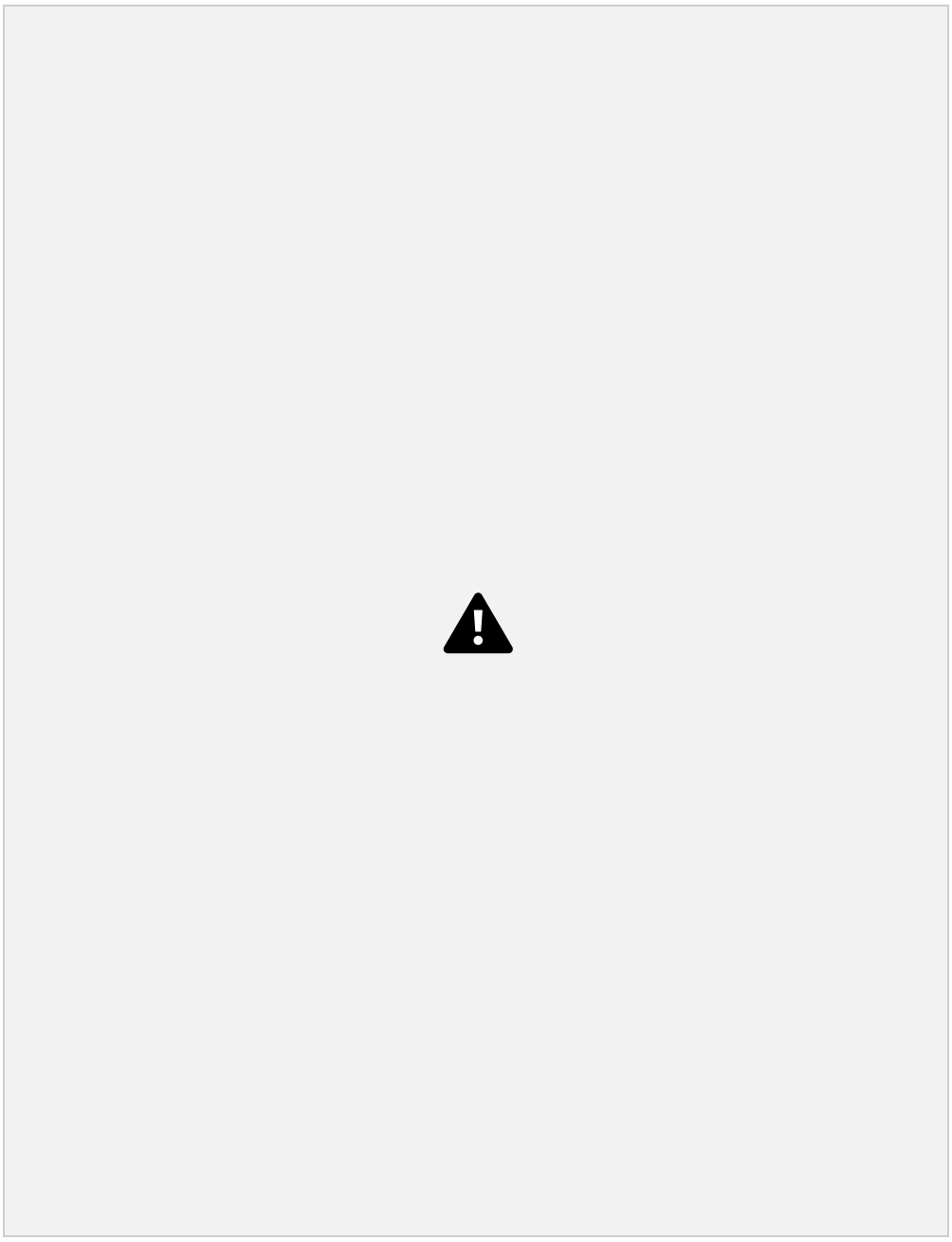
usage to avoid billing surprises. If you exceed the Free Tier limits, you'll incur standard pay-as-go charges.

Note

If you've been an AWS customer for more than 12 months, you're no longer eligible for Free Tier usage and you won't see the **EC2 Free Tier** box that is described in the following procedure.

To track your Free Tier usage

1. In the navigation pane, choose **EC2 Dashboard**.
2. Find the **EC2 Free Tier** box (at top right).



3. In the **EC2 Free Tier** box, check your Free Tier usage, as follows:

   • Under **EC2 Free Tier offers in use**, take note of the warnings:

     • **End of month forecast** – This warns that you will incur charges this month if you continue with your current usage pattern.

     • **Exceeds Free Tier** – This warns that you've exceeded your Free Tier limits and you're already incurring charges.

   • Under **Offer usage (monthly)**, take note of your usage of Linux instances, Windows

instances, and EBS storage. The percentage indicates how much of your Free Tier limits you've used this month. If you're at 100%, you will incur charges for further use.

> **Note**
> This information appears only after you've created an instance. However, usage information is not updated in real time; it's updated three times a day.

4. To avoid incurring further charges, delete any resources that are either incurring charges now, or will incur charges if you exceed your Free Tier limit usage.
   - For the instructions to delete your instance, go to the next step in this tutorial.
   - To check if you have resources in other Regions that might be incurring charges, in the **EC2 Free Tier** box, choose **View Global EC2 resources** to open the **EC2 Global View**. For more information, see Amazon EC2 Global View.
5. To view your resource usage for all AWS services under the AWS Free Tier, at the bottom of the **EC2 Free Tier** box, choose **View all AWS Free Tier offers**. For more information, see Using the AWS Free Tier in the *AWS Billing User Guide*.

Step 4: Clean up your instance

After you've finished with the instance that you created for this tutorial, you should clean up by terminating the instance. If you want to do more with this instance before you clean up, see Next steps.

> **Important**
> Terminating an instance effectively deletes it; you can't reconnect to an instance after you've terminated it.

If you launched an instance that is not within the AWS Free Tier, you'll stop incurring charges for that instance as soon as the instance status changes to shutting down or terminated. To keep your instance for later, but not incur charges, you can stop the instance now and then start it again later. For more information, see Stop and start your instance.

To terminate your instance

1. In the navigation pane, choose **Instances**. In the list of instances, select the instance.

2. Choose **Instance state**, **Terminate instance**.
3. Choose **Terminate** when prompted for confirmation.

   Amazon EC2 shuts down and terminates your instance. After your instance is terminated, it remains visible on the console for a short while, and then the entry is automatically deleted. You cannot remove the terminated instance from the console display yourself.

Next steps

After you start your instance, you might want to try some of the following exercises:

• Learn how to remotely manage your EC2 instance using the Run command. For more information, see AWS Systems Manager Run Command in the *AWS Systems Manager User Guide*. • Configure a CloudWatch alarm to notify you if your usage exceeds the Free Tier. For more information, see Tracking your AWS Free Tier usage in the *AWS Billing User Guide*. • Add an EBS volume. For more information, see Create an Amazon EBS volume and Attach an Amazon EBS volume to an instance.

• Install the LAMP stack. For more information, see Install LAMP on Amazon Linux 2. • Learn about instance purchasing options. For more information, see Instance purchasing options. • Get advice about instance types. For more information, see Get instance type recommendations for a new workload.

To ensure the maximum benefit from Amazon EC2, we recommend that you perform the following best practices.

Security

- Manage access to AWS resources and APIs using identity federation with an identity provider and IAM roles whenever possible. For more information, see Creating IAM policies in the *IAM User Guide*.
- Implement the least permissive rules for your security group. For more information, see Security group rules.
- Regularly patch, update, and secure the operating system and applications on your instance. For more information about updating AL2023, see Updating AL2023 in the *AL2023 User Guide*. For more information about updating Amazon Linux 2 or the Amazon Linux AMI, see Manage software on your Linux instance in the *Amazon EC2 User Guide for Linux Instances*.
- Use Amazon Inspector to automatically discover and scan Amazon EC2 instances for software vulnerabilities and unintended network exposure. For more information, see the Amazon Inspector User Guide.
- Use AWS Security Hub controls to monitor your Amazon EC2 resources against security best practices and security standards. For more information about using Security Hub, see Amazon Elastic Compute Cloud controls in the *AWS Security Hub User Guide*.

Storage

- Understand the implications of the root device type for data persistence, backup, and recovery. For more information, see Storage for the root device.
- Use separate Amazon EBS volumes for the operating system versus your data. Ensure that the volume with your data persists after instance termination. For more information, see Preserve data when an instance is terminated.
- Use the instance store available for your instance to store temporary data. Remember that the data stored in instance store is deleted when you stop, hibernate, or terminate your instance. If you use instance store for database storage, ensure that you have a cluster with a replication factor that ensures fault tolerance.
- Encrypt EBS volumes and snapshots. For more information, see Amazon EBS encryption.

- Use instance metadata and custom resource tags to track and identify your AWS resources. For more information, see Instance metadata and user data and Tag your Amazon EC2 resources. • View your current limits for Amazon EC2. Plan to request any limit increases in advance of the

time that you'll need them. For more information, see Amazon EC2 service quotas. • Use AWS Trusted Advisor to inspect your AWS environment, and then make recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. For more information, see AWS Trusted Advisor in the *AWS Support User Guide*.

Backup and recovery

- Regularly back up your EBS volumes using Amazon EBS snapshots, and create an Amazon Machine Image (AMI) from your instance to save the configuration as a template for launching future instances. For more information on AWS services that help achieve this use case, see AWS Backup and Amazon Data Lifecycle Manager.
- Deploy critical components of your application across multiple Availability Zones, and replicate your data appropriately.
- Design your applications to handle dynamic IP addressing when your instance restarts. For more information, see Amazon EC2 instance IP addressing.
- Monitor and respond to events. For more information, see Monitor Amazon EC2. • Ensure that you are prepared to handle failover. For a basic solution, you can manually attach a network interface or Elastic IP address to a replacement instance. For more information, see Elastic network interfaces. For an automated solution, you can use Amazon EC2 Auto Scaling. For more information, see the Amazon EC2 Auto Scaling User Guide.
- Regularly test the process of recovering your instances and Amazon EBS volumes to ensure data and services are restored successfully.

Networking

- Set the time-to-live (TTL) value for your applications to 255, for IPv4 and IPv6. If you use a smaller value, there is a risk that the TTL will expire while application traffic is in transit, causing reachability issues for your instances.

SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

For examples specific to Amazon EC2, see Code examples for Amazon EC2 using AWS SDKs.

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Generate code for your console actions using Console-to Code

> Console-to-Code is in preview release for Amazon EC2 and is subject to change. Available only in the US East (N. Virginia) Region.

The console provides a guided path for creating resources and testing prototypes. If you want to create the same resources at scale, you'll need automation code. Console-to-Code is a feature of the Amazon EC2 console that can help you get started with your automation code. Console-to Code records your console actions, including default values and compatible parameters. It then uses generative AI to suggest code in your preferred infrastructure-as code (IaC) format for the actions you want. You can use the code as a starting point, customizing it to make it production ready for your specific use case.

There is no additional cost for using Console-to-Code.

How it works

Console-to-Code can help you get started with your automation code, as follows:

1. You perform actions in the console, such as launching an instance or enabling detailed monitoring.
2. Console-to-Code records all your actions, including all the default settings and compatible parameters that the console provides.
3. You choose the actions that you want to use in your automation scripts. These can be mutating or read-only (non-mutating) actions, or both types of actions.
4. Console-to-Code generates code in your desired infrastructure-as-code (IaC) format, for example, TypeScript.
5. You copy the code to use in your code development tool or download it to share.
6. You then use the code as a starting point for your automation scripts. You'll need to validate that the code meets your intent and that the parameters will configure your resources as expected. You'll need to customize the code to make it production-ready for your use case. Once you're satisfied with the code, you can use it in your automation scripts.

For the instructions on how to use Console-to-Code in the Amazon EC2 console, see Use Console

to-Code.

Limitations

The following limitations apply when using Console-to-Code.

Supported Regions

Currently only available in the US East (N. Virginia) Region.

Supported code formats

Console-to-Code can currently generate infrastructure-as-code (IaC) in the following code formats:

- CDK Java
- CDK Python
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

Recorded actions table

The following table lists and describes the columns in the **Recorded actions** table in the Console to-Code console.

Column title	Description
Console page	The console page on which the action was performed.
Operation	The API operation.
Type	The type of action. • **Mutating** – API actions that create, modify, or delete resources. • **Read only** – API actions that retrieve data about resources (generally all Describe* actions).

Column title	Description

CLI command	Details about the action that was taken, including the parameters and values.
Creation time	The time the action was taken.

Use Console-to-Code

Use the following instructions to generate code using Console-to-Code in the Amazon EC2 console.

To view an animation of these steps, see **View an animation: Generate code using Console-to Code in the Amazon EC2 console**.

To generate code using Console-to-Code

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Use the console to create resources and test prototypes. For example, use the console to configure and launch instances and enable detailed monitoring.

   Console-to-Code records every action that you perform.
3. In the left navigation pane, choose **Console-to-Code**.
4. In the **Recorded actions** table, review your actions that were recorded, and decide which actions to include for code generation.

   - Use the search field to filter the table by a specific console page or action. As you start to type, the table is filtered.
   - Use the **Type** drop-down to filter by all actions, mutating actions, or read-only actions.

   Note

   Only actions taken during the current session are listed. Actions taken during previous sessions are not retained.
5. Select the check box next to each action for which you require code to be generated. Use

Note

Up to 5 actions can be selected at one time.

6. Choose the **Generate {code} code** button.

   The button label defaults to the last-selected code format. To select a different code format, choose the arrow next to the button.

7. Under **Review code**, choose **Copy** to copy the code to use in your development tool or **Download** to download the file for sharing.

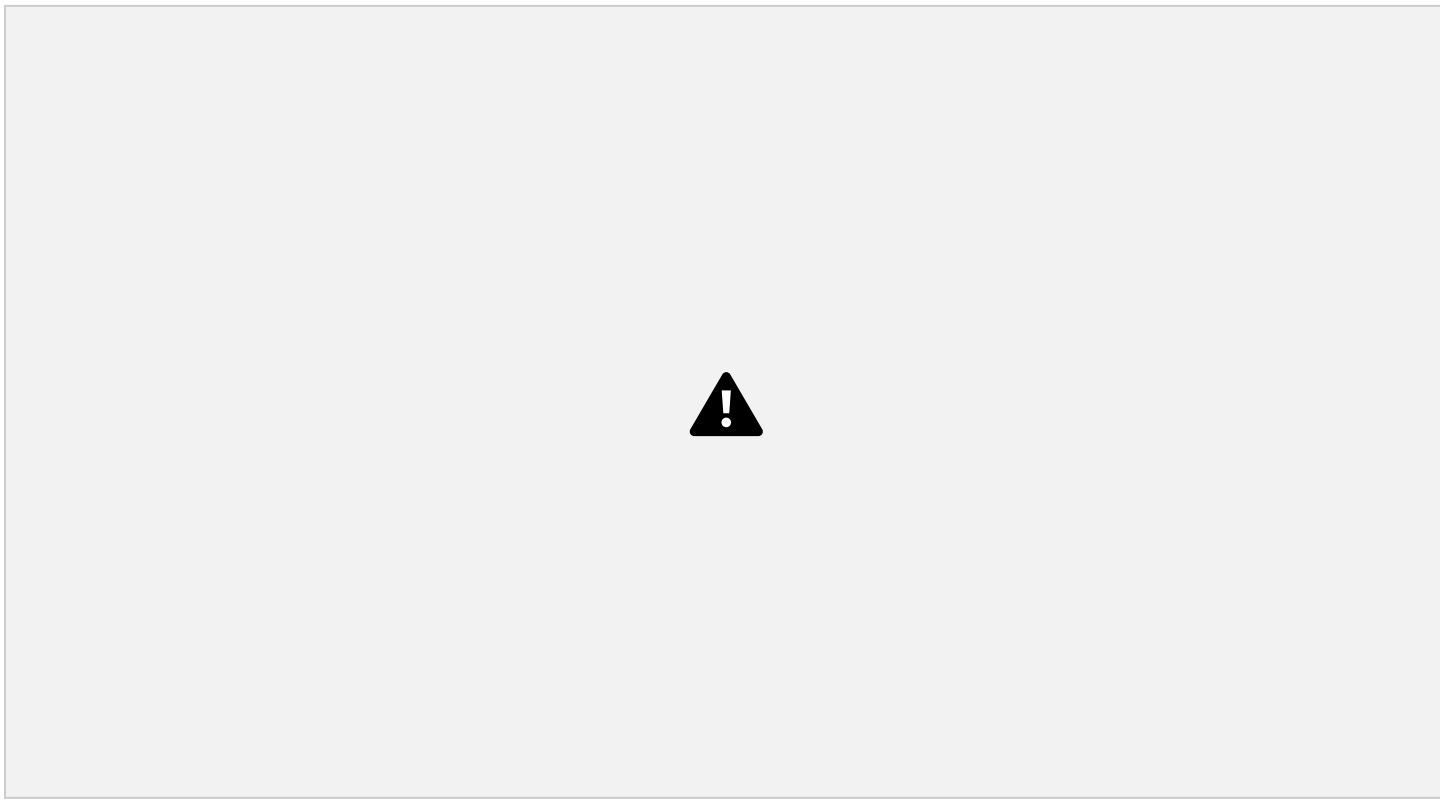
8. Use the code as a starting point for your infrastructure-as-code. You'll need to customize the code to make it production-ready for your specific use case.

Note

If you find that the code is not production ready, please provide us with feedback on how it can be improved (see the following step 9). AWS Support can’t assist you with the generated code or your customized code development.

9. (Optional) Choose the thumbs-up or thumbs-down to let us know if Console-to-Code helped. If you choose the thumbs-down, you can then choose **Provide feedback** to tell us how we can improve the code to better help you.

View an animation: Generate code using Console-to-Code in the Amazon EC2 console



Tutorials for Amazon EC2 instances running Linux

The following tutorials show you how to perform common tasks using EC2 instances running Linux. AWS provides Amazon Linux 2023, Amazon Linux 2, and the Amazon Linux AMI. For more information, see Amazon Linux 2023, Amazon Linux 2, and Amazon Linux AMI. For video tutorials, see AWS Instructional Videos and Labs.

Note

For AL2023 tutorials, see Tutorials in the *Amazon Linux 2023 User Guide*.

Tutorials

- Install LAMP
- Configure SSL/TLS
- Host a WordPress blog
- Tutorial: Increase the size of an Amazon EBS volume on an EC2 instance

Install LAMP

This section includes tutorials that show you how to install an Apache web server with PHP and MariaDB on an Amazon EC2 instance.

> **Note**
>
> For the AL2023 LAMP tutorial, see Tutorial: Install a LAMP server on AL2023 in the *Amazon Linux 2023 User Guide*.

Install LAMP on

- Install LAMP on Amazon Linux 2
- Install LAMP on Amazon Linux

Install LAMP on Amazon Linux 2

The following procedures help you install an Apache web server with PHP and MariaDB (a community-developed fork of MySQL) support on your Amazon Linux 2 instance (sometimes called a LAMP web server or LAMP stack). You can use this server to host a static website or deploy a dynamic PHP application that reads and writes information to a database.

> **Important**
>
> If you are trying to set up a LAMP web server on a different distribution, such as Ubuntu or Red Hat Enterprise Linux, this tutorial will not work. For Amazon Linux AMI, see Install LAMP on Amazon Linux. For Ubuntu, see the following Ubuntu community documentation:

ApacheMySQLPHP. For other distributions, see their specific documentation.

Option: Complete this tutorial using automation

To complete this tutorial using AWS Systems Manager Automation instead of the following tasks, run the AWSDocs-InstallALAMPServer-AL2 Automation document.

Tasks

- Step 1: Prepare the LAMP server
- Step 2: Test your LAMP server
- Step 3: Secure the database server
- Step 4: (Optional) Install phpMyAdmin
- Troubleshoot
- Related topics

Step 1: Prepare the LAMP server

Prerequisites

- This tutorial assumes that you have already launched a new instance using Amazon Linux 2, with a public DNS name that is reachable from the internet. For more information, see Step 1: Launch an instance. You must also have configured your security group to allow SSH (port 22), HTTP (port 80), and HTTPS (port 443) connections. For more information about these prerequisites, see Authorize inbound traffic for your Linux instances.

- The following procedure installs the latest PHP version available on Amazon Linux 2, currently php8.2. If you plan to use PHP applications other than those described in this tutorial, you should check their compatibility with php8.2.

To prepare the LAMP server

1. Connect to your instance.
2. To ensure that all of your software packages are up to date, perform a quick software update on your instance. This process may take a few minutes, but it is important to make sure that you have the latest security updates and bug fixes.

   The -y option installs the updates without asking for confirmation. If you would like to

examine the updates before installing, you can omit this option.

```
[ec2-user ~]$ sudo yum update -y
```

3. Install the mariadb10.5 Amazon Linux Extras repositories to get the latest version of the MariaDB package.

```
[ec2-user ~]$ sudo amazon-linux-extras install mariadb10.5
```

If you receive an error stating sudo: amazon-linux-extras: command not found, then your instance was not launched with an Amazon Linux 2 AMI (perhaps you are using the Amazon Linux AMI instead). You can view your version of Amazon Linux using the following command.

```
cat /etc/system-release
```

4. Install the php8.2 Amazon Linux Extras repositories to get the latest version of the PHP package for Amazon Linux 2.

```
[ec2-user ~]$ sudo amazon-linux-extras install php8.2
```

5. Now that your instance is current, you can install the Apache web server, MariaDB, and PHP software packages. Use the yum install command to install multiple software packages and all related dependencies at the same time

```
[ec2-user ~]$ sudo yum install -y httpd
```

 You can view the current versions of these packages using the following command:

```
yum info package_name
```

6. Start the Apache web server.

```
[ec2-user ~]$ sudo systemctl start httpd
```

7. Use the **systemctl** command to configure the Apache web server to start at each system boot.

```
[ec2-user ~]$ sudo systemctl enable httpd
```

You can verify that **httpd** is on by running the following command:

```
[ec2-user ~]$ sudo systemctl is-enabled httpd
```

8. Add a security rule to allow inbound HTTP (port 80) connections to your instance if you have not already done so. By default, a **launch-wizard-*N*** security group was set up for your instance during initialization. This group contains a single rule to allow SSH connections.

   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. b.

   Choose **Instances** and select your instance.

   c. On the **Security** tab, view the inbound rules. You should see the following rule:

Port range	Protocol	Source
22	tcp	0.0.0.0/0

   Warning

   Using 0.0.0.0/0 allows all IPv4 addresses to access your instance using SSH. This is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, you authorize only a specific IP address or range of addresses to access your instance.

   d. Choose the link for the security group. Using the procedures in Add rules to a security group, add a new inbound security rule with the following values:

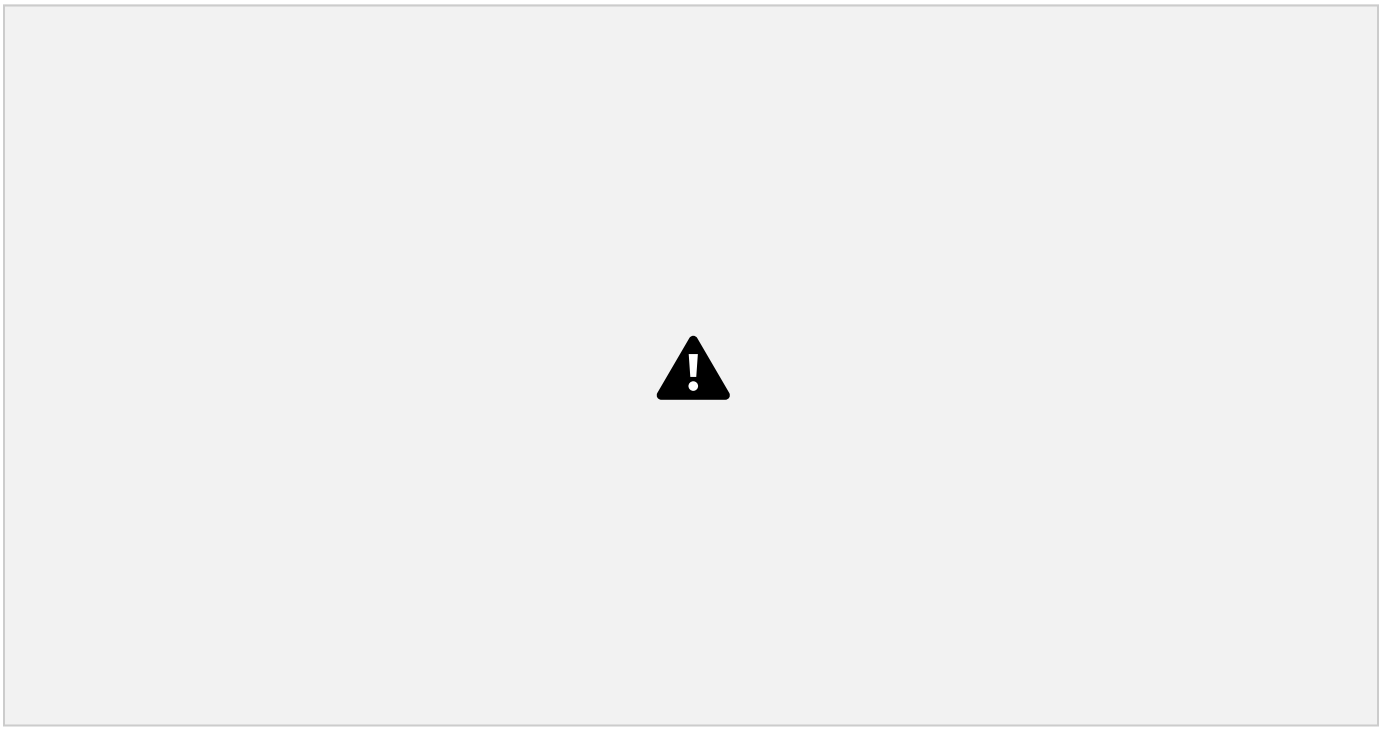
   - **Type**: HTTP
   - **Protocol**: TCP
   - **Port Range**: 80
   - **Source**: Custom

9. Test your web server. In a web browser, type the public DNS address (or the public IP address) of your instance. If there is no content in /var/www/html, you should see the Apache test page. You can get the public DNS for your instance using the Amazon EC2 console (check the **Public DNS** column; if this column is hidden, choose **Show/Hide Columns** (the gear-shaped icon) and choose **Public DNS**).

   Verify that the security group for the instance contains a rule to allow HTTP traffic on port 80. For more information, see Add rules to a security group.

   Important

If you are not using Amazon Linux, you may also need to configure the firewall on your instance to allow these connections. For more information about how to configure the firewall, see the documentation for your specific distribution.



Apache **httpd** serves files that are kept in a directory called the Apache document root. The Amazon Linux Apache document root is /var/www/html, which by default is owned by root.

To allow the ec2-user account to manipulate files in this directory, you must modify the ownership and permissions of the directory. There are many ways to accomplish this task. In this tutorial, you add ec2-user to the apache group, to give the apache group ownership of the / var/www directory and assign write permissions to the group.

To set file permissions

1. Add your user (in this case, ec2-user) to the apache group.

   ```
   [ec2-user ~]$ sudo usermod -a -G apache ec2-user
   ```

2. Log out and then log back in again to pick up the new group, and then verify your membership.

   a. Log out (use the **exit** command or close the terminal window):

   ```
   [ec2-user ~]$ exit
   ```

b. To verify your membership in the apache group, reconnect to your instance, and then run the following command:

```
[ec2-user ~]$ groups
ec2-user adm wheel apache systemd-journal
```

3. Change the group ownership of /var/www and its contents to the apache group.

```
[ec2-user ~]$ sudo chown -R ec2-user:apache /var/www
```

4. To add group write permissions and to set the group ID on future subdirectories, change the directory permissions of /var/www and its subdirectories.

```
[ec2-user ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod   2775 {} \;
```

5. To add group write permissions, recursively change the file permissions of /var/www and its subdirectories:


```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
```

Now, ec2-user (and any future members of the apache group) can add, delete, and edit files in the Apache document root, enabling you to add content, such as a static website or a PHP application.

To secure your web server (Optional)

A web server running the HTTP protocol provides no transport security for the data that it sends or receives. When you connect to an HTTP server using a web browser, the URLs that you visit, the content of webpages that you receive, and the contents (including passwords) of any HTML forms that you submit are all visible to eavesdroppers anywhere along the network pathway. The best practice for securing your web server is to install support for HTTPS (HTTP Secure), which protects your data with SSL/TLS encryption.

For information about enabling HTTPS on your server, see Configure SSL/TLS on Amazon Linux 2.

Step 2: Test your LAMP server

If your server is installed and running, and your file permissions are set correctly, your ec2-user

account should be able to create a PHP file in the /var/www/html directory that is available from the internet.

To test your LAMP server

1. Create a PHP file in the Apache document root.

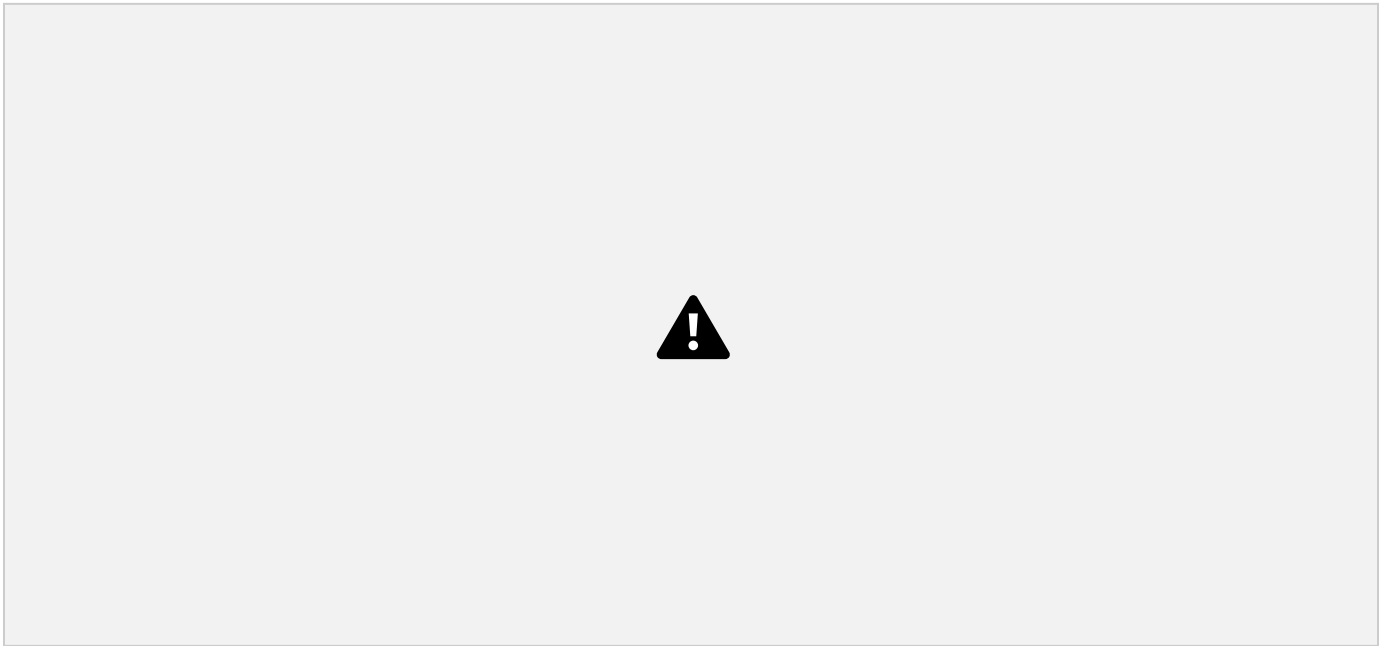
   ```
   [ec2-user ~]$ echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
   ```

   If you get a "Permission denied" error when trying to run this command, try logging out and logging back in again to pick up the proper group permissions that you configured in To set file permissions.

2. In a web browser, type the URL of the file that you just created. This URL is the public DNS address of your instance followed by a forward slash and the file name. For example:

   ```
   http://my.public.dns.amazonaws.com/phpinfo.php
   ```

 You should see the PHP information page:



If you do not see this page, verify that the /var/www/html/phpinfo.php file was created properly in the previous step. You can also verify that all of the required packages were installed with the following command.

```
[ec2-user ~]$ sudo yum list installed httpd mariadb-server php-mysqlnd
```

If any of the required packages are not listed in your output, install them with the **sudo yum**

install *package* command. Also verify that the php7.2 and lamp-mariadb10.2-php7.2 extras are enabled in the output of the **amazon-linux-extras** command.

3. Delete the phpinfo.php file. Although this can be useful information, it should not be broadcast to the internet for security reasons.

```
[ec2-user ~]$ rm /var/www/html/phpinfo.php
```

You should now have a fully functional LAMP web server. If you add content to the Apache document root at /var/www/html, you should be able to view that content at the public DNS address for your instance.

Step 3: Secure the database server

The default installation of the MariaDB server has several features that are great for testing and development, but they should be disabled or removed for production servers. The **mysql_secure_installation** command walks you through the process of setting a root password and removing the insecure features from your installation. Even if you are not planning on using the MariaDB server, we recommend performing this procedure.

To secure the MariaDB server

1. Start the MariaDB server.

```
[ec2-user ~]$ sudo systemctl start mariadb
```

2. Run **mysql_secure_installation**.

```
[ec2-user ~]$ sudo mysql_secure_installation
```

   a. When prompted, type a password for the root account.

      i. Type the current root password. By default, the root account does not have a password set. Press Enter.

      ii. Type **Y** to set a password, and type a secure password twice. For more information about creating a secure password, see https://identitysafe.norton.com/password generator/. Make sure to store this password in a safe place.

Setting a root password for MariaDB is only the most basic measure for securing your database. When you build or install a database-driven application, you typically create a database service user for that application and avoid using the root account for anything but database administration.

b. Type **Y** to remove the anonymous user accounts.

c. Type **Y** to disable the remote root login.

d. Type **Y** to remove the test database.

e. Type **Y** to reload the privilege tables and save your changes.

3. (Optional) If you do not plan to use the MariaDB server right away, stop it. You can restart it when you need it again.


```
[ec2-user ~]$ sudo systemctl stop mariadb
```

4. (Optional) If you want the MariaDB server to start at every boot, type the following command.

```
[ec2-user ~]$ sudo systemctl enable mariadb
```

Step 4: (Optional) Install phpMyAdmin

phpMyAdmin is a web-based database management tool that you can use to view and edit the MySQL databases on your EC2 instance. Follow the steps below to install and configure phpMyAdmin on your Amazon Linux instance.

Important

We do not recommend using phpMyAdmin to access a LAMP server unless you have enabled SSL/TLS in Apache; otherwise, your database administrator password and other data are transmitted insecurely across the internet. For security recommendations from the developers, see Securing your phpMyAdmin installation. For general information about securing a web server on an EC2 instance, see Configure SSL/TLS on Amazon Linux 2.

To install phpMyAdmin

1. Install the required dependencies.

```
[ec2-user ~]$ sudo yum install php-mbstring php-xml -y
```

2. Restart Apache.

```
[ec2-user ~]$ sudo systemctl restart httpd
```

3. Restart php-fpm.

```
[ec2-user ~]$ sudo systemctl restart php-fpm
```

4. Navigate to the Apache document root at /var/www/html.

```
[ec2-user ~]$ cd /var/www/html
```


5. Select a source package for the latest phpMyAdmin release from https://www.phpmyadmin.net/downloads. To download the file directly to your instance, copy the link and paste it into a **wget** command, as in this example:

```
[ec2-user html]$ wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all languages.tar.gz
```

6. Create a phpMyAdmin folder and extract the package into it with the following command.

```
[ec2-user html]$ mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all languages.tar.gz -C phpMyAdmin --strip-components 1
```

7. Delete the *phpMyAdmin-latest-all-languages.tar.gz* tarball.

```
[ec2-user html]$ rm phpMyAdmin-latest-all-languages.tar.gz
```

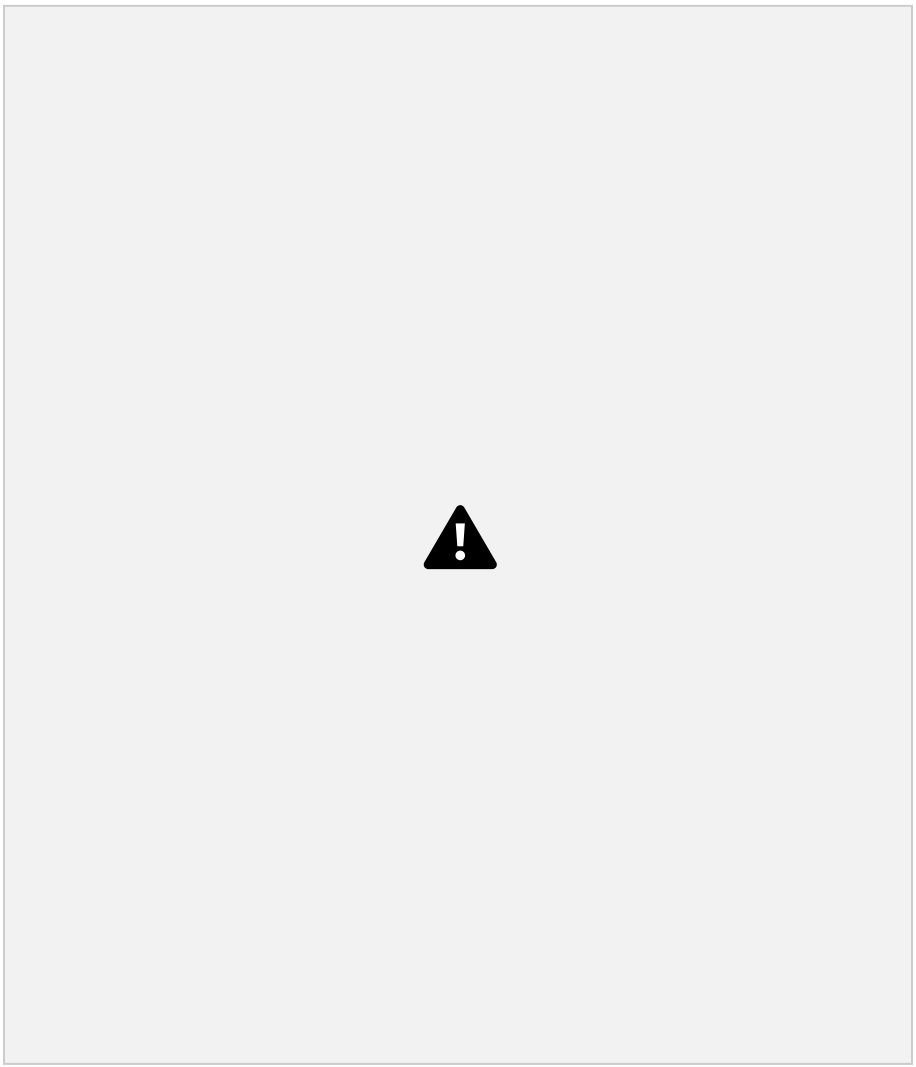
8. (Optional) If the MySQL server is not running, start it now.

```
[ec2-user ~]$ sudo systemctl start mariadb
```

9. In a web browser, type the URL of your phpMyAdmin installation. This URL is the public DNS address (or the public IP address) of your instance followed by a forward slash and the name of your installation directory. For example:

```
http://my.public.dns.amazonaws.com/phpMyAdmin
```

You should see the phpMyAdmin login page:



10. Log in to your phpMyAdmin installation with the root user name and the MySQL root password you created earlier.

    Your installation must still be configured before you put it into service. We suggest that you

begin by manually creating the configuration file, as follows:

a. To start with a minimal configuration file, use your favorite text editor to create a new file, and then copy the contents of config.sample.inc.php into it.

b. Save the file as config.inc.php in the phpMyAdmin directory that contains index.php.

c. Refer to post-file creation instructions in the Using the Setup script section of the phpMyAdmin installation instructions for any additional setup.

For information about using phpMyAdmin, see the phpMyAdmin User Guide.

Troubleshoot

This section offers suggestions for resolving common problems you may encounter while setting up a new LAMP server.

I can't connect to my server using a web browser

Perform the following checks to see if your Apache web server is running and accessible.

- **Is the web server running?**

  You can verify that **httpd** is on by running the following command:

  ```
  [ec2-user ~]$ sudo systemctl is-enabled httpd
  ```

  If the **httpd** process is not running, repeat the steps described in To prepare the LAMP server.

- **Is the firewall correctly configured?**

  Verify that the security group for the instance contains a rule to allow HTTP traffic on port 80. For more information, see Add rules to a security group.

I can't connect to my server using HTTPS

Perform the following checks to see if your Apache web server is configured to support HTTPS. •

Is the web server correctly configured?

After you install Apache, the server is configured for HTTP traffic. To support HTTPS, enable TLS on the server and install an SSL certificate. For information, see Configure SSL/TLS on Amazon Linux 2.

• **Is the firewall correctly configured?**

  Verify that the security group for the instance contains a rule to allow HTTPS traffic on port 443. For more information, see Add rules to a security group.

Related topics

For more information about transferring files to your instance or installing a WordPress blog on your web server, see the following documentation:

• Transfer files to your Linux instance using WinSCP

• Transfer files to Linux instances using an SCP client

• Host a WordPress blog on Amazon Linux 2

For more information about the commands and software used in this tutorial, see the following webpages:

• Apache web server: http://httpd.apache.org/

• MariaDB database server: https://mariadb.org/

• PHP programming language: http://php.net/

• The chmod command: https://en.wikipedia.org/wiki/Chmod

• The chown command: https://en.wikipedia.org/wiki/Chown

For more information about registering a domain name for your web server, or transferring an existing domain name to this host, see Creating and Migrating Domains and Subdomains to Amazon Route 53 in the *Amazon Route 53 Developer Guide*.

Install LAMP on Amazon Linux

The following procedures help you install an Apache web server with PHP and MySQL support on your Amazon Linux instance (sometimes called a LAMP web server or LAMP stack). You can use this server to host a static website or deploy a dynamic PHP application that reads and writes information to a database.

Important

If you are trying to set up a LAMP web server on a different distribution, such as Ubuntu or Red Hat Enterprise Linux, this tutorial will not work. For Amazon Linux 2, see Install LAMP

on Amazon Linux 2. For Ubuntu, see the following Ubuntu community documentation: ApacheMySQLPHP. For other distributions, see their specific documentation.

Option: Complete this tutorial using automation

To complete this tutorial using AWS Systems Manager Automation instead of the following tasks, run the AWSDocs-InstallALAMPServer-AL Automation document.

Step 1: Prepare the LAMP server

Prerequisites

This tutorial assumes that you have already launched a new instance using the Amazon Linux AMI, with a public DNS name that is reachable from the internet. For more information, see Step 1: Launch an instance. You must also have configured your security group to allow SSH (port 22), HTTP (port 80), and HTTPS (port 443) connections. For more information about these prerequisites, see Authorize inbound traffic for your Linux instances.

To install and start the LAMP web server with the Amazon Linux AMI

1. Connect to your instance.
2. To ensure that all of your software packages are up to date, perform a quick software update on your instance. This process may take a few minutes, but it is important to make sure that you have the latest security updates and bug fixes.

   The -y option installs the updates without asking for confirmation. If you would like to examine the updates before installing, you can omit this option.

   ```
   [ec2-user ~]$ sudo yum update -y
   ```

3. Now that your instance is current, you can install the Apache web server, MySQL, and PHP software packages.

   Important

   Some applications may not be compatible with the following recommended software environment. Before installing these packages, check whether your LAMP applications

   are compatible with them. If there is a problem, you may need to install an alternative environment. For more information, see The application software I want to run on my server is incompatible with the installed PHP version or other software

   Use the **yum install** command to install multiple software packages and all related dependencies at the same time.

```
[ec2-user ~]$ sudo yum install -y httpd24 php72 mysql57-server php72-mysqlnd
```

   If you receive the error No package *package-name* available, then your instance was not launched with the Amazon Linux AMI (perhaps you are using Amazon Linux 2 instead). You can view your version of Amazon Linux with the following command.

```
cat /etc/system-release
```

4. Start the Apache web server.

```
[ec2-user ~]$ sudo service httpd start
Starting httpd: [ OK ]
```

5. Use the **chkconfig** command to configure the Apache web server to start at each system boot.

```
[ec2-user ~]$ sudo chkconfig httpd on
```

   The **chkconfig** command does not provide any confirmation message when you successfully use it to enable a service.

   You can verify that **httpd** is on by running the following command:

```
[ec2-user ~]$ chkconfig --list httpd
```

```
httpd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Here, **httpd** is on in runlevels 2, 3, 4, and 5 (which is what you want to see).

6. Add a security rule to allow inbound HTTP (port 80) connections to your instance if you have not already done so. By default, a **launch-wizard-*N*** security group was set up for your instance during initialization. This group contains a single rule to allow SSH connections.

 a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. b. Choose **Instances** and select your instance.

c. On the **Security** tab, view the inbound rules. You should see the following rule:

Port range	Protocol	Source
22	tcp	0.0.0.0/0

Warning

Using 0.0.0.0/0 allows all IPv4 addresses to access your instance using SSH. This is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, you authorize only a specific IP address or range of addresses to access your instance.

d. Choose the link for the security group. Using the procedures in Add rules to a security group, add a new inbound security rule with the following values:

- **Type**: HTTP
- **Protocol**: TCP
- **Port Range**: 80
- **Source**: Custom

7. Test your web server. In a web browser, type the public DNS address (or the public IP address) of your instance. You can get the public DNS address for your instance using the Amazon EC2 console. If there is no content in /var/www/html, you should see the Apache test page. When you add content to the document root, your content appears at the public DNS address of your instance instead of the test page.

   Verify that the security group for the instance contains a rule to allow HTTP traffic on port 80. For more information, see Add rules to a security group.

   If you are not using Amazon Linux, you may also need to configure the firewall on your instance to allow these connections. For more information about how to configure the firewall, see the

documentation for your specific distribution.

Apache **httpd** serves files that are kept in a directory called the Apache document root. The Amazon Linux Apache document root is /var/www/html, which by default is owned by root.


```
[ec2-user ~]$ ls -l /var/www
total 16
drwxr-xr-x 2 root root 4096 Jul 12 01:00 cgi-bin
drwxr-xr-x 3 root root 4096 Aug 7 00:02 error
drwxr-xr-x 2 root root 4096 Jan 6 2012 html
drwxr-xr-x 3 root root 4096 Aug 7 00:02 icons
drwxr-xr-x 2 root root 4096 Aug 7 21:17 noindex
```

To allow the ec2-user account to manipulate files in this directory, you must modify the ownership and permissions of the directory. There are many ways to accomplish this task. In this tutorial, you add ec2-user to the apache group, to give the apache group ownership of the / var/www directory and assign write permissions to the group.

To set file permissions

1. Add your user (in this case, ec2-user) to the apache group.

   ```
   [ec2-user ~]$ sudo usermod -a -G apache ec2-user
   ```

2. Log out and then log back in again to pick up the new group, and then verify your membership.

   a. Log out (use the **exit** command or close the terminal window):

   ```
   [ec2-user ~]$ exit
   ```

   b. To verify your membership in the apache group, reconnect to your instance, and then run the following command:

   ```
   [ec2-user ~]$ groups
   ec2-user wheel apache
   ```

3. Change the group ownership of /var/www and its contents to the apache group.

   ```
   [ec2-user ~]$ sudo chown -R ec2-user:apache /var/www
   ```

4. To add group write permissions and to set the group ID on future subdirectories, change the directory permissions of /var/www and its subdirectories.

```
[ec2-user ~]$ sudo chmod 2775 /var/www
```


```
[ec2-user ~]$ find /var/www -type d -exec sudo chmod 2775 {} \;
```

5. To add group write permissions, recursively change the file permissions of /var/www and its subdirectories:

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
```

Now, ec2-user (and any future members of the apache group) can add, delete, and edit files in the Apache document root, enabling you to add content, such as a static website or a PHP application.

(Optional) Secure your web server

A web server running the HTTP protocol provides no transport security for the data that it sends or receives. When you connect to an HTTP server using a web browser, the URLs that you visit, the content of webpages that you receive, and the contents (including passwords) of any HTML forms that you submit are all visible to eavesdroppers anywhere along the network pathway. The best practice for securing your web server is to install support for HTTPS (HTTP Secure), which protects your data with SSL/TLS encryption.

For information about enabling HTTPS on your server, see Configure SSL/TLS on Amazon Linux.

Step 2: Test your Lamp server

If your server is installed and running, and your file permissions are set correctly, your ec2-user account should be able to create a PHP file in the /var/www/html directory that is available from the internet.

To test your LAMP web server

1. Create a PHP file in the Apache document root.

```
[ec2-user ~]$ echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
```

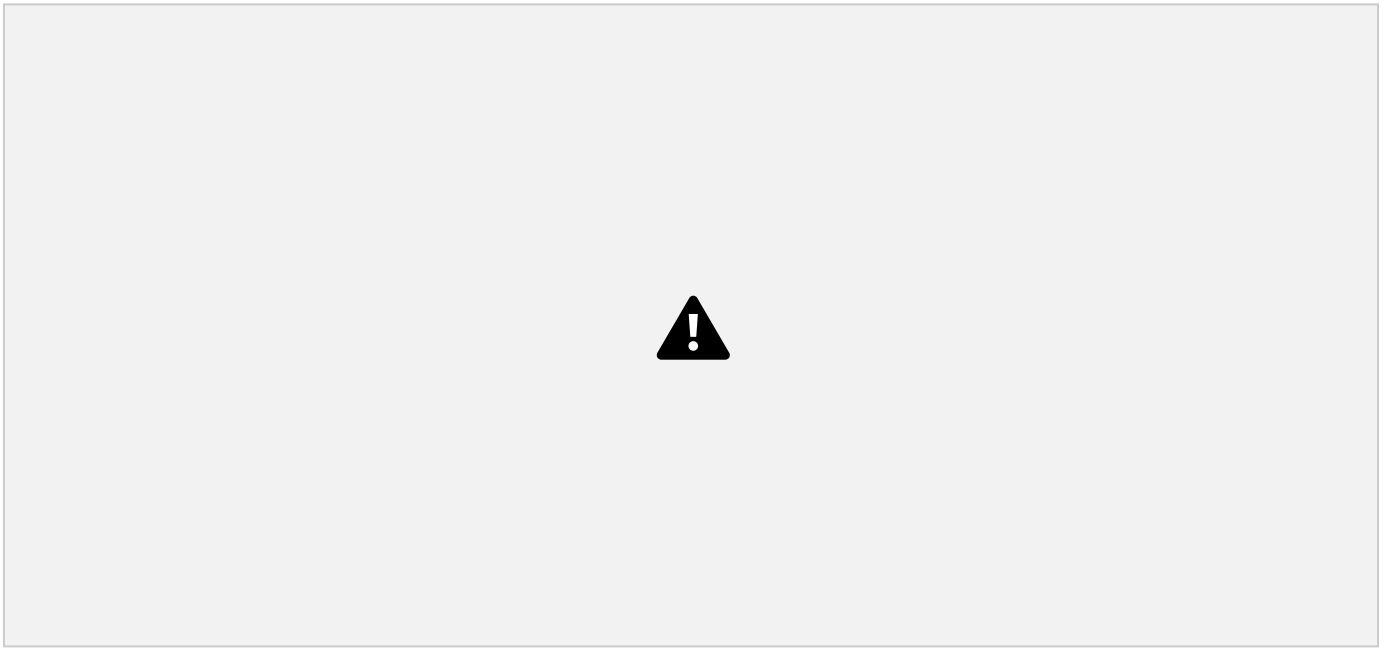
   If you get a "Permission denied" error when trying to run this command, try logging out and logging back in again to pick up the proper group permissions that you configured in Step 1:

Prepare the LAMP server.

2. In a web browser, type the URL of the file that you just created. This URL is the public DNS address of your instance followed by a forward slash and the file name. For example:

 http://*my.public.dns.amazonaws.com*/phpinfo.php

You should see the PHP information page:



If you do not see this page, verify that the /var/www/html/phpinfo.php file was created properly in the previous step. You can also verify that all of the required packages were installed with the following command. The package versions in the second column do not need to match this example output.

```
[ec2-user ~]$ sudo yum list installed httpd24 php72 mysql57-server php72-mysqlnd Loaded plugins:
priorities, update-motd, upgrade-helper
Installed Packages
httpd24.x86_64 2.4.25-1.68.amzn1 @amzn updates
mysql56-server.x86_64 5.6.35-1.23.amzn1 @amzn updates
php70.x86_64 7.0.14-1.20.amzn1 @amzn updates
php70-mysqlnd.x86_64 7.0.14-1.20.amzn1 @amzn updates
```

If any of the required packages are not listed in your output, install them using the **sudo yum install *package*** command.

3. Delete the phpinfo.php file. Although this can be useful information, it should not be broadcast to the internet for security reasons.

```
[ec2-user ~]$ rm /var/www/html/phpinfo.php
```

Step 3: Secure the database server

The default installation of the MySQL server has several features that are great for testing and development, but they should be disabled or removed for production servers. The **mysql_secure_installation** command walks you through the process of setting a root password and removing the insecure features from your installation. Even if you are not planning on using the MySQL server, we recommend performing this procedure.

To secure the database server

1. Start the MySQL server.

```
[ec2-user ~]$ sudo service mysqld start
Initializing MySQL database:
...

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
...

Starting mysqld: [ OK ]
```

2. Run **mysql_secure_installation**.

```
[ec2-user ~]$ sudo mysql_secure_installation
```

   a. When prompted, type a password for the root account.

      i. Type the current root password. By default, the root account does not have a password set. Press Enter.

      ii. Type **Y** to set a password, and type a secure password twice. For more information about creating a secure password, see https://identitysafe.norton.com/password generator/. Make sure to store this password in a safe place.

Setting a root password for MySQL is only the most basic measure for securing your database. When you build or install a database-driven application, you typically create a database service user for that application and avoid using the root account for anything but database administration.

b. Type **Y** to remove the anonymous user accounts.

c. Type **Y** to disable the remote root login.

d. Type **Y** to remove the test database.

e. Type **Y** to reload the privilege tables and save your changes.

3. (Optional) If you do not plan to use the MySQL server right away, stop it. You can restart it when you need it again.

```
[ec2-user ~]$ sudo service mysqld stop
Stopping mysqld: [ OK ]
```

4. (Optional) If you want the MySQL server to start at every boot, type the following command.

```
[ec2-user ~]$ sudo chkconfig mysqld on
```

You should now have a fully functional LAMP web server. If you add content to the Apache document root at /var/www/html, you should be able to view that content at the public DNS address for your instance.

Step 4: (Optional) Install phpMyAdmin

To install phpMyAdmin

phpMyAdmin is a web-based database management tool that you can use to view and edit the MySQL databases on your EC2 instance. Follow the steps below to install and configure phpMyAdmin on your Amazon Linux instance.

Important

We do not recommend using phpMyAdmin to access a LAMP server unless you have enabled SSL/TLS in Apache; otherwise, your database administrator password and other data are transmitted insecurely across the internet. For security recommendations from the developers, see Securing your phpMyAdmin installation.

Note

The Amazon Linux package management system does not currently support the automatic installation of phpMyAdmin in a PHP 7 environment. This tutorial describes how to install phpMyAdmin manually.

1. Log in to your EC2 instance using SSH.
2. Install the required dependencies.

```
[ec2-user ~]$ sudo yum install php72-mbstring.x86_64 -y
```

3. Restart Apache.

```
[ec2-user ~]$ sudo service httpd restart
Stopping httpd: [ OK ]
Starting httpd: [ OK ]
```

4. Navigate to the Apache document root at /var/www/html.

```
[ec2-user ~]$ cd /var/www/html
[ec2-user html]$
```

5. Select a source package for the latest phpMyAdmin release from https://www.phpmyadmin.net/downloads. To download the file directly to your instance, copy the link and paste it into a **wget** command, as in this example:

```
[ec2-user html]$ wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all languages.tar.gz
```

6. Create a phpMyAdmin folder and extract the package into it using the following command.

```
[ec2-user html]$ mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all languages.tar.gz -C phpMyAdmin --strip-components 1
```

7. Delete the *phpMyAdmin-latest-all-languages.tar.gz* tarball.

```
[ec2-user html]$ rm phpMyAdmin-latest-all-languages.tar.gz
```

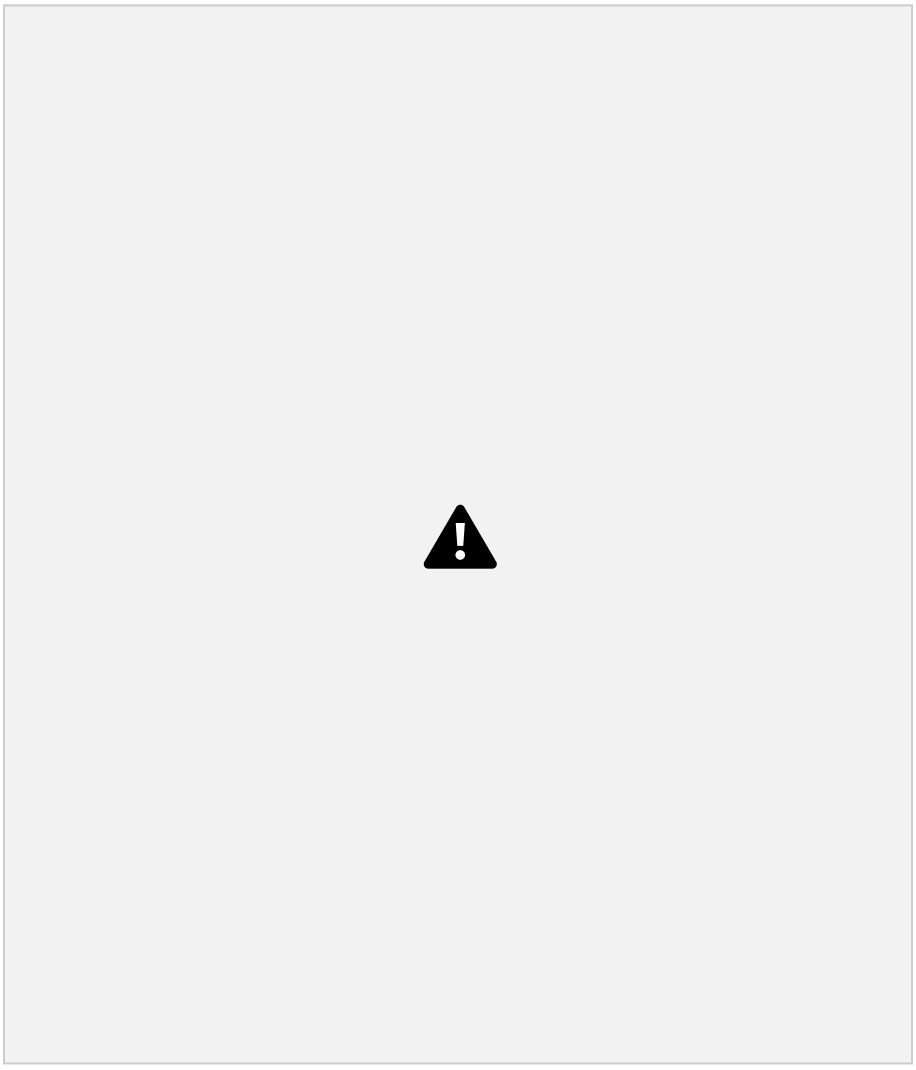
8. (Optional) If the MySQL server is not running, start it now.

```
[ec2-user ~]$ sudo service mysqld start
Starting mysqld: [ OK ]
```

9. In a web browser, type the URL of your phpMyAdmin installation. This URL is the public DNS address (or the public IP address) of your instance followed by a forward slash and the name of your installation directory. For example:

   ```
   http://my.public.dns.amazonaws.com/phpMyAdmin
   ```

   You should see the phpMyAdmin login page:

10. Log in to your phpMyAdmin installation with the root user name and the MySQL root password you created earlier.

Your installation must still be configured before you put it into service. To configure

phpMyAdmin, you can manually create a configuration file, use the setup console, or combine both approaches.

For information about using phpMyAdmin, see the phpMyAdmin User Guide.

Troubleshoot

This section offers suggestions for resolving common problems you may encounter while setting up a new LAMP server.

I can't connect to my server using a web browser.

Perform the following checks to see if your Apache web server is running and accessible.

- **Is the web server running?**

  You can verify that **httpd** is on by running the following command:

  ```
  [ec2-user ~]$ chkconfig --list httpd
  httpd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
  ```

  Here, **httpd** is on in runlevels 2, 3, 4, and 5 (which is what you want to see).

  If the **httpd** process is not running, repeat the steps described in Step 1: Prepare the LAMP server.

- **Is the firewall correctly configured?**

  Verify that the security group for the instance contains a rule to allow HTTP traffic on port 80. For more information, see Add rules to a security group.

The application software I want to run on my server is incompatible with the installed PHP version or other software

This tutorial recommends installing the most up-to-date versions of Apache HTTP Server, PHP, and MySQL. Before installing an additional LAMP application, check its requirements to confirm that it is compatible with your installed environment. If the latest version of PHP is not supported, it is possible (and entirely safe) to downgrade to an earlier supported configuration. You can also

install more than one version of PHP in parallel, which solves certain compatibility problems with a minimum of effort. For information about configuring a preference among multiple installed PHP

versions, see Amazon Linux AMI 2016.09 Release Notes.

How to downgrade

The well-tested previous version of this tutorial called for the following core LAMP packages:

- httpd24
- php56
- mysql55-server
- php56-mysqlnd

If you have already installed the latest packages as recommended at the start of this tutorial, you must first uninstall these packages and other dependencies as follows:

```
[ec2-user ~]$ sudo yum remove -y httpd24 php72 mysql57-server php72-mysqlnd perl-DBD MySQL57
```

Next, install the replacement environment:

```
[ec2-user ~]$ sudo yum install -y httpd24 php56 mysql55-server php56-mysqlnd
```

If you decide later to upgrade to the recommended environment, you must first remove the customized packages and dependencies:

```
[ec2-user ~]$ sudo yum remove -y httpd24 php56 mysql55-server php56-mysqlnd perl-DBD MySQL56
```

Now you can install the latest packages, as described earlier.

Related topics

For more information about transferring files to your instance or installing a WordPress blog on your web server, see the following documentation:

- Transfer files to your Linux instance using WinSCP
- Transfer files to Linux instances using an SCP client