**Installing Docker**

Docker is available on many different operating systems, including most modern Linux distributions, like Ubuntu, and even Mac OSX and Windows. For more information about how to install Docker on your particular operating system, go to the [Docker installation guide](https://docs.docker.com/engine/installation/#installation).

You don't even need a local development system to use Docker. If you are using Amazon EC2 already, you can launch an Amazon Linux instance and install Docker to get started.

**To install Docker on an Amazon Linux instance**

1. Launch an instance with the Amazon Linux AMI. For more information, see [Launching an Instance](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launching-instance.html) in the *Amazon EC2 User Guide for Linux Instances*.
2. Connect to your instance. For more information, see [Connect to Your Linux Instance](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html) in the *Amazon EC2 User Guide for Linux Instances*.
3. Update the installed packages and package cache on your instance.

**Copy**

**sudo yum update -y**

1. Install the most recent Docker Community Edition package.

**Copy**

**sudo yum install -y docker**

1. Start the Docker service.

**Copy**

**sudo service docker start**

1. Add the ec2-user to the docker group so you can execute Docker commands without using sudo.

**Copy**

**sudo usermod -a -G docker ec2-user**

1. Log out and log back in again to pick up the new docker group permissions.
2. Verify that the ec2-user can run Docker commands without sudo.

**Copy**

**docker info**

**Note**

In some cases, you may need to reboot your instance to provide permissions for the ec2-user to access the Docker daemon. Try rebooting your instance if you see the following error:

**Copy**

Cannot connect to the Docker daemon. Is the docker daemon running on this host?

**(Optional) Sign up for a Docker Hub Account**

Docker uses images that are stored in repositories to launch containers with. The most common Docker image repository (and the default repository for the Docker daemon) is Docker Hub. Although you don't need a Docker Hub account to use Amazon ECS or Docker, having a Docker Hub account gives you the freedom to store your modified Docker images so you can use them in your ECS task definitions.

For more information about Docker Hub, and to sign up for an account, go to [https://hub.docker.com](https://hub.docker.com/).

Docker Hub offers public and private registries. You can create a private registry on Docker Hub and configure [Private Registry Authentication](http://docs.aws.amazon.com/AmazonECS/latest/developerguide/private-auth.html) on your ECS container instances to use your private images in task definitions.

**(Optional) Amazon EC2 Container Registry**

Another registry option is Amazon EC2 Container Registry (Amazon ECR). Amazon ECR is a managed AWS Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images. For Amazon ECR product details, featured customer case studies, and FAQs, see the [Amazon EC2 Container Registry product detail pages](http://aws.amazon.com/ecr). To finish this walkthrough using Amazon ECR, see [Create a Docker Image](http://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-basics.html#docker_hub_create_upload) in the *Amazon EC2 Container Registry User Guide*.

**Create a Docker Image and Upload it to Docker Hub**

Amazon ECS task definitions use Docker images to launch containers on the container instances in your clusters. In this section, you create a Docker image of a simple PHP web application, and test it on your local system or EC2 instance, and then push the image to your Docker Hub registry so you can use it in an ECS task definition.

**To create a Docker image of a PHP web application**

1. Install **git** and use it to clone the simple PHP application from your GitHub repository onto your system.
   1. Install git.

**Copy**

**sudo yum install -y git**

* 1. Clone the simple PHP application onto your system.

**Copy**

**git clone https://github.com/awslabs/ecs-demo-php-simple-app**

1. Change directories to the ecs-demo-php-simple-app folder.

**Copy**

**cd ecs-demo-php-simple-app**

1. Examine the Dockerfile in this folder. A Dockerfile is a manifest that describes the base image to use for your Docker image and what you want installed and running on it. For more information about Dockerfiles, go to the [Dockerfile Reference](https://docs.docker.com/engine/reference/builder/).

**Copy**

**cat Dockerfile**

Output:

**Copy**

FROM ubuntu:12.04

# Install dependencies

RUN apt-get update -y

RUN apt-get install -y git curl apache2 php5 libapache2-mod-php5 php5-mcrypt php5-mysql

# Install app

RUN rm -rf /var/www/\*

ADD src /var/www

# Configure apache

RUN a2enmod rewrite

RUN chown -R www-data:www-data /var/www

ENV APACHE\_RUN\_USER www-data

ENV APACHE\_RUN\_GROUP www-data

ENV APACHE\_LOG\_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]

This Dockerfile uses the Ubuntu 12.04 image. The RUN instructions update the package caches, install some software packages for the web server and PHP support, and then add your PHP application to the web server's document root. The EXPOSE instruction exposes port 80 on the container, and the CMD instruction starts the web server.

1. Build the Docker image from your Dockerfile. Substitute *my-dockerhub-username*with your Docker Hub user name.

**Note**

Some versions of Docker may require the full path to your Dockerfile in the following command, instead of the relative path shown below.

**Copy**

**docker build -t *my-dockerhub-username*/amazon-ecs-sample .**

1. Run **docker images** to verify that the image was created correctly and that the image name contains a repository that you can push to (in this example, your Docker Hub user name).

**Copy**

**docker images**

Output:

**Copy**

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

*my-dockerhub-username*/amazon-ecs-sample latest 43c52559a0a1 12 minutes ago 258.1 MB

ubuntu 12.04 78cef618c77e 3 weeks ago 133.7 MB

1. Run the newly built image. The -p 80:80 option maps the exposed port 80 on the container to port 80 on the host system. For more information about **docker run**, go to the [Docker run reference](https://docs.docker.com/engine/reference/run/).

**Copy**

**docker run -p 80:80 *my-dockerhub-username*/amazon-ecs-sample**

**Note**

Output from the Apache web server is displayed in the terminal window. You can ignore the "Could not reliably determine the server's fully qualified domain name" message.

1. Open a browser and point to the server that is running Docker and hosting your container.
   1. If you are using an EC2 instance, this is the **Public DNS** value for the server, which is the same address you use to connect to the instance with SSH. Make sure that the security group for your instance allows inbound traffic on port 80.
   2. If you are running Docker locally, point your browser to <http://localhost/>.
   3. If you are using **docker-machine** on a Windows or Mac computer, find the IP address of the VirtualBox VM that is hosting Docker with the **docker-machine ip**command, substituting *machine-name* with the name of the docker machine you are using.

**Copy**

**docker-machine ip *machine-name***