



Training and  
Certification

# Introduction to AWS OpsWorks

## Self-Paced Lab Guide

Version 2.5

100-SPL22-25-EN-LG-Intro-OpsWorks\_06242013

Copyright © 2014 Amazon Web Services, Inc. and its affiliates.  
All rights reserved.

This work may not be reproduced or redistributed, in whole or in part,  
without prior written permission from Amazon Web Services, Inc.

Commercial copying, lending, or selling is prohibited.

Corrections or feedback on the course, please email us at:  
[aws-course-feedback@amazon.com](mailto:aws-course-feedback@amazon.com)

For all other questions, please email us at:  
<https://aws.amazon.com/contact-us/aws-training/>

## Table of Contents

<b>Lab Overview .....</b>	<b>4</b>
Technical Knowledge Prerequisites .....	4
Download PuTTY.....	4
Topics Covered.....	4
Overview .....	4
<b>Using the Commands and Scripts in this Lab.....</b>	<b>4</b>
IMPORTANT NOTE.....	4
<b>AWS OpsWorks .....</b>	<b>5</b>
<b>Login to the AWS Management Console .....</b>	<b>5</b>
Using qwikLABS to login to the AWS Management Console.....	5
Verify Your Region in the AWS Management Console.....	7
<b>Add Your First Stack.....</b>	<b>8</b>
<b>Add a Layer to your Stack .....</b>	<b>8</b>
<b>Add an instance.....</b>	<b>9</b>
<b>Launching your first instance .....</b>	<b>9</b>
<b>Add your App.....</b>	<b>10</b>
<b>Deploy your App.....</b>	<b>11</b>
<b>Add a MySQL database.....</b>	<b>11</b>
<b>Update the App configuration .....</b>	<b>12</b>
<b>Behind the curtains – the AWS OpsWorks instance life cycle .....</b>	<b>12</b>
<b>Redeploy with database migrations .....</b>	<b>14</b>
<b>Scale your Stack.....</b>	<b>15</b>
<b>Scale based on time .....</b>	<b>17</b>
<b>Scale based on load .....</b>	<b>18</b>
<b>Monitor your instances .....</b>	<b>20</b>
<b>Manage volumes and IPs.....</b>	<b>23</b>
<b>Add and change permissions .....</b>	<b>24</b>
<b>Stopping instances .....</b>	<b>24</b>
<b>End Your Lab .....</b>	<b>25</b>
<b>Conclusion .....</b>	<b>26</b>
<b>What Should I Do Next? .....</b>	<b>26</b>

## Lab Overview

### Technical Knowledge Prerequisites

To successfully complete this lab, you should be familiar with basic Linux server administration and comfortable using the Linux command-line tools.

### Download PuTTY

If you do not already have the PuTTY client installed on your machine, you can download and then launch it from here:

<http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

### Topics Covered

By the end of this lab, you will be able to:

- Create a new AWS OpsWorks stack
- Understand how OpsWorks uses Layers to configure Amazon EC2 instances
- Understand how to use other AWS resources such as Elastic Load Balancing with OpsWorks

### Overview

In this lab you will learn how to use the basic features of AWS OpsWorks, an application management service offered by AWS. You don't need any prior knowledge to complete the lab.

The lab makes use of two other services: Amazon EC2 and Elastic Load Balancing.

Amazon EC2 is a web service that provides compute capacity and Elastic Load Balancing distributes incoming application traffic across multiple Amazon EC2 instances.

This lab follows a fictional scenario to guide you through the lab exercise. In this scenario, you are responsible for deploying and operating the IT infrastructure for a new and upcoming online image sharing and rating service. This application has the working title `opsworks-demo-rails-photo-share-app`. You are also responsible for deploying new versions of that application and scaling the infrastructure during peak times.

All the needed source code is already written for you for this lab.

## Using the Commands and Scripts in this Lab

### IMPORTANT NOTE

If you want to copy and paste ANY code or scripts from this lab guide into an SSH session instead of manually typing them in, please make sure that you first copy and paste it to a text file and THEN copy and paste into the SSH session.

Copying text from Word documents or PDF file frequently introduces line breaks or extra (sometimes hidden) characters when you paste into SSH. We've seen more labs fail because

student pasted directly from PDF into their SSH session and the commands didn't execute properly. Please use your text editor for all code and script copy and paste operations.

## AWS OpsWorks

AWS OpsWorks is an application management service that makes it easy for DevOps users to model and manage the entire application from load balancers to databases. Start from templates for common technologies like Ruby, Node.JS, PHP, and Java, or build your own using Chef recipes to install software packages and perform any task that you can script. AWS OpsWorks can scale your application using automatic load-based or time-based scaling and maintain the health of your application by detecting failed instances and replacing them. You have full control of deployments and automation of each component.

AWS OpsWorks lets you model the different components of your application as layers in a stack, and maps your logical architecture to a physical architecture. You can see all resources associated with your application, and their status, in one place.

AWS OpsWorks provides an event-driven configuration system with rich deployment tools that allow you to efficiently manage your applications over their lifetime, including support for customizable deployments, rollback, partial deployments, patch management, automatic instance scaling, and auto healing.

AWS OpsWorks lets you define template configurations for your entire environment in a format that you can maintain and version just like your application source code. You can reproduce the software configuration on new instances and apply changes to all running instances, ensuring consistent configuration at any time.

AWS OpsWorks supports any software that has a scripted installation. Because OpsWorks uses the Chef framework, you can bring your own recipes or leverage hundreds of community-built configurations.

## Login to the AWS Management Console

### Using qwikLABS to login to the AWS Management Console

Welcome to this self-paced lab! The first step is for you to login to Amazon Web Services.

1. To the right of the lab title, click the **Start Lab** button to launch your qwikLABS. If you are prompted for a token, use the one distributed to you (or the token you purchased).



**Note:** A status bar shows the progress of the lab environment creation process. The AWS Management Console is accessible during lab resource creation, but your AWS resources may not be fully available until the process is complete.



2. On the lab details page, notice the lab properties.
  - a. **Setup Time** - The estimated time to set up the lab environment.
  - b. **Duration** - The time the lab will run before automatically shutting down.

TIME REMAINING	
01:55:50	
Setup Time (min.)	1
Duration (min.)	60
Access (min.)	120

3. In the AWS Management Console section of the qwikLABS page, copy the **Password** to the clipboard.

**AWS Management Console**

User Name:

Password:

4. Click the **Open Console** button:



5. Log into the AWS Management Console using the following steps.
  - a. In the **User Name** field type **awsstudent**.
  - b. In the **Password** field, paste the password copied from the lab details page.

- c. Click **Sign in**.

A screenshot of the AWS Management Console sign-in page. It features three input fields: 'Account:' with the value '420886876503', 'User Name:' with the value 'awsstudent', and 'Password:' with masked characters '.....'. The password field is highlighted with a red rectangular box. Below the password field is a checkbox labeled 'I have an MFA Token (more info)'. At the bottom is a blue 'Sign In' button.

**Note:** The AWS account is automatically generated by qwikLABS. Also, the login credentials for the **awsstudent** account are provisioned by qwikLABS using AWS Identity Access Management.

### Verify Your Region in the AWS Management Console

You are now logged into the Management Console. Before proceeding, we need to verify the AWS region in which we are going to create our server.

AWS OpsWorks provides the ability to place instances in multiple locations. Locations are composed of Availability Zones and Regions. Regions are dispersed and located in separate geographic areas (US, EU, etc.). Availability Zones are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.

By launching instances in separate Regions, you can design your application to be closer to specific customers or to meet legal or other requirements. By launching instances in separate Availability Zones, you can protect your applications from localized regional failures.

6. From the home page of Management Console, select **EC2** from the left hand navigation bar.
7. Look in the upper right corner of the Management Console and make a note of the AWS Region that your lab is configured for. The AWS Region was set for your lab on the qwikLAB launch page.
  - a. As an alternative, you can also click on the ADDL INFO tab on the qwikLAB launch page. This will also display the AWS Region for the lab.

## Add Your First Stack

A Stack orchestrates all Amazon EC2 instances and other AWS resources you need to run your apps. You usually have a Stack per product/app and stage. So you might end up having Stacks called like:

- My Social Network Game A
- Content Management System Prod
- Customer A Staging
- Enterprise Software X
- www.example.com

In this lab you will create a Stack that hosts your image sharing and rating app.

8. Click on the orange box icon at the upper left of the AWS Management Console.
9. Select **AWS OpsWorks** from the green Deployment & Management section to access the OpsWorks console.
10. Click the **Add Your First Stack** button and you will see a form appear.
11. Enter a **Name** (i.e. OpsWorks Lab).
12. Select the **AWS Region**. Make sure this is the same region as where the lab is running from Step 7 or 7a.

The other settings deal with the location of your Amazon EC2 instances, which operating system is used, instance specifics, security settings, and configuration management. All those settings can be left as they are.

13. Press **Add Stack** to create the Stack.

This might take a few moments, as there are some resources that are being created and provisioned for you as part of the Stack creation. As soon as the process is completed you will be redirected to the Stack dashboard.

## Add a Layer to your Stack

As mentioned earlier, a Stack is your collection of Amazon EC2 instances and other resources that serve a specific purpose. To orchestrate the Amazon EC2 instances you need to create Layers in a Stack. A Layer describes the configuration and behavior of a group of servers.

If you were asked to describe your IT architecture to someone, you might draw several boxes on a white board to show how the load balancers, app servers, databases, and other systems you own communicate and work together. Each of these groups corresponds to a Layer in OpsWorks. Your image-sharing application is being implemented using the web framework Ruby on Rails, but it could have been implemented using other languages or frameworks, as you can bring all your favorite languages, frameworks, and tools to AWS OpsWorks. Most of them just work with the default settings that we provide or with what the OpsWorks community has publically shared.



14. Click **Add a Layer** to get redirected to a form.

The Rails App Server Layer type is already preselected. OpsWorks offers different prebuilt Layer types for you to choose from. If you don't find what you are looking for, you can extend existing Layers or even create your own Layers. There is a large OpsWorks community which continuously shares new configuration scripts online. Today you can easily find thousands of "cookbooks" on code sharing portals like GitHub.

15. This Lab works without any configuration changes so leave all the Rails-specific settings as they are and press **Add Layer**.

The Layer will be created and you will be redirected to the Layers overview to see the newly created Rails App Server Layer.

In this scenario, you will start with a static version of the application to pitch the idea to your co-founders/investors/colleagues - you just need that single Rails App Server Layer for now.

## Add an instance

After you have created a Stack with an app server Layer you need to add an instance in it.

16. Start defining your first instance and click **Add an instance** on the Rails App Server Layer.

You will see that no instances are currently created.

17. Click **Add an instance**.

A hostname such as **rails-app1** is suggested for the new instance. We will leave all default settings as they appear here, but you have the ability to change instance specific configurations to override Stack defaults.

18. Continue by pressing **Add Instance**.

After adding the instance you see one instance defined in your Stack that belongs to the Rails App Server Layer.

## Launching your first instance

The instance is defined in OpsWorks but not started on Amazon EC2 yet.

19. Start the instance by clicking the **start** button.

OpsWorks will now request an Amazon EC2 instance with the configuration you specified. If you stay on this page, you will see the instances status being updated while it gets configured and launched.

The instance will go through the following states: requested, pending, booting, running setup, and online. The public IP address of the instance will be displayed as soon as it is available. Launching an instance usually takes around 5 minutes. For this lab, you can go ahead and continue to the next step while the instance is launching, as the next step does not require a running instance.

## Add your App

So far, you have created a Stack with a Rails app server Layer and an Amazon EC2 instance. This completes the setup of your 'hardware' for now. The next step is defining the actual application that you want to run on that instance.

Apps are usually some source code you or others have created. It can be anything from a popular open source content management system to your own code. For this lab, we've provided the Rails application for you.

To deploy the application to an instance, we need to let OpsWorks know where the source code is located.

20. Start by clicking on **Apps** in the left side navigation.

As you can see there are no apps defined yet. OpsWorks can handle multiple apps per Stack.

This is especially useful if you split your software in small services, like your web frontend, API, or workers.

21. Open the app creation form by clicking **Add an app**.

22. Start by choosing a name – in your case you would probably just enter a name like **photo app**. You don't need to change the other base settings, as the defaults are fine.

The source code of the app is publicly available on [github.com](https://github.com/aws-labs/opsworks-demo-rails-photo-share-app), a popular code management solution that is used by many open source projects.

23. Enter the repository URL:

**`https://github.com/aws-labs/opsworks-demo-rails-photo-share-app.git`**

Note: If you copy and paste the URL, copy and paste first to a .txt file using notepad or some other text application, then copy and paste into the console. This will ensure that there are no hidden characters or format issues with the URL.

24. Enter the **Branch/Revision** optional field: **version1**

25. For this lab, we can skip the domain handling and SSL settings, so finish by clicking **Add App**.

OpsWorks now is aware where to get the source code as well of the other app-related settings.

You should be back on the App overview page and now see the app you just created.

## Deploy your App

Before you deploy the app, please confirm that the instance is up and running.

26. Click on **Instances** in the navigation pane to see if the app server is already online. If the instance is not online yet, please wait until it is.
27. As soon as the instance is online change back to **Apps** in the navigation pane.
28. Create a new deployment by clicking on **Deploy** next to your app.
29. In the deployment form, leave the default setting in the **Command** field.
30. Optionally, enter a comment to annotate your deployment (i.e. First deploy of static app version). The comment field can also be used to indicate which bugs you have fixed, which features changed, or to provide a link to your issue/bug tracker.
31. Leave all of the other settings as their default.
32. Click **Deploy** to start the deployment. You will see the deployment progress.

As soon as the deployment is done the page is updated with all deployment details and logs from the instances.

33. Click on your app server's **Hostname** to go the instance detail page. Here you will find instance details such as the public IP address.
34. Click on the IP to open a new browser tab and confirm the success of the deployment. If everything worked as designed, you will see your initial static version deployed.

In this scenario, you presented your app to the stakeholders and it was hit. Congratulations. It is time to implement the real thing, an app version that is able to upload and store images.

35. Return to OpsWorks to continue.

## Add a MySQL database

Since this photo app is designed to store the user's images in a MySQL database, the next step is to add the MySQL DB Layer to your Stack.

36. Go back to the **Layers** overview in the navigation pane.
37. Add a Layer by clicking the **plus icon** below the Rails App Server Layer.

38. Change the Layer type from **HAProxy** to **MySQL**.

39. Press **Add Layer**.

You are redirected back to the Layer overview and now see two Layers, the old Rails App Server and the new MySQL Server Layer.

40. Click on **Add an instance** in the MySQL Layer to get redirected to the instances overview.

41. **Add an instance** to the MySQL layer.

42. You can again leave all the default settings as they are.

You will see two instances now, one in the Rails Layer (already serving the static app) and the new instance in the MySQL Layer. **Start** the MySQL DB instance now.

## Update the App configuration

While the database instance starts, change the app configuration to use version 2 of the app. Version 2 has the image upload functionality implemented and the ability to store the images in the database.

43. Click on **Apps**.

44. Click **Edit** next to the application name.

45. In the form, change Data source type field from None to **OpsWorks**. You should then see the database instance created in the previous step selected.

46. In the form, change the branch/revision field from empty to **version2**.

47. Click **Save** on the bottom of the form.

This change will result in checking out the version2 branch instead of the master branch from github for the next deployment. In production environments, you should always lock your app to a specific branch/revision/tag so that you do not accidentally push code you don't want to be pushed yet.

## Behind the curtains – the AWS OpsWorks instance life cycle

One thing that might be worth explaining is how the Rails app server and the MySQL instance get created, configured, and deployed. Here is a brief introduction to the AWS OpsWorks instance life cycle.

48. Go back to the **Instances** overview..

49. Click one the rails app instance's **host name** to go to its detail page.

Beneath all Amazon EC2 configuration and status details you will see a list of the last OpsWorks event logs. At this point, three different command types should be shown: setup, configure, deploy. If the DB server is online you will see a fourth command type.

So what is going on?

A setup event is triggered on every instance launch. It starts a process that installs and configures the instance. What exactly happens during the instance creation is defined through the Layer configuration. This means each instance starts from the same, curated base image and is turned into what you want on the fly. You can see the recipes used to configure your instances by selecting the Recipes link under the layer. Each of the recipes is listed and you can select it to see the actual recipe.

After a successful instance setup, OpsWorks triggers a configure event on every running instance in the Stack, including the one that just ran the setup. As the Rails app server was the only running instance in the Stack at the time, the configure event was just triggered on the instance itself. During a configure event an instance checks if its configuration is correct and if not, adapts to the changes in the Stack. In this specific case, the app server instance did not need any configuration changes.

The setup/configure mechanism enables you to programmatically change configurations, create or delete files, execute any code/shell script, create users – in short everything you can do via a command line on a server. Leveraging that mechanism, your instances will be always up to date and configured to handle your workload.

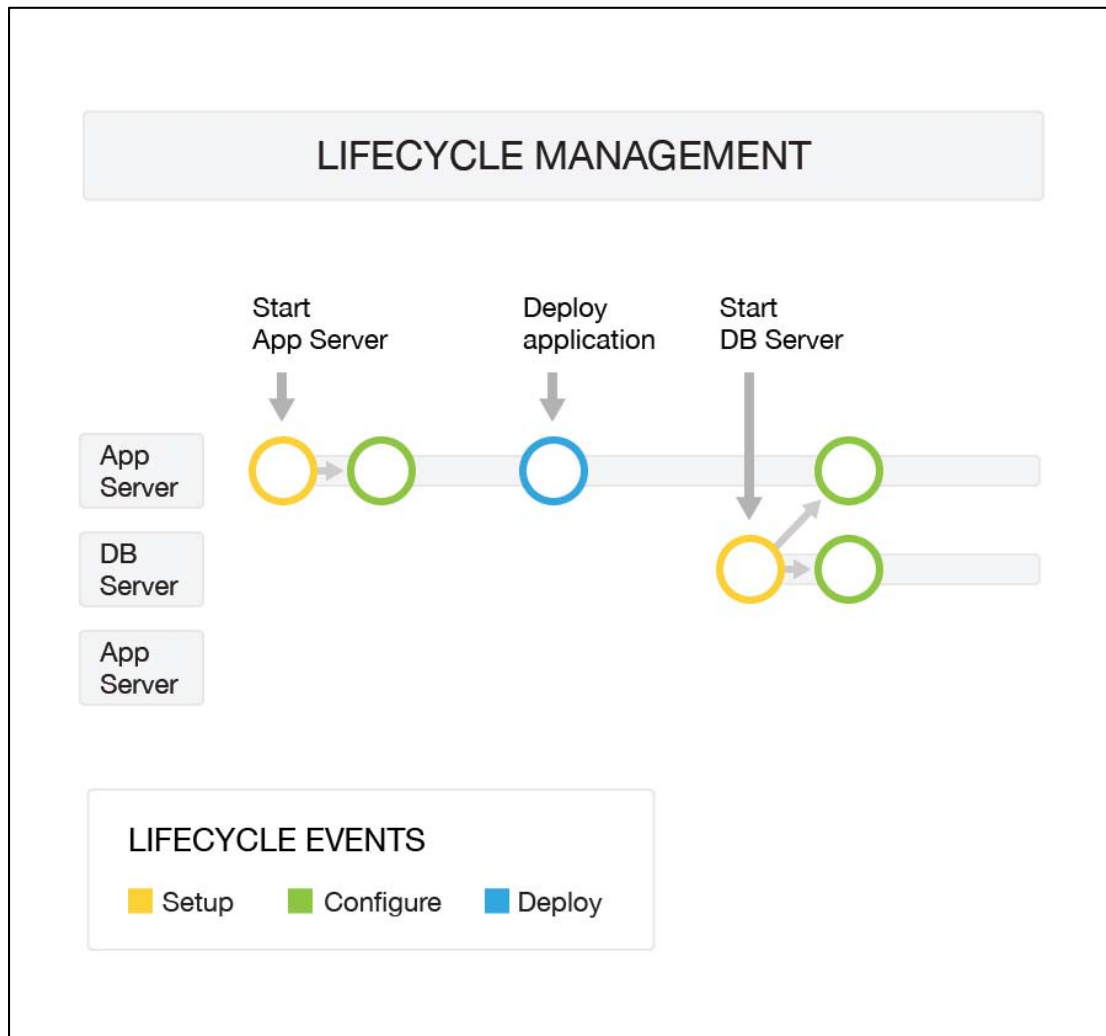
Here are a few common use cases where this might be ideal:

- An app server is started/stopped and your load balancer updates its configuration to include/exclude it.
- You use an open source NoSQL database and add a new instance to an existing set of servers. The new shard has to be included in the configuration and data might need to be rebalanced.
- Your monitoring software needs to be aware of the current instances and its configuration to collect the right metrics and create or change alarms.

After the configure event you see the deploy event you triggered earlier. This again triggered a process on the instance, which got the source code of github, configured the Rails environment, and configured the web server to serve the app.

If the DB instance is online, you should see a fourth entry in the list - a configure event. This was triggered by the new MySQL DB instance. After the successful DB server setup OpsWorks triggered the configure event all instances that are online. In contrast to the earlier configure event, this one changed the Rails instance. It made the Rails app aware of an existing MySQL DB and set its connection details in the apps configuration.

Here a graphical overview of what happened step by step.



Now we will work with the new database configuration to deploy version2 of the app!

## Redeploy with database migrations

50. In the **Instances** view, wait for the MySQL instance to finish booting.
51. Click on **Apps** in the navigation pane and the **deploy** action next to your app. You will see the deployment configuration form again.
52. This time **check the migration toggle** to enable a database migration. This is built-in functionality offered by Rails that lets you programmatically change your DB.
53. If you like, you can again type something in a comment field to indicate the changes you want to deploy.

54. Keep the rest as it is and hit **Deploy** to start the deployment.

On the deployment detail view you will see both instances in the deployment overview working.

The deploy event on the app server will check out the new code, update the gems (libraries) the Ruby on Rails app uses, trigger the database migration, and in the end change the symlink to the new version.

On the MySQL instance, a different process occurs, as explained in the section above (Behind the curtains – the AWS OpsWorks instance life cycle).

55. As soon as the deployment completes successfully, change back to your browser tab with the app and refresh it to see the new version. If you closed the tab already, go to the app server by clicking the hostname and click the IP address to reopen the app.

You will see the new version of the image-sharing app, which lets you upload images. During the DB migration, we even added a first image in the DB for you. Upload your own images to see the app in action, if you like.



Congratulations, you have created and configured the infrastructure needed to host a data driven web app and even deployed two different versions of it.

## Scale your Stack

In this scenario, the good news is that the actual working version of your app is a huge success and people love it. The bad news is that your app is so popular, your one app server can't stand the load anymore.

You can scale your app by adding a load balancer. With OpsWorks you have the choice between self managed HAProxy – you would add instances inside a HAProxy layer – and Elastic Load Balancing (ELB). In this lab we will use the Elastic Loading Balancing service.

56. To use ELB, go to your **Layers** page in the navigation pane. You will see the two layers you have defined earlier.
57. Click on **Network** in the Rails App Server Layer. You will be redirected to the Rails App Server Layer network configuration page. Click on **Add an ELB**.
58. Scroll down to Elastic Load Balancing and select the given ELB already created for you in **Available ELBs**. Scroll down and click **Save**.
59. After saving, you should see the Layers settings page and a link to the new ELB. Click it.

Note: The ELB may take a few moments to pick up the app server.

60. After a few moments, you should see that the ELB is serving traffic to the app server. You can confirm that by clicking the DNS name. A new tab will open and you should see the app again as the ELB forwards you to your one app server.

You are now set to add more Rails instances that serve your app.

61. Click on **Instances** in the navigation pane.
62. Add a new app server. Click **+ Instance**.
63. This time, change the Availability Zone (AZ)/subnet to one different than the default. This will distribute the load between two AZs. Click **Add Instance**.

A short reminder: Amazon EC2 is hosted in multiple locations world-wide. These locations are composed of regions and Availability Zones. Each region is a separate geographic area. Each region has multiple, isolated locations known as Availability Zones.

Each Availability Zone is isolated, but the Availability Zones in a region are connected through low-latency links. If you distribute your instances across multiple Availability Zones and one instance fails, you can design your application so that an instance in another Availability Zone can handle requests.

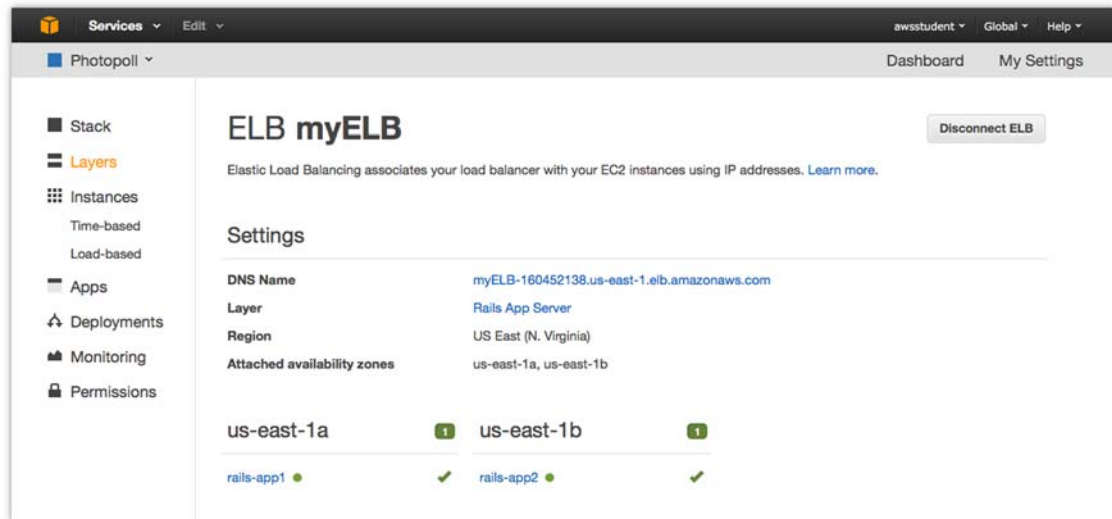
64. Start the new instance by clicking the **start** button.

You now have two app servers. OpsWorks will now reconfigure the ELB for you.

65. Go back to the ELB by clicking the ELB name on the top right of the Rails App Server Layer. Here you should see the new instance in the second AZ/subnet.

After the second app server comes online and the health check is successful you will see that the ELB serves traffic to this instance as well. As this might take some time to fully spin up, just continue with the lab and check later if you like. You should see the following after some time.





## Scale based on time

You now have a fault tolerant setup for your app servers using an ELB to distribute the load equally to two instances in different AZs. But let's suppose that your app usage has grown further and established a regular load pattern.

The next goal is to add additional compute power in your known peak times (in this case, Mon – Fri during lunch hours in the US).

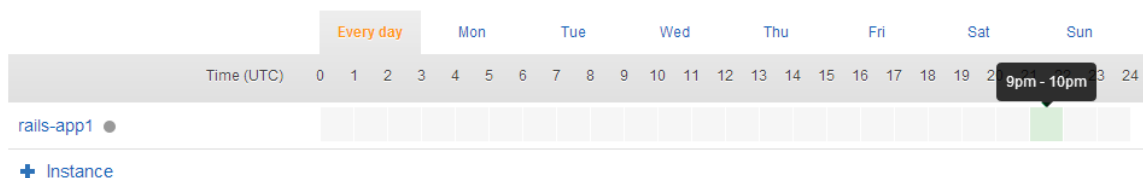
66. To do this, click on **Instances/Time-based** in the navigation. This will take you to the configuration settings of instances that are supposed to be only online during specific times of the day.

67. Add a time-based instance in the Rails layer. You can again keep all the defaults.

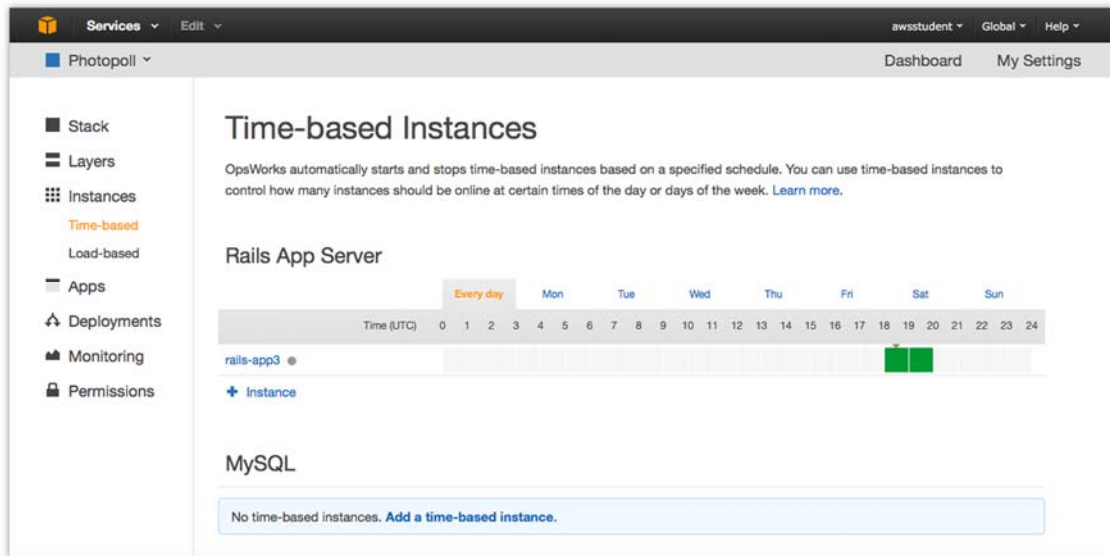
After you have added the instance you see a scheduler – with a green triangle indicating the current time in UTC (Coordinated Universal Time).

68. Click on the grey square below the green triangle to start the instance during the current hour.

### Rails App Server



69. Also select the adjacent slot on the right to ensure it is not turned off if you are doing this step in the lab close to the end of the hour. The result should look something like this.

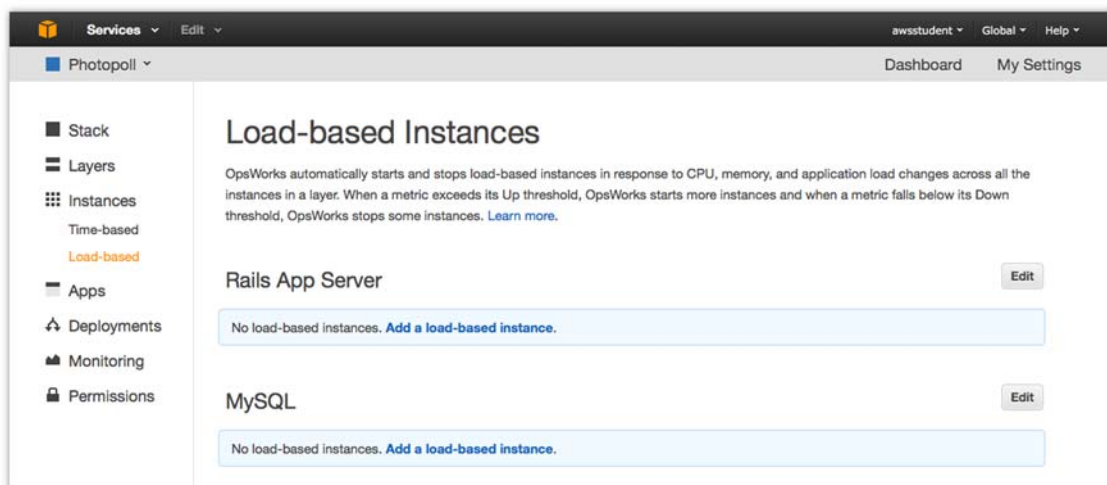


With the scheduler you can add instances to Layers and define the times in which each instance is supposed to be online or offline. By clicking on specific days you can adjust patterns that are based, for example, on high or low usage on weekends.

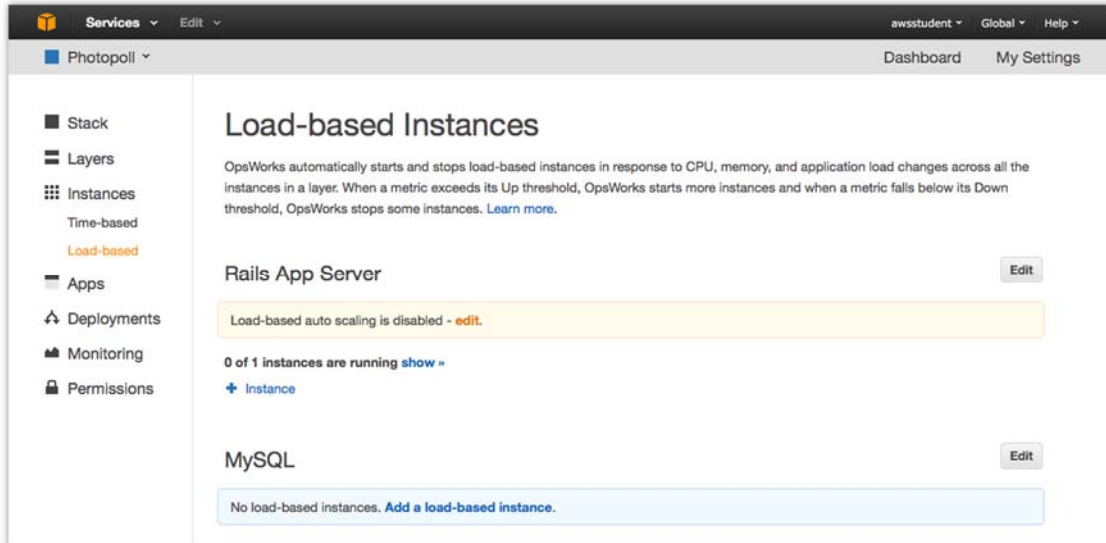
## Scale based on load

Your load cycle will experience unpredictable spikes, so you cannot just add instances that are started or stopped based on the current time – you will need to set parameters based on the average metrics of a Layer. Consider the load implications that might occur as the result of a social media campaign or a TV commercial highlighting your app.

70. To configure this, click on **Instances/Load-based** instances in the navigation.

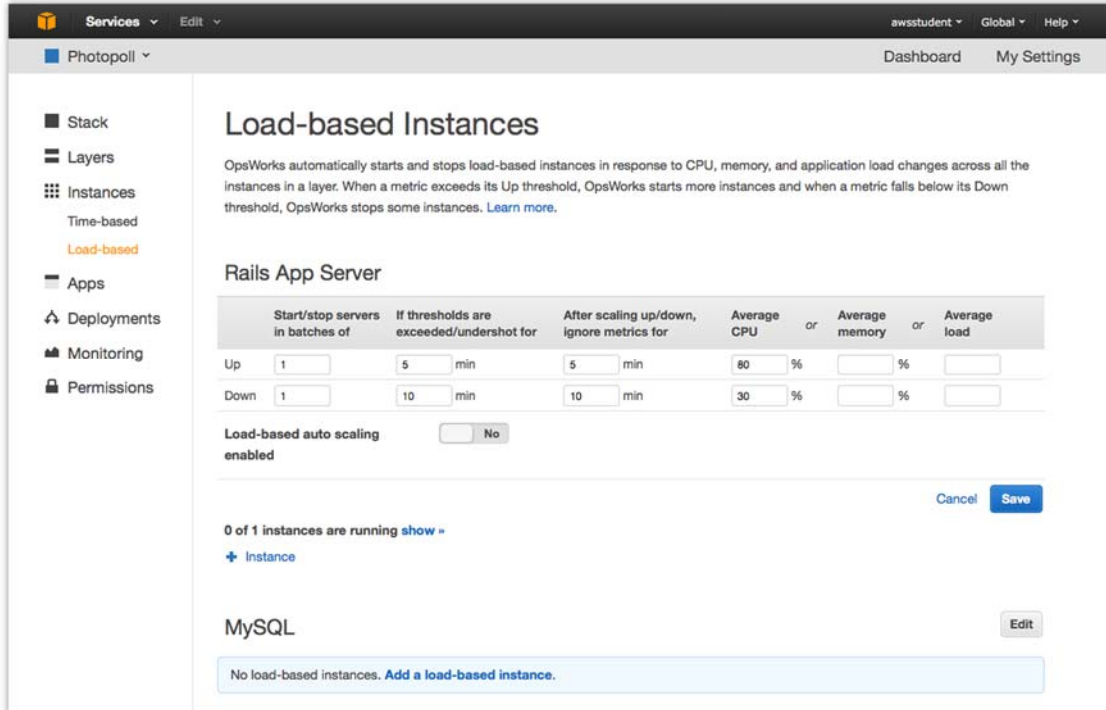


71. Add an instance in the Rails App Server Layer with all defaults selected. You should see this afterwards.



The screenshot shows the AWS OpsWorks console interface. On the left is a navigation menu with options: Stack, Layers, Instances (Time-based, Load-based), Apps, Deployments, Monitoring, and Permissions. The main content area is titled 'Load-based Instances' and includes a description of how OpsWorks manages instances based on CPU, memory, and application load. Below this, there are sections for 'Rails App Server' and 'MySQL'. The 'Rails App Server' section shows a yellow banner stating 'Load-based auto scaling is disabled - edit.' and indicates '0 of 1 instances are running'. The 'MySQL' section shows a blue banner stating 'No load-based instances.' and includes a link to 'Add a load-based instance.'

72. Click on **edit** to configure the thresholds on which the load-based instances should start or stop.



This screenshot shows the 'Load-based Instances' configuration page for the 'Rails App Server' layer. It includes a table for setting scaling thresholds. The 'Load-based auto scaling' toggle is currently set to 'No'.

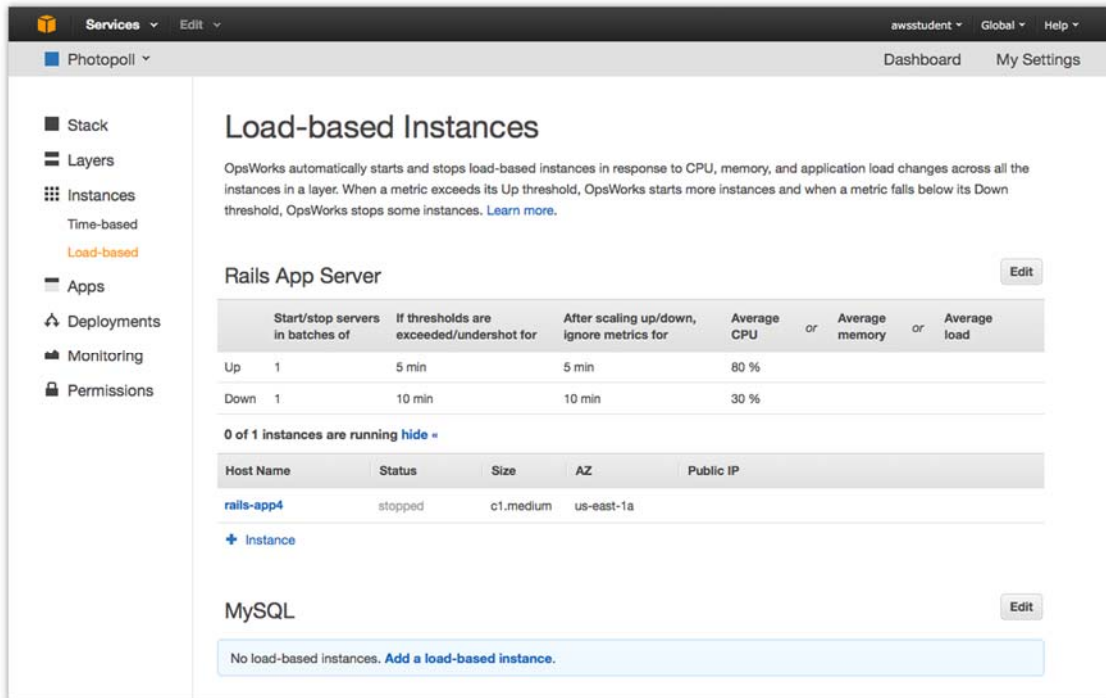
	Start/stop servers in batches of	If thresholds are exceeded/undershot for	After scaling up/down, ignore metrics for	Average CPU	or	Average memory	or	Average load
Up	1	5 min	5 min	80 %				
Down	1	10 min	10 min	30 %				

Below the table, the 'Load-based auto scaling' toggle is set to 'No'. At the bottom right of the configuration section are 'Cancel' and 'Save' buttons. The instance count remains at '0 of 1 instances are running'.

You can define how many instances to start or stop if certain thresholds are exceeded or undershot for a certain time. You can set the grace period after such an event and the thresholds of the provided metrics.

73. Keep all settings, switch the Load-based auto scaling to on and save the change.

74. You now see the configuration and can extend the instances view by clicking **show** to list the instances.



The screenshot shows the AWS OpsWorks console interface. On the left is a navigation sidebar with options: Stack, Layers, Instances (with sub-options Time-based and Load-based), Apps, Deployments, Monitoring, and Permissions. The main content area is titled 'Load-based Instances' and includes a description of how OpsWorks automatically starts and stops instances based on CPU, memory, and application load. Below this is a table for 'Rails App Server' configuration:

	Start/stop servers in batches of	If thresholds are exceeded/undershot for	After scaling up/down, ignore metrics for	Average CPU	or	Average memory	or	Average load
Up	1	5 min	5 min	80 %				
Down	1	10 min	10 min	30 %				

Below the table, it states '0 of 1 instances are running' with a 'hide' link. A table lists the instance 'rails-app4' with status 'stopped', size 'c1.medium', and availability zone 'us-east-1a'. There is a '+ Instance' link to add more. At the bottom, the 'MySQL' section shows 'No load-based instances' with a link to 'Add a load-based instance'.

The current average CPU consumption of your app servers is very likely below 80% and therefore the instance will not start. But should that ever change due to a high load on your app servers OpsWorks would start an additional instance for you. The upper bound of instances started is the number of instances defined in this view. So currently, it would never exceed the one additional instance.

## Monitor your instances

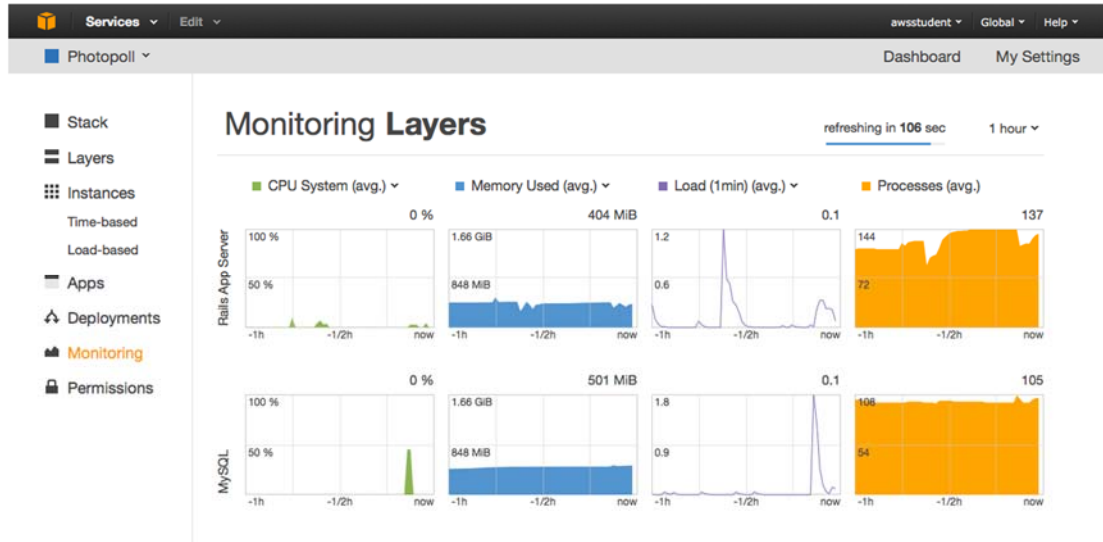
As you might want to fine-tune the thresholds for the load-based instances you have to be able to monitor your actual metrics.

75. Let's do that by clicking on **Monitoring** in the navigation. You should see averages over the two layers.

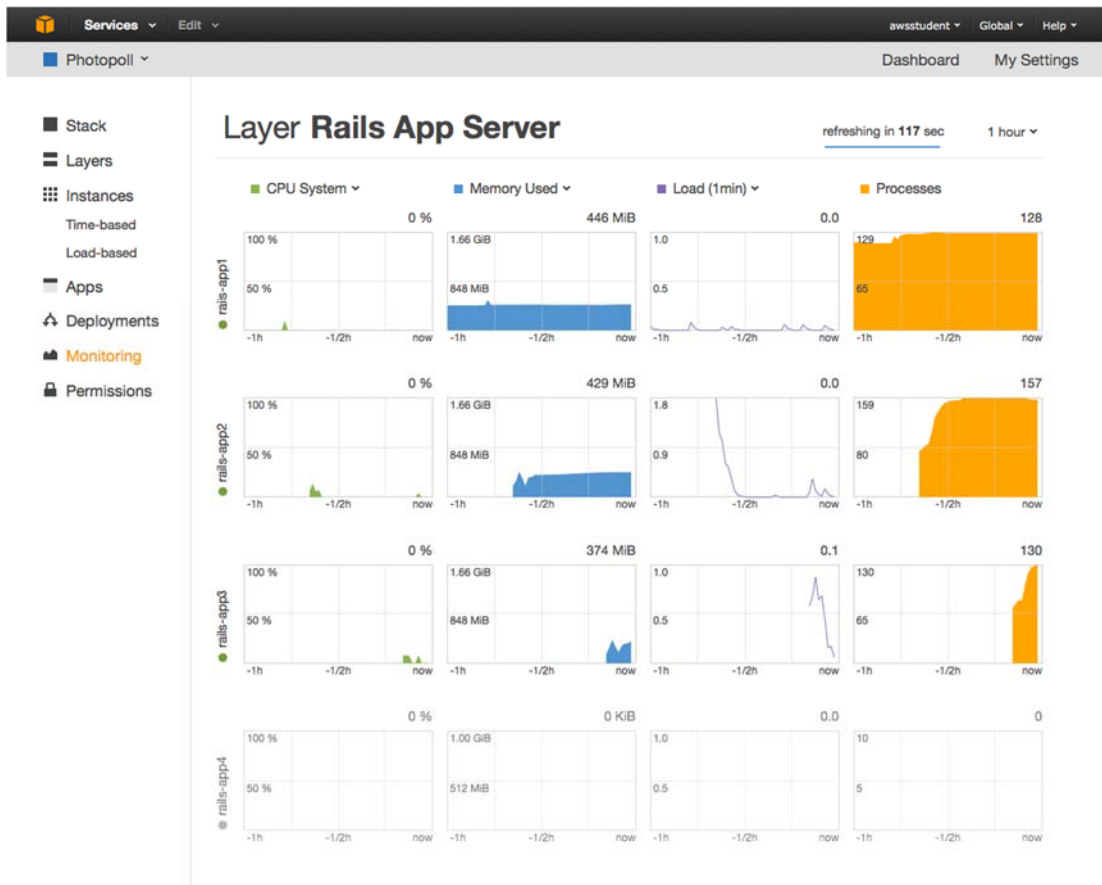
76. You can change the columns to see the 13 different metrics OpsWorks gathers for you on a one-minute interval:

- CPU: Idle, User, System, IO wait, Nice
- Memory: Total, Used, Swap, Free, Buffers
- Load: 1-minute, 5-minute, 15-minute average
- Number of processes

77. Change the time frame you are viewing at in the top right corner to show the last hour.

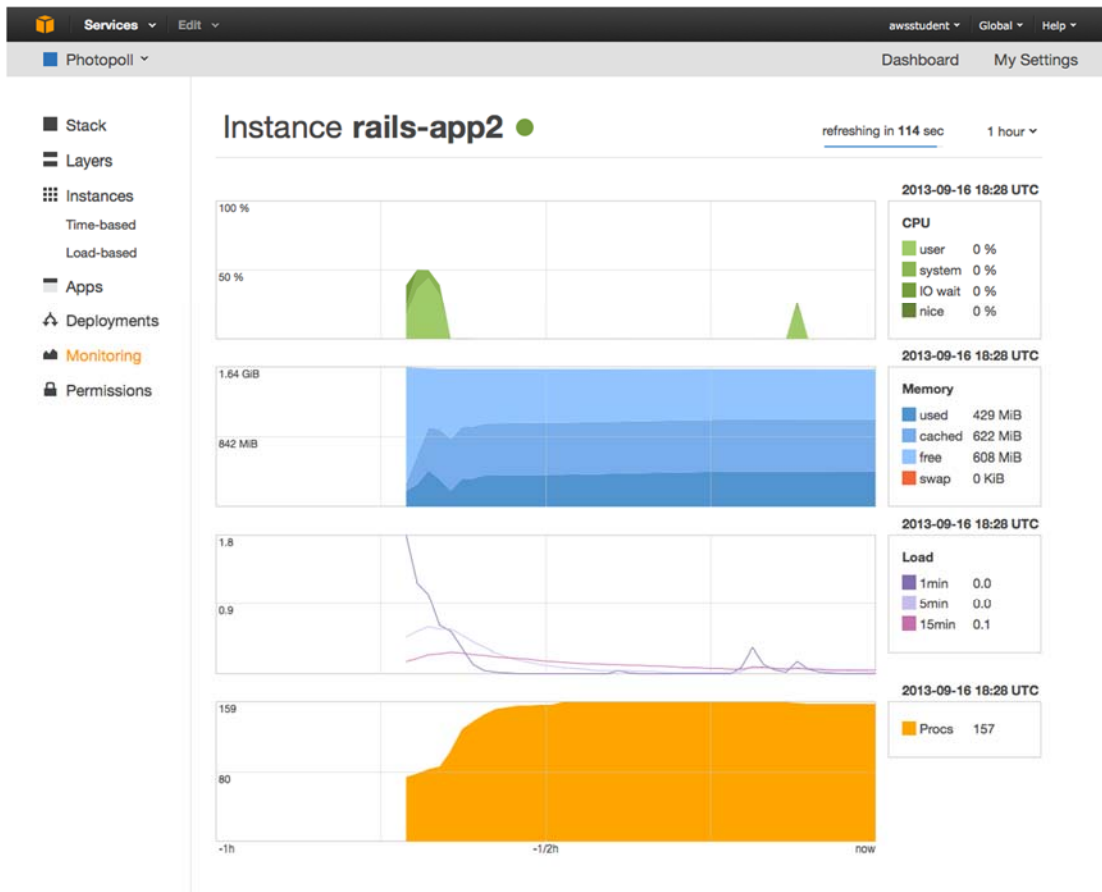


By clicking on one of the Charts of the Rails App Servers you drill into that Layer and see the individual instance's metrics.



What you can see here is that the first instance is running the longest and the last instance, the load based one, did not start. For the second and third instance you can see a load spike. This was the result of the activity experienced when the instance started up.

78. To see all metrics of the second instance click on one of the charts of that instance and you will drill down further.



Here you can see all instance metrics next to each other. This will help you to get an idea why certain values rose or fell.

## Manage volumes and IPs

Now we will look at where the MySQL instance actually stores its data.

79. Click on **Resources** in the navigation.

You will see the volumes used by this stack. In your case this should be one volume. It's the EBS volume the MySQL instance uses to store the images. EBS stands for Elastic Block Store and is offered by Amazon EC2 to store persistent data. Even if you turned your MySQL server off and turn it on later, your database still contains all the images you have stored in it. The instance actually would be a new Amazon EC2 instance. It would go through the setup process during which OpsWorks takes care of reattaching the old volume for you.

The resources view gives you an overview not just of EBS volumes but also Elastic IPs used in your Stack. Elastic IPs can be used if, for example, you wanted to point your domain to an HAProxy load balancer. Most servers in OpsWorks won't need to use an Elastic IP, as OpsWorks instance lifecycle makes sure each instance in your Stack knows the IPs and other information all the time and be configured accordingly.

The resources view also lets you register existing resources in your account to a Stack, so that you can easily migrate to OpsWorks or between Stacks. Each resource can be edited and moved between instances.

## Add and change permissions

In our scenario, the popularity of your image-sharing app grows and grows. You actually have to hire more people to work on it.

80. To add permissions for your new hires, navigate to **Permissions** in the menu.

You will see the `awsstudent` user you are currently using. We have added several other new employees for you in IAM. The Identity and Access Management service enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups and use permissions to allow and deny their permissions to AWS resources.

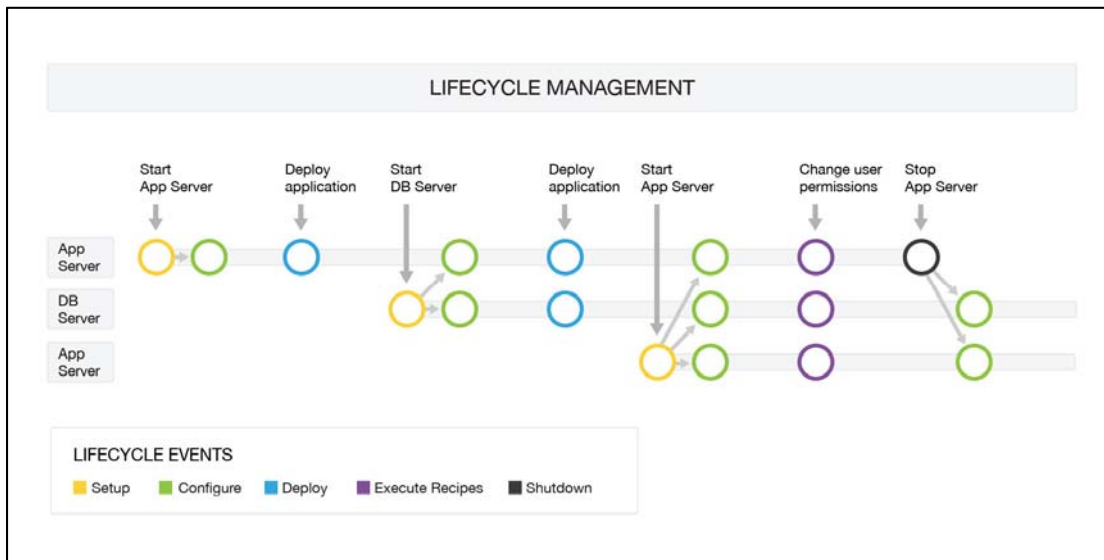
## Stopping instances

You are nearly done with the lab. Let's explore one last thing, stopping instances.

81. Click in **Time-based** in the navigation and uncheck all time ranges. OpsWorks will now shut down the app server, as it is not supposed to be online anymore.
82. Click on **Instances** and you should see the app server in the instances list change its status. A shutdown event will be triggered on that instance. The shutdown event itself gives you the opportunity to execute scripts before the instance is finally done. So you could move some logs over to a remote location or similar cleanup tasks.

You remember the configure event triggered on all instances as soon as an instance in the Stack ran its setup, right? The same is true for a shutdown event. At the same time a configure event is triggered on all other instances. The only difference is that the configure element is not triggered on itself, as this wouldn't make any sense.

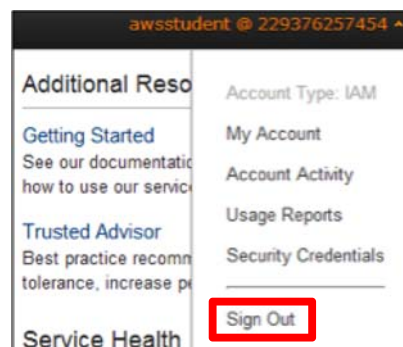




Triggering configure on all other instances will ensure a working configuration on all instances. A load balancer for example would update its configuration and remove the IP of the instance that was shut down.

## End Your Lab

83. Return to the AWS Management Console.
84. From the menu on the upper right of the screen, click **awsstudent @ [YourAccountNumber]** and choose **Sign out** (where **[YourAccountNumber]** is the AWS account generated by qwikLABS):



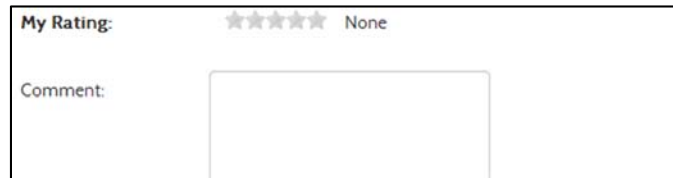
85. Close any active SSH client sessions or remote desktop sessions.

86. Click the **End Lab** button on the qwikLABS lab details page.



87. When prompted for confirmation, click **OK**.

88. For **My Rating**, rate the lab (using the applicable number of stars), optionally type a **Comment**, and click **Submit**.

A feedback form with a white background and a black border. It contains two sections: "My Rating:" with five stars and the word "None" to its right, and "Comment:" with a text input field below it.

**Note:** The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied.

You may close the dialog if you do not wish to provide feedback.

## Conclusion

Congratulations! You now have successfully:

- You were able to launch different kinds of instances serving a web application that relies on a database.
- You updated instances configurations, deployed a new version of your app, looked at graphs, and logged into instances.

## What Should I Do Next?

The following labs will help you understand how to leverage other features of AWS that add additional functionality to your AWS implementation.

To learn how to:

- Create a Windows server instance:
  - Creating Amazon EC2 Instances for Windows
- Create and attach additional storage to your Amazon EC2 instance:
  - Amazon Elastic Block Store (EBS)

- Use AWS Elastic Load Balancer (ELB) to balance Web traffic between two or more instances:
  - Elastic Load Balancing (ELB)
- Enable automatic creation of new Amazon EC2 instances during periods of heavy load:
  - Auto Scaling for Linux
- Create a new server instance by bidding on instance pricing:
  - Launching Amazon EC2 Spot Instances
- Add a relational database to your Virtual Private Cloud:
  - Using Amazon RDS for Applications

For feedback, suggestions, or corrections, please email: [aws-course-feedback@amazon.com](mailto:aws-course-feedback@amazon.com)