**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.

**sudo yum update -y**

1. Install the most recent Docker Community Edition package.

**sudo yum install -y docker**

1. Start the Docker service.

**sudo service docker start**

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

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

1. Log out and log back in again to pick up the new docker group permissions. You can accomplish this by closing your current SSH terminal window and reconnecting to your instance in a new one. Your new SSH session will have the appropriate docker group permissions.
2. Verify that the ec2-user can run Docker commands without sudo.

**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:

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

**Create a Docker Image**

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 web application, and test it on your local system or EC2 instance, and then push the image to a container registry (such as Amazon ECR or Docker Hub) so you can use it in an ECS task definition.

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

1. Create a file called Dockerfile. 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/" \t "_blank).

**touch Dockerfile**

1. Edit the Dockerfile you just created and add the following content.
2. FROM ubuntu:12.04
3. # Install dependencies
4. RUN apt-get update -y
5. RUN apt-get install -y apache2
6. # Install apache and write hello world message
7. RUN echo "Hello World!" > /var/www/index.html
8. # Configure apache
9. RUN a2enmod rewrite
10. RUN chown -R www-data:www-data /var/www
11. ENV APACHE\_RUN\_USER www-data
12. ENV APACHE\_RUN\_GROUP www-data
13. ENV APACHE\_LOG\_DIR /var/log/apache2
14. 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 then write the "Hello World!" content 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.

**Note**

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

**docker build -t hello-world .**

1. Run **docker images** to verify that the image was created correctly.

**docker images --filter reference=hello-world**

Output:

REPOSITORY TAG IMAGE ID CREATED SIZE

hello-world latest e9ffedc8c286 4 minutes ago 258MB

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/).

**docker run -p 80:80 hello-world**

**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.
   * 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.
   * If you are running Docker locally, point your browser to <http://localhost/>.
   * 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.

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

1. You should see a web page with your "Hello World!" statement.
2. Stop the Docker container by typing **Ctrl + c**.

**(Optional) Push your image to Amazon Elastic Container Registry**

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 Elastic Container Registry product detail pages](http://aws.amazon.com/ecr).

**Note**

This section requires the AWS CLI. If you do not have the AWS CLI installed on your system, see [Installing the AWS Command Line Interface](http://docs.aws.amazon.com/cli/latest/userguide/installing.html) in the *AWS Command Line Interface User Guide*.

**To tag your image and push it to Amazon ECR**

1. Create an Amazon ECR repository to store your hello-world image. Note the repositoryUri in the output.

**aws ecr create-repository --repository-name hello-world**

Output:

{

"repository": {

"registryId": "*aws\_account\_id*",

"repositoryName": "hello-world",

"repositoryArn": "arn:aws:ecr:us-east-1:*aws\_account\_id*:repository/hello-world",

"createdAt": 1505337806.0,

"repositoryUri": "*aws\_account\_id*.dkr.ecr.us-east-1.amazonaws.com/hello-world"

}

}

1. Tag the hello-world image with the repositoryUri value from the previous step.

**docker tag hello-world *aws\_account\_id*.dkr.ecr.*us-east-1*.amazonaws.com/hello-world**

1. Run the **aws ecr get-login --no-include-email** command to get the **docker login**authentication command string for your registry.

**Note**

The **get-login** command is available in the AWS CLI starting with version 1.9.15; however, we recommend version 1.11.91 or later for recent versions of Docker (17.06 or later). You can check your AWS CLI version with the **aws --version** command. If you are using Docker version 17.06 or later, include the --no-include-email option after get-login. If you receive an Unknown options: --no-include-email error, install the latest version of the AWS CLI. For more information, see [Installing the AWS Command Line Interface](http://docs.aws.amazon.com/cli/latest/userguide/installing.html) in the *AWS Command Line Interface User Guide*.

**aws ecr get-login --no-include-email**

1. Run the **docker login** command that was returned in the previous step. This command provides an authorization token that is valid for 12 hours.

**Important**

When you execute this **docker login** command, the command string can be visible to other users on your system in a process list (**ps -e**) display. Because the **docker login** command contains authentication credentials, there is a risk that other users on your system could view them this way and use them to gain push and pull access to your repositories. If you are not on a secure system, you should consider this risk and log in interactively by omitting the -p *password* option, and then entering the password when prompted.

1. Push the image to Amazon ECR with the repositoryUri value from the earlier step.

**docker push *aws\_account\_id*.dkr.ecr.*us-east-1*.amazonaws.com/hello-world**

**Next Steps**

After the image push is finished, you can use your image in your Amazon ECS task definitions, which you can use to run tasks with.

**Note**

This section requires the AWS CLI. If you do not have the AWS CLI installed on your system, see [Installing the AWS Command Line Interface](http://docs.aws.amazon.com/cli/latest/userguide/installing.html) in the *AWS Command Line Interface User Guide*.

**To register a task definition with the hello-world image**

1. Create a file called hello-world-task-def.json with the following contents, substituting the repositoryUri from the previous section for the image field.
2. {
3. "family": "hello-world",
4. "containerDefinitions": [
5. {
6. "name": "hello-world",
7. "image": "*aws\_account\_id*.dkr.ecr.*us-east-1*.amazonaws.com/hello-world",
8. "cpu": 10,
9. "memory": 500,
10. "portMappings": [
11. {
12. "containerPort": 80,
13. "hostPort": 80
14. }
15. ],
16. "entryPoint": [
17. "/usr/sbin/apache2",
18. "-D",
19. "FOREGROUND"
20. ],
21. "essential": true
22. }
23. ]

}

1. Register a task definition with the hello-world-task-def.json file.

**aws ecs register-task-definition --cli-input-json file://hello-world-task-def.json**

The task definition is registered in the hello-world family as defined in the JSON file.

**To run a task with the hello-world task definition**

**Important**

Before you can run tasks in Amazon ECS, you need to launch container instances into a default cluster. For more information about how to set up and launch container instances, see [Setting Up with Amazon ECS](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/get-set-up-for-amazon-ecs.html) and [Getting Started with Amazon ECS using Fargate](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_GetStarted.html).

* Use the following AWS CLI command to run a task with the hello-world task definition.

**aws ecs run-task --task-definition hello-world**