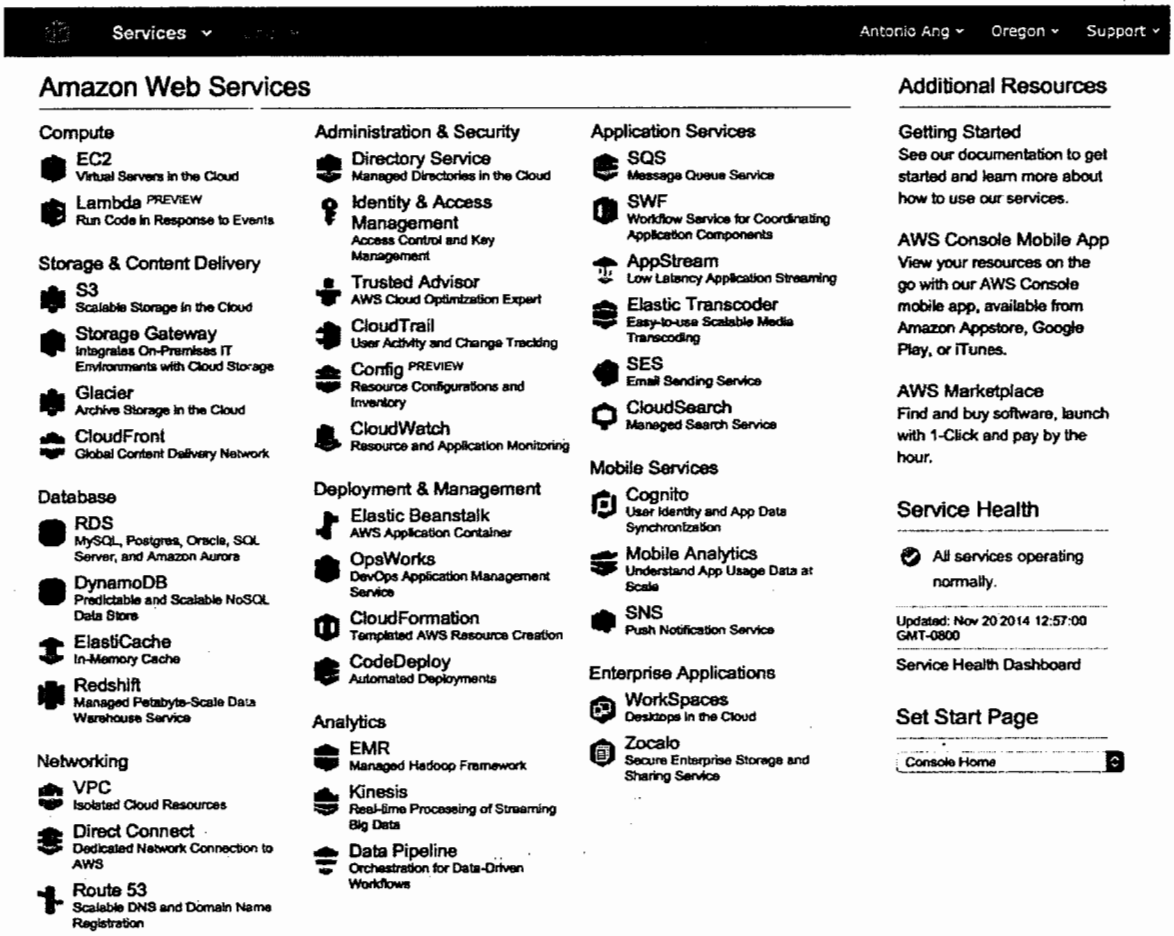
Lab 18

BUILT YOUR FIRST OPSWORKS STACK

STEP 1: Log In to the Amazon Web Service Console

This laboratory experience is about Amazon Web Services and you will use the AWS Management Console in order to complete all the lab steps.



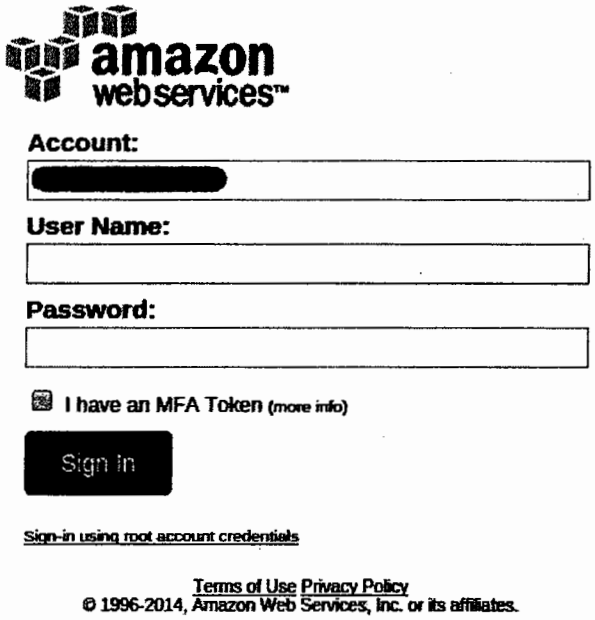
The AWS Management Console is a web control panel for managing all your AWS resources, from EC2 instances to SNS topics. The console enables cloud management for all aspects of the AWS account, including managing security credentials, or even setting up new IAM Users.

Log in to the AWS Management Console

In order to start the laboratory experience, open the Amazon Console by clicking this button:

Open AWS Console

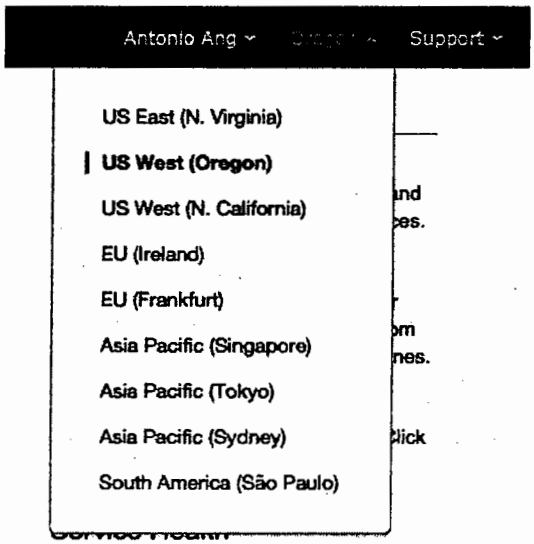
Log in with the username **xxxx** and the password **xxxx**



Select the right AWS Region

Amazon Web Services is available in different regions all over the world, and the console lets you provision resources across multiple regions. You usually choose a region that best suits your business needs to optimize your customer's experience, but you must use the region **US West (Oregon)** for this laboratory.

You can select the **US West (Oregon)** region using the upper right dropdown menu on the AWS Console page.



STEP 2: Create a VPC

Amazon OpsWorks lets you easily orchestrate the different parts of your application using**Chef** to perform the actual automation. It presents the different AWS resources that make up your app as multiple layers, each composed of resources. A typical app might have two layers, an app server layer (where your Ruby/NodeJS/Python/PHP app actually runs) and a database layer (backed by RDS). Typically, you'd manage each instance and RDS installation separately, but with OpsWorks you can manage all instances in the "app server" layer together.

The advantage of using Chef is that you can use AWS' published OpsWorks cookbooks, open source community cookbooks, build your own, or mix and match. AWS publishes cookbooks for typical Rails applications, Nginx proxies, memcached servers, monitoring, haproxy, and more.

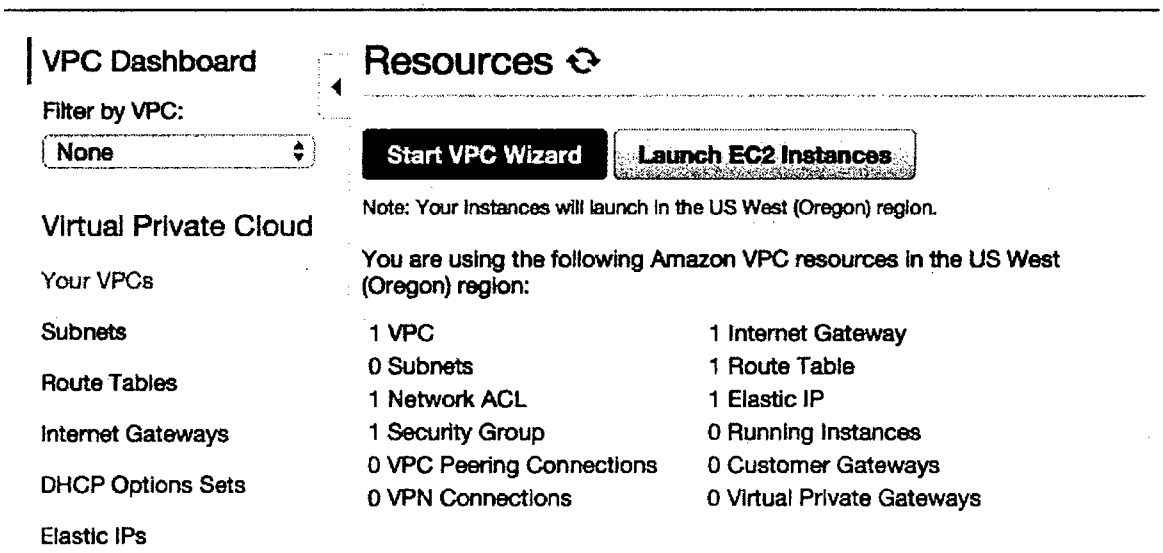
But before we get started building our first OpsWork stack, I'd like to remind you that it is just a collection of resources, and often doesn't create underlying resources like VPC networks automatically do. So we'll need to digress and make a VPC for all our instances to inhabit first.

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center -- with the benefits of using the scalable infrastructure of AWS. It is logically isolated from other virtual networks in the AWS cloud.

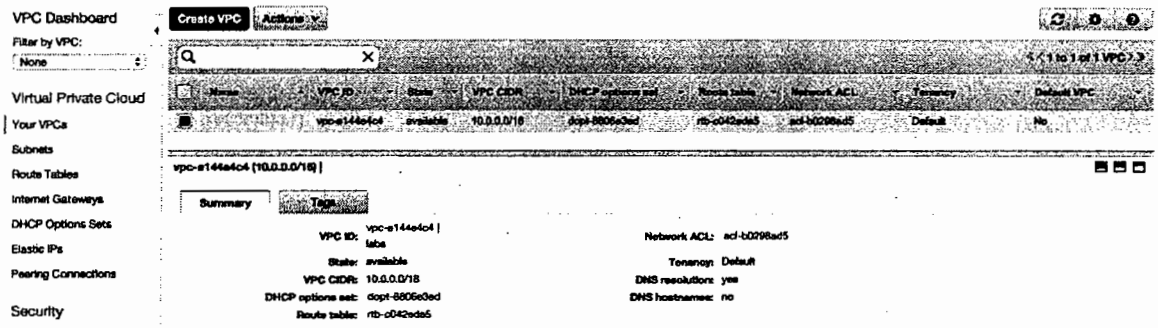
You can create a new VPC using the AWS Management Console.

Select the VPC service from the Management Console dashboard:

From the VPC dashboard, click on **Your VPCs** link in the sidebar menu.

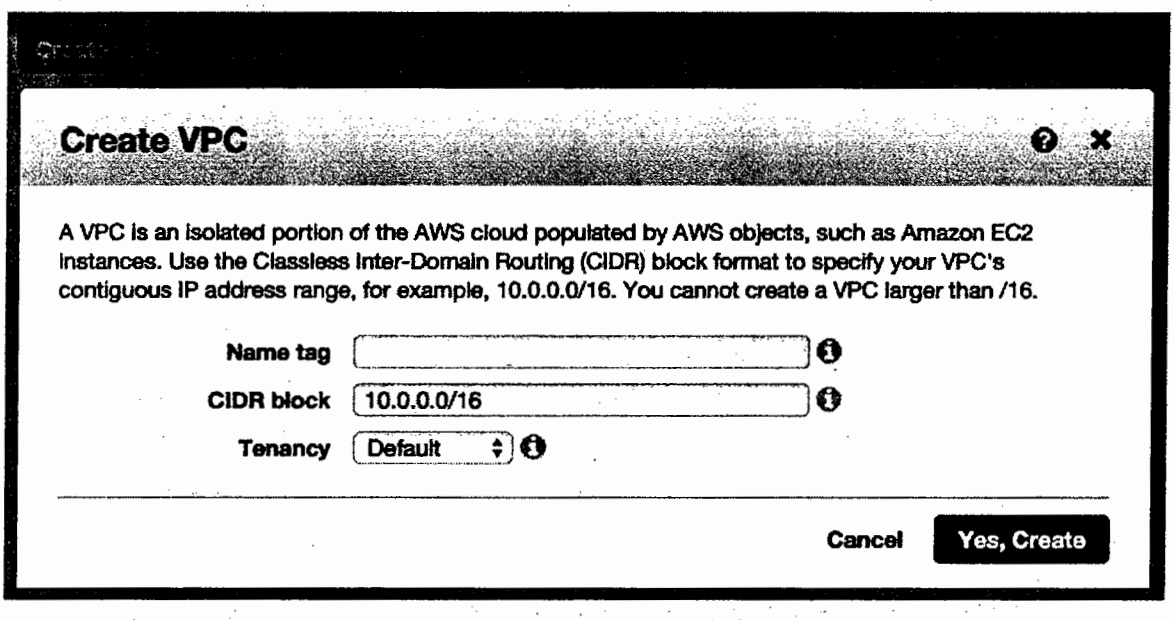


Your VPCs page lists all previously created VPCs (any new AWS account comes with a default fully-working VPC); click on the **Create VPC** blue button to begin creating a new VPC.



In the Create VPC dialog box, specify the following VPC details as necessary, then click **Yes, Create**.

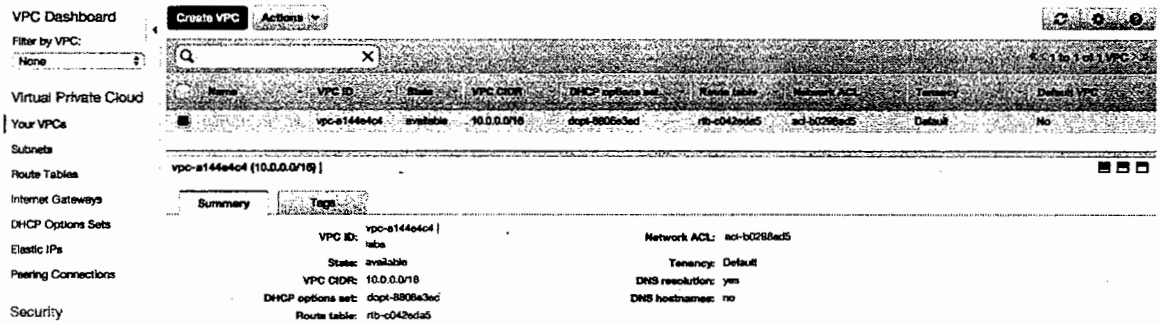
- ✓ **Name tag**: **fizzbuzz-vpc**. This is the name for your VPC; doing so creates a tag with a key of Name and the value that you specify.
- ✓ **CIDR block**: **10.0.0.0/24**. You should specify a CIDR block from the private (non-publicly routable) IP address ranges as specified in RFC 1918.
- ✓ **Tenancy**: **default**. Dedicated tenancy ensures your instances run on single-tenant hardware.



Amazon creates the requested VPC and the following linked services:

- ✓ a **DHCP options set** (this set enables DNS for instances that need to communicate over the VPC's Internet gateway)
- ✓ a **Route Table** (it contains a set of rules, called *routes*, that are used to determine where network traffic is directed)
- ✓ a **Network ACL** (it is a list of rules to determine whether traffic is allowed in or out of any subnet associated with the network ACL)

Note that no Subnets or Internet Gateways are automatically created -- you need to add them autonomously.



Now you are ready to create your VPC subnets and customize the routing table.

STEP 3: Create a VPC subnet

A **VPC subnet** is a range of IP addresses in your VPC. You can add one or more subnets in each Availability Zone, but each subnet must reside entirely within one Availability Zone and cannot span zones. **Availability Zones** are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.

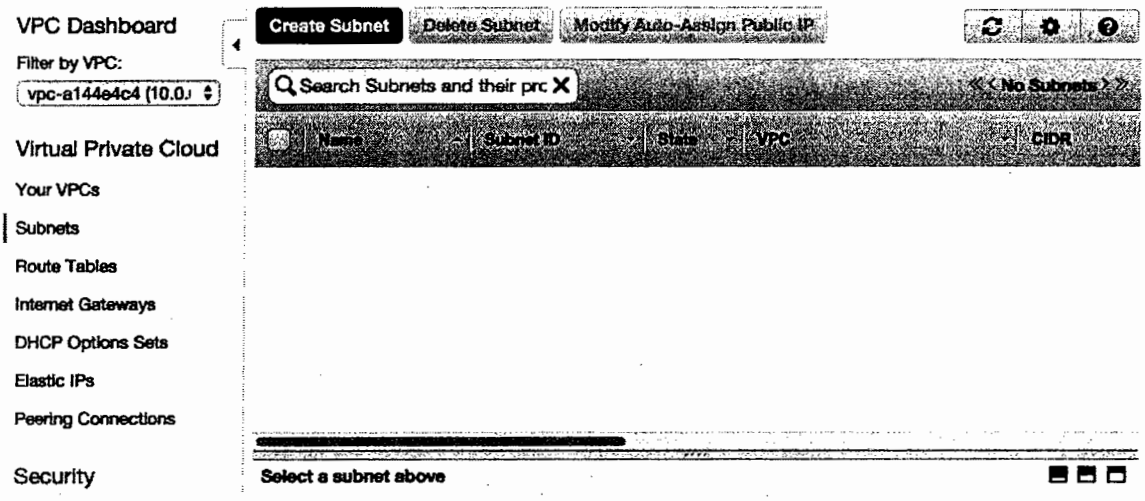
You can create a new subnet for your previously created VPC using the AWS Management Console.

Select the VPC service from the Management Console dashboard:

From the VPC dashboard, click the **Subnets** link in the sidebar menu.

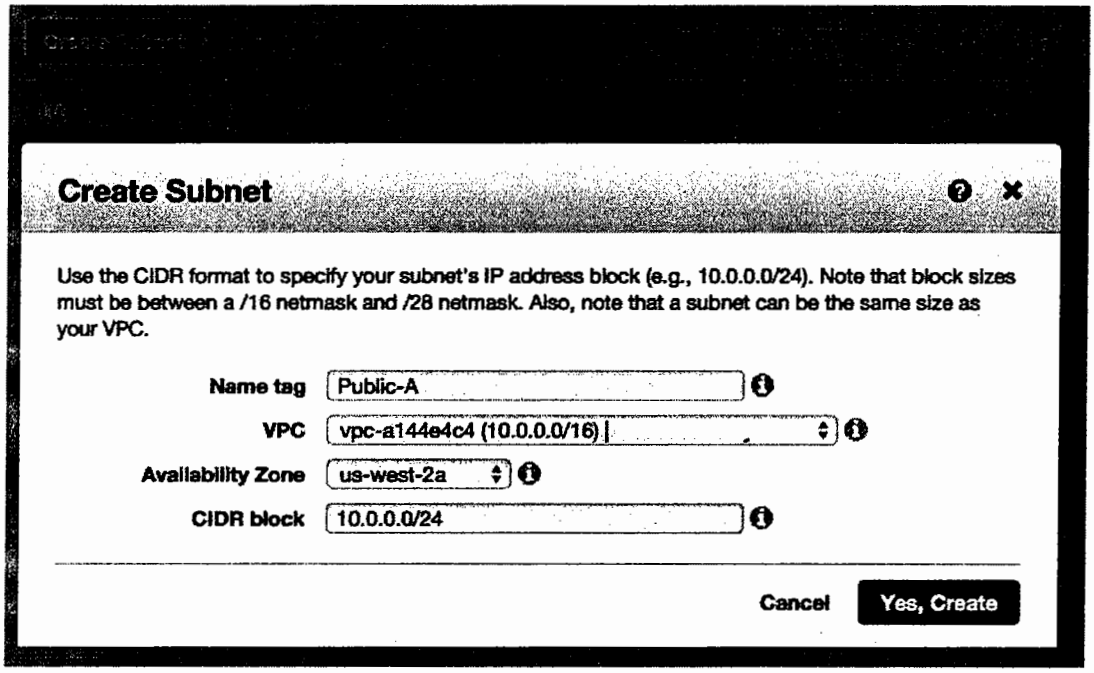
Your Subnets page lists all previously created subnets, you can use the **Filter by VPC** feature for listing only the services linked to a specific VPC.

Click on the **Create Subnet** blue button to begin creating a new subnet.

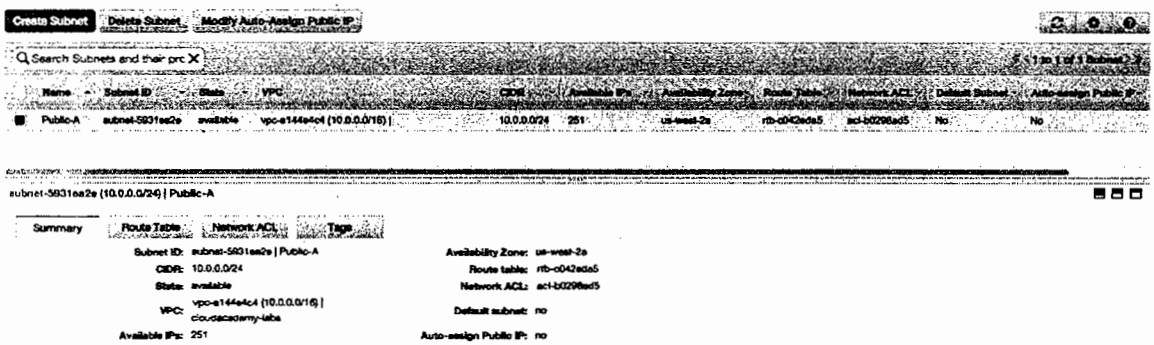


In the Create Subnet dialog box, specify the following Subnet details then click **Yes, Create**.

- ✓ **Name tag**: **appservers**. This is the name for your subnet; doing so creates a tag with a key of Name and the value that you specify.
- ✓ **VPC**: **fizzbuzz-vpc**.
- ✓ **Availability Zone**: **us-west2a**.
- ✓ **CIDR block**: **10.0.0.0/25**. You should specify a CIDR block in the selected VPC.



As you can see, the created subnet is automatically attached to the default VPC Route table and the default Network ACL.



STEP 4: Create a VPC Internet Gateway

An **Internet Gateway** is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the Internet. It imposes no availability risks or bandwidth constraints on your network traffic. An Internet gateway serves two purposes: to provide a target in your VPC route tables for Internet-routable traffic, and to

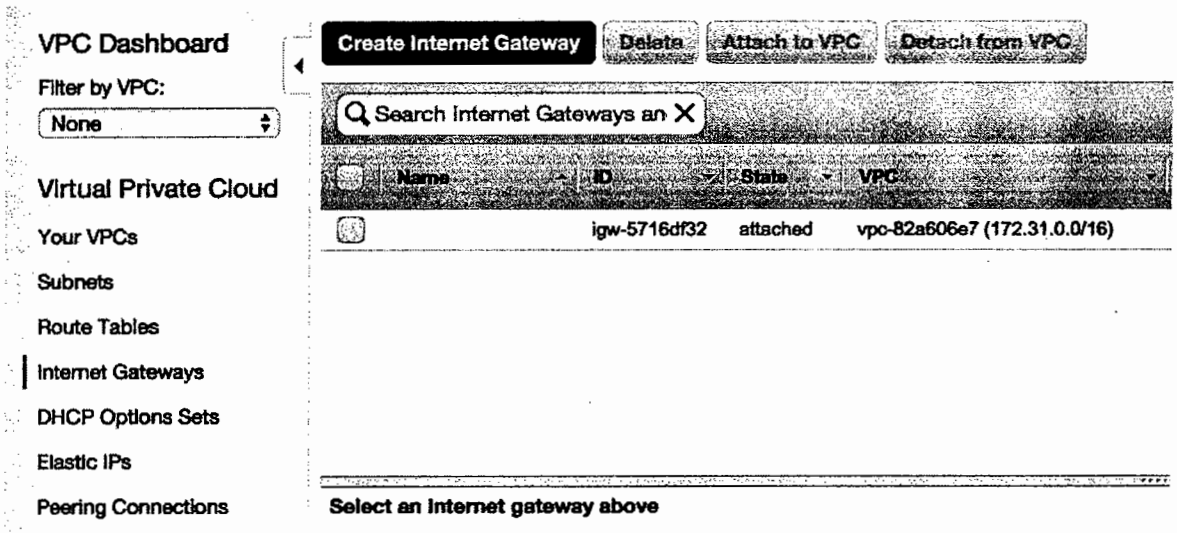
perform network address translation (NAT) for instances that have been assigned public IP addresses.

You can create a new **Internet Gateway** for your previously created VPC using the AWS Management Console.

Select the VPC service from the AWS Management Console dashboard:

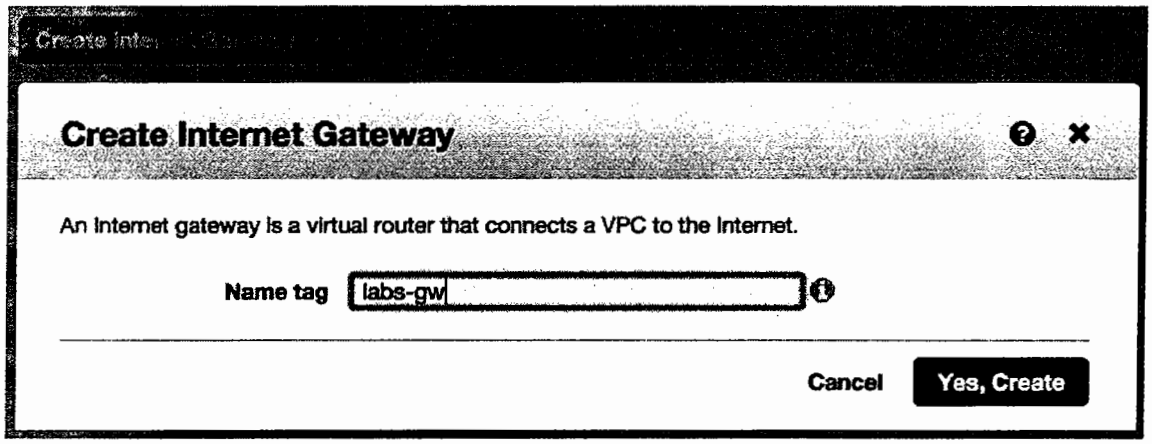
From the VPC dashboard, click the **Internet Gateways** link in the sidebar menu.

The **Internet Gateways** page lists all previously created gateways. Click on the **Create Internet Gateway** blue button to begin creating a new gateway.



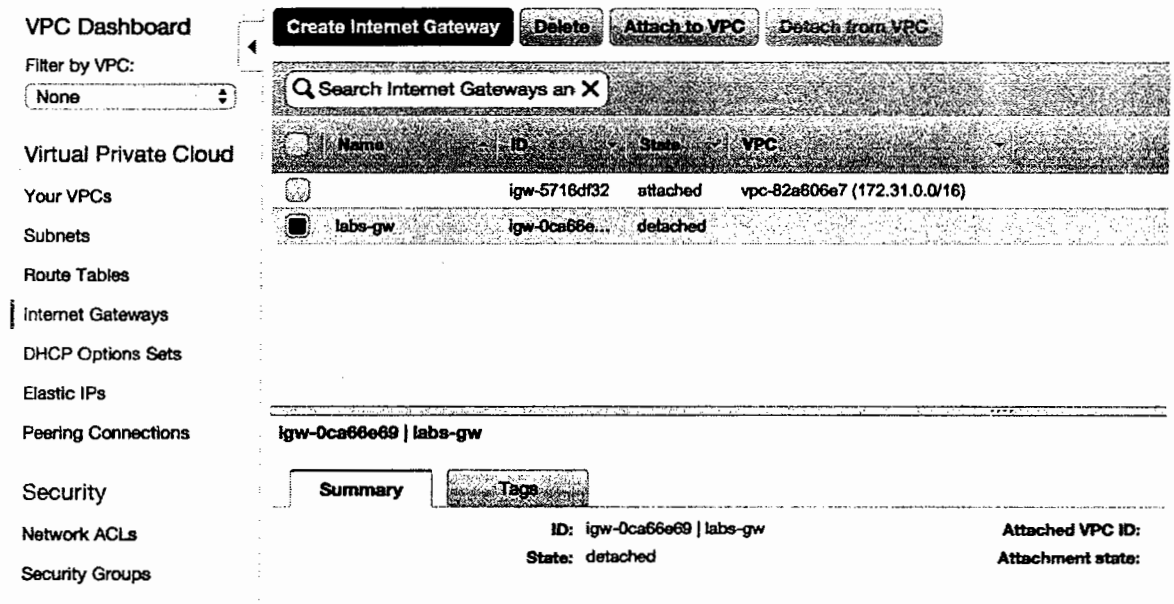
Creating a gateway is a one step operation, you only need to choose a meaningful name.

Use **fizzbuzz-gateway** as **Name tag** and then click **Yes, Create**.

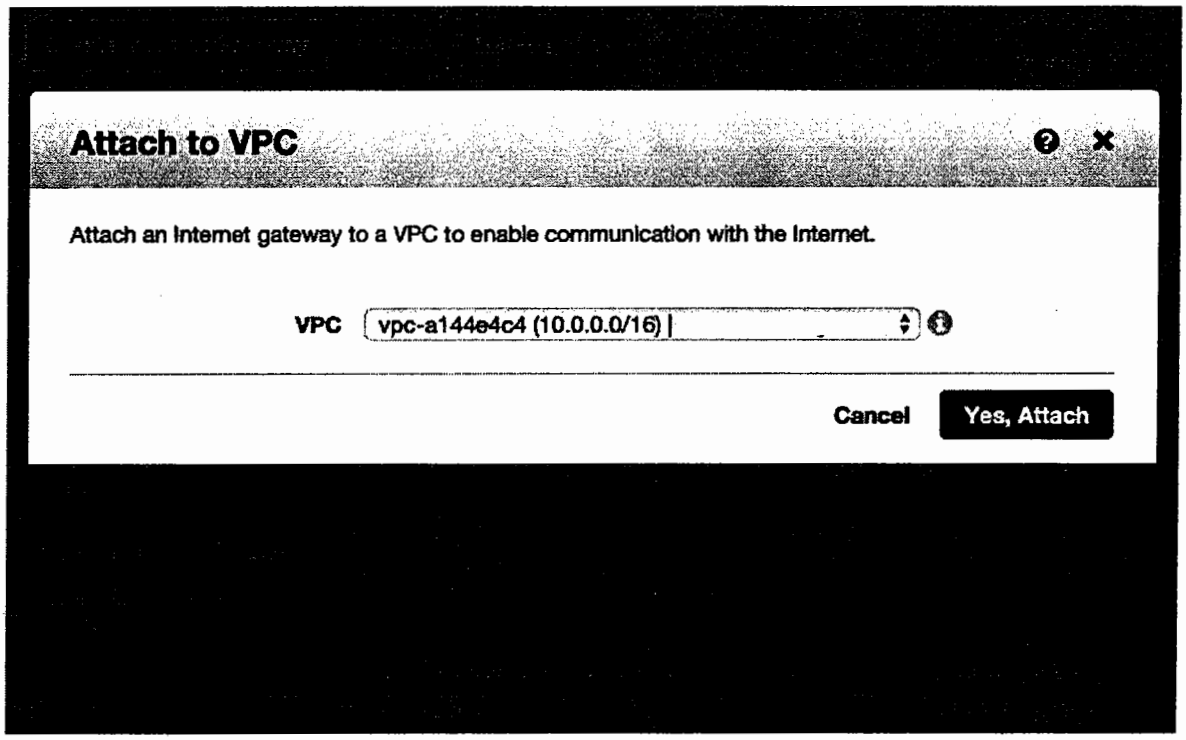


How to attach the Internet Gateway to a VPC

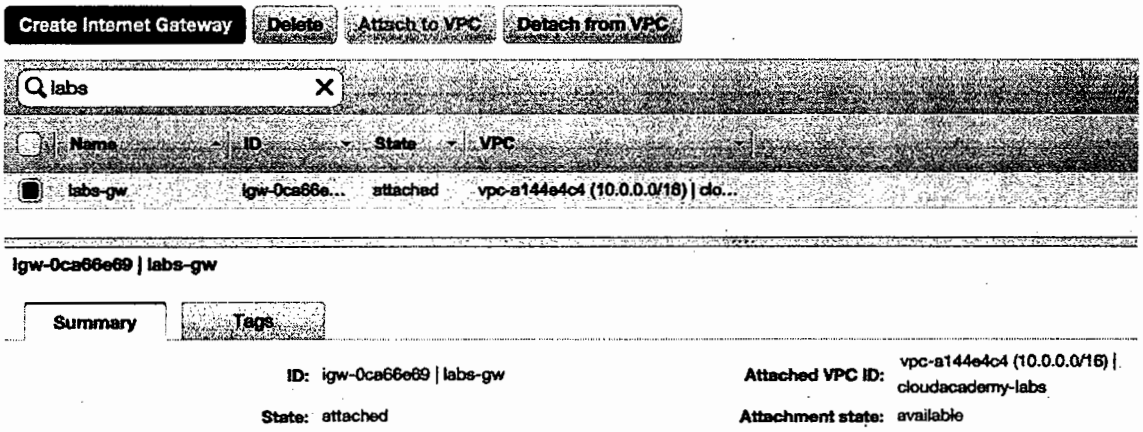
Select the Internet gateway that you just created, and then click **Attach to VPC**.



In the Attach to VPC dialog box, select the VPC **fizzbuzz-vpc** from the list, and then click **Yes, Attach**.



Your new Internet Gateway is ready to be used by the EC2 instances of the selected VPC.



STEP 5: Connect the Internet Gateway to the VPC Route Table

To use an **Internet gateway** your subnet's **route table** must contain a route that directs Internet-bound traffic to the Internet gateway. You can scope the route to all destinations not explicitly known to the route table (0.0.0.0/0), or you can scope the route to a narrower range of IP addresses; for example, the public IP addresses of your company's public endpoints outside

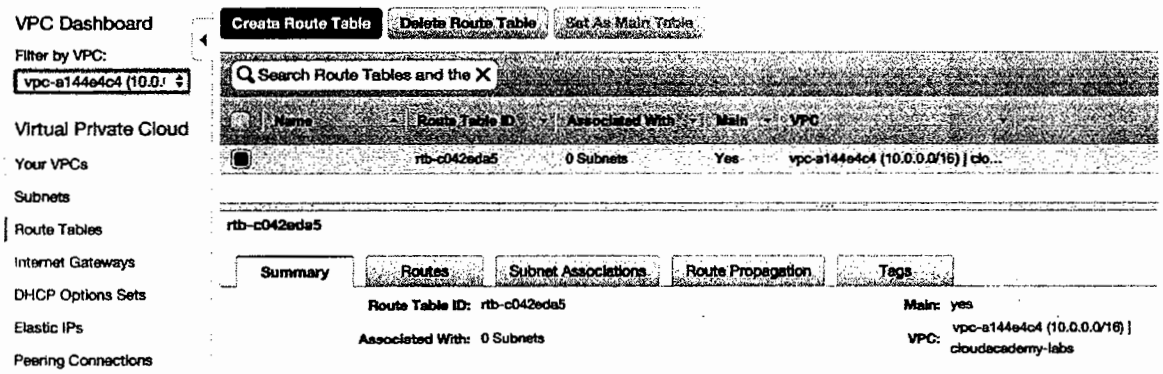
of AWS, or the Elastic IP addresses of other Amazon EC2 instances outside your VPC. If your subnet is associated with a route table that has a route to an Internet gateway, it's known as a **public subnet**.

You can add routes to your previously created VPC **Route Table** using the AWS Management Console.

Select the VPC service from the AWS Management Console dashboard:

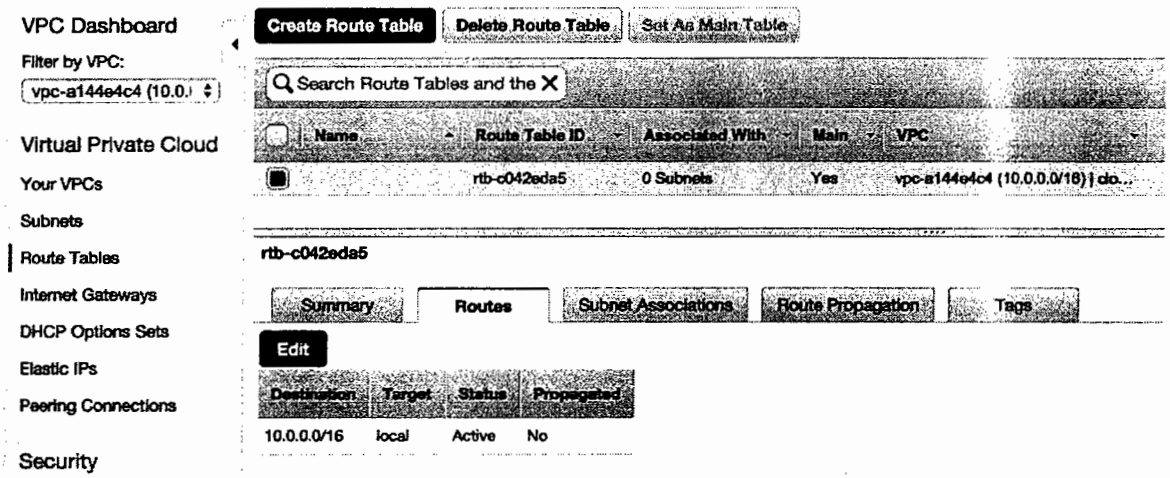
From the VPC dashboard, click the **Route tables** link in the sidebar menu.

The **Route tables** page lists all previously created route tables. In order to select the Route Table of your **fizzbuzz-vpc** VPC, you can check the VPC column or use the **Filter by VPC**feature in the left sidebar for listing the Route Tables attached to **fizzbuzz-vpc**.

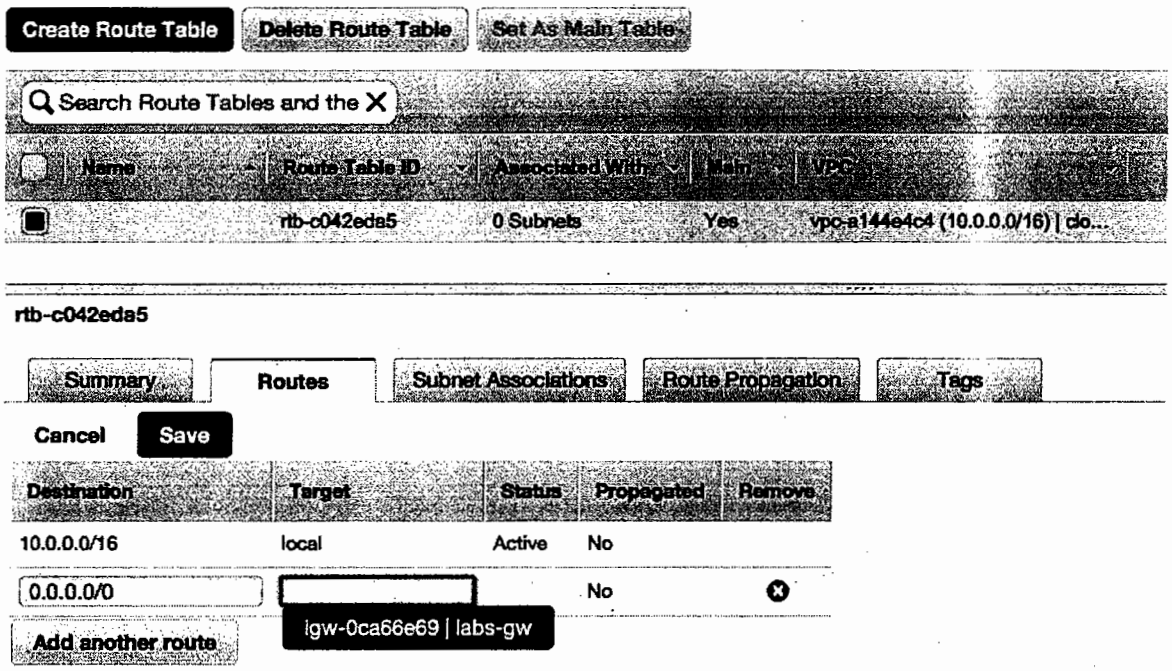


Select the **Main** route table to show its detailed information and then select the **Routes** tab pane.

Routes is set of rules which are used to determine where network traffic is directed. For adding a new route, click the blue **Edit** button.



Enter **0.0.0.0/0** a destination CIDR block and then select the previously created Internet Gateway from the Target list. Click **Save** when you're done.



Thanks to the new route rule, all VPC external traffic will be routed to the Internet Gateway and then to the Internet.

rtb-c042eda5

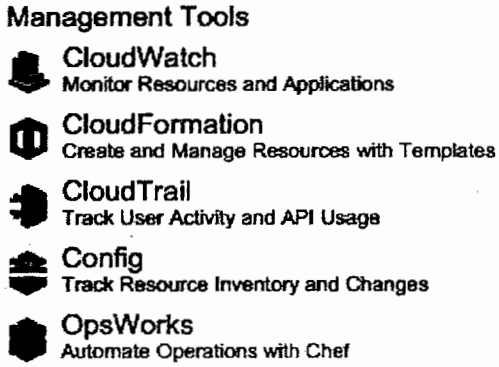
Summary | Routes | Subnet Associations

Edit

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	igw-0ca66e69	Active	No

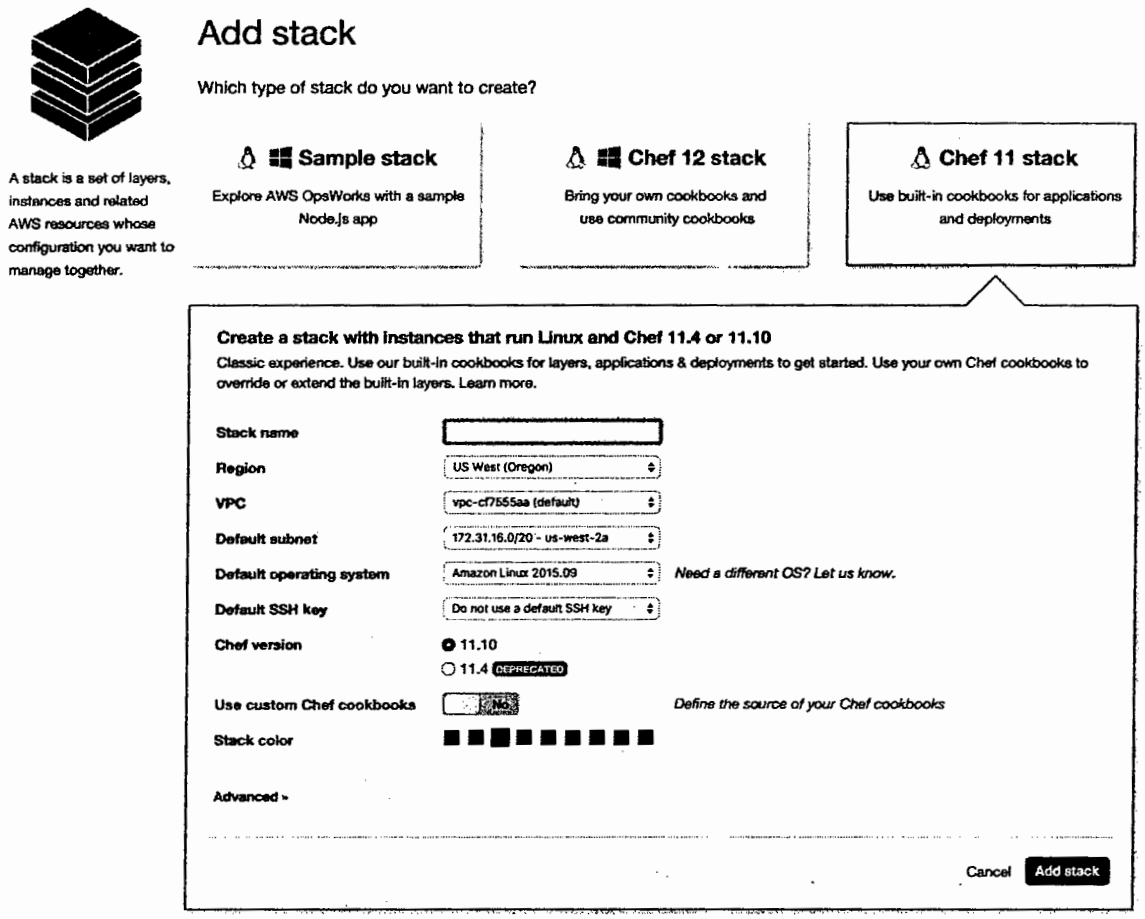
STEP 6: Create the Stack

Now we'll get started building your first OpsWorks stack. To start, go to the AWS console and look for the OpsWorks logo.



Once you've reached the OpsWorks dashboard, we'll create a new stack by clicking on the "Add your first stack" button.

In the opened screen select Chef 11 stack



You should insert the following info in the other fields:

Stack name: **fizzbuzzapp**

VPC: **fizzbuzz-vpc**

Defaul subnet: **appservers**

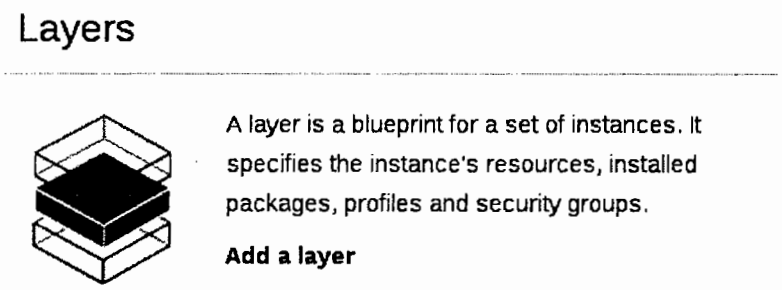
You can leave the other fields with the default settings

Now you can click on Add stack. In the next step, you'll add the app server layer to your OpsWorks stack, and you'll almost be ready to run it.

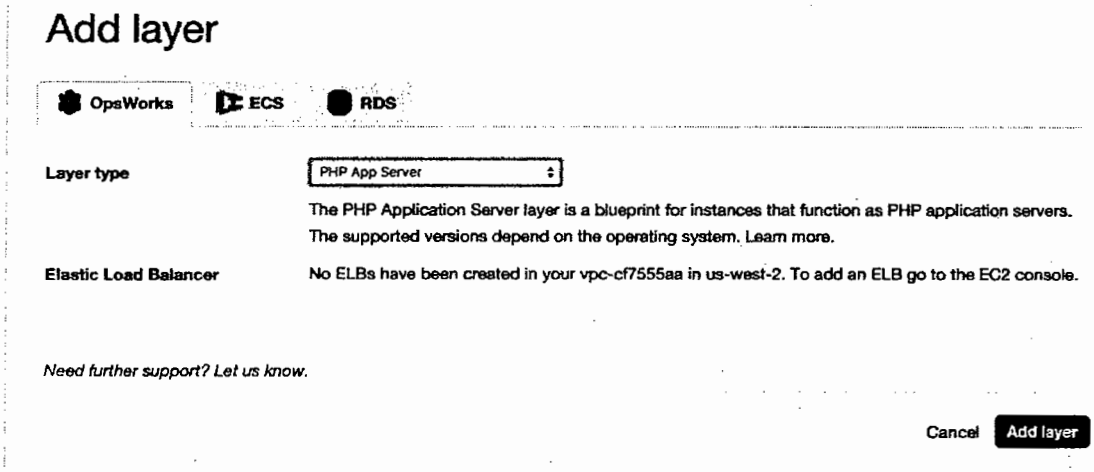
STEP 7: Create a Layer

In the last step, you created an OpsWorks Stack. A Stack is a collection of Layers that make up the different parts of your application. In this lab, we'll use a sample PHP application.

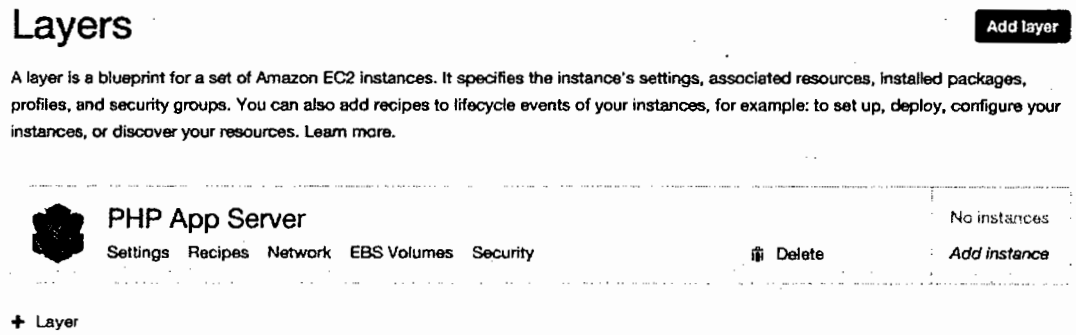
First, add a new layer from the OpsWorks console for your **fizzbuzzapp** stack.



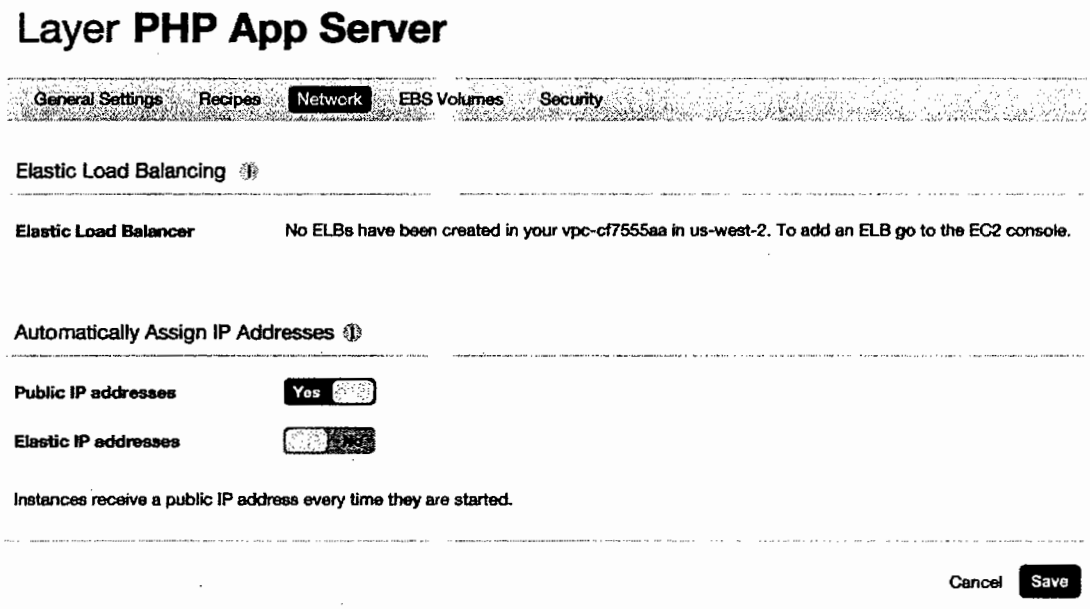
There is a default role in the drop-down menu for PHP app server, select that.



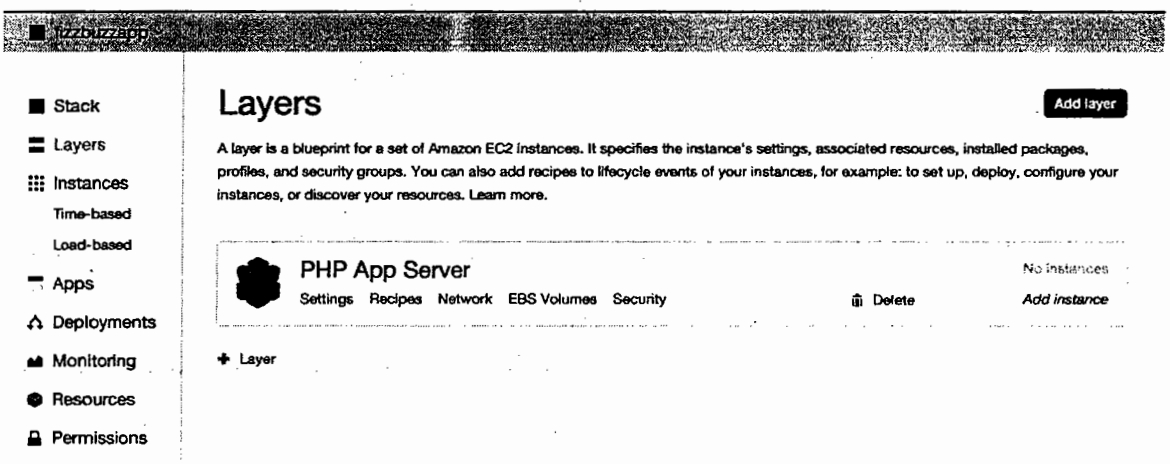
If this were a real application, we'd want to create an ELB to use. OpsWorks can automatically add and remove instances from an Elastic Load Balancer when they are finished deploying or being deleted, which beats manually managing instances.



The new layer is ready to go, but because we did not create an ELB for this layer, we need to check if our instances have a public IP address, otherwise they will not be publically accessible. To do that, click the "Network" button to open the layer's network settings.



Make sure that the option "Public IP addresses" is set to yes and hit save. Now that instances will be reachable, we can create the instances to run your app on. Let's create a new t2.micro instance to run our demo on. To do this, click on Layers, and then in the Add instance button:



Use the following settings to create the instance. The default subnet here is fine, just make sure you've got the right instance size and you'll be set.

PHP App Server

No instances. Add an instance.

New | Existing OpsWorks | EC2 instances and own servers

Hostname: php-app1

Size: t2.micro — Select the compute and memory size for your instance. Learn more.

Subnet: 10.0.0.0/25 - us-east-1a - appserver

Advanced »

Cancel | Add Instance

Instances — Start All Instances

An instance represents a server. It can belong to one or more layers, that define the instance's settings, resources, installed packages, profiles and security groups. When you start the instance, OpsWorks uses the associated layer's blueprint to create and configure a corresponding EC2 instance. Learn more.

1	0	0	0	1	0
Instances	online	setting up	shutting down	stopped	error

PHP App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	stopped	t2.micro	24/7	us-west-2a	-	start delete

+ Instance

Once you've added an instance, just click "Start All Instances" to get going. You can start/stop individual instances, but with only one instance it doesn't matter much.

PHP App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	booting	t2.micro	24/7	us-east-1a	-	stop

+ Instance

Booting your instance should take no more than ten minutes. While you wait, you'll see it go through a couple of statuses. The "running_setup" is when the Chef cookbooks are running on the instance.

PHP App Server

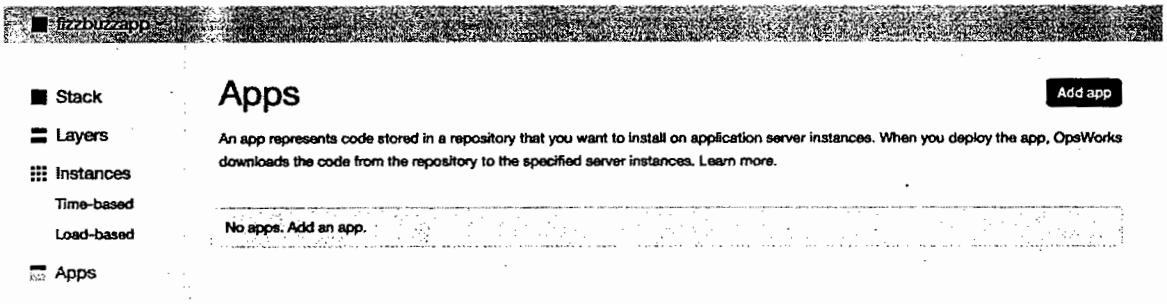
Hostname	Status	Size	Type	AZ
php-app1	online	t2.micro	24/7	us-east-1e

In this step, we've created a layer for our app servers and created an instance to host our application. In the next step, we'll tell OpsWorks how to deploy our application.

STEP 8: Create an App

An OpsWorks application lets you deploy your app (in our case a PHP demo app) to as many servers as you need. Under the hood, OpsWorks has a Chef cookbook that handles cloning your git repository and restarting the app server. Private repositories are supported, but we'll be using the free AWS Labs PHP Demo from GitHub.

First, find the Apps section in the sidebar and click "Add Application".



We'll need to fill in some information before we continue deploying the application. We'll name it **fizzbuzz** in keeping with our theme so far. For this lab, we won't set up a data source (typically a SQL database) but it will be covered in a more advanced lab.

Here is the public Repository URL:

https://github.com/awslabs/opsworks-demo-php-simple-app

Add App

Settings

Name	fizzbuzz
Type	PHP
Document root	Optional

Data Sources

Data source type	RDS OpsWorks None

Application Source

Repository type	Git
Repository URL	https://github.com/awslabs/opsworks-c
Repository SSH key	Optional
Branch/Revision	version1

It's important to use the **version1** branch since that repository doesn't have a master branch -- or deployments will fail.

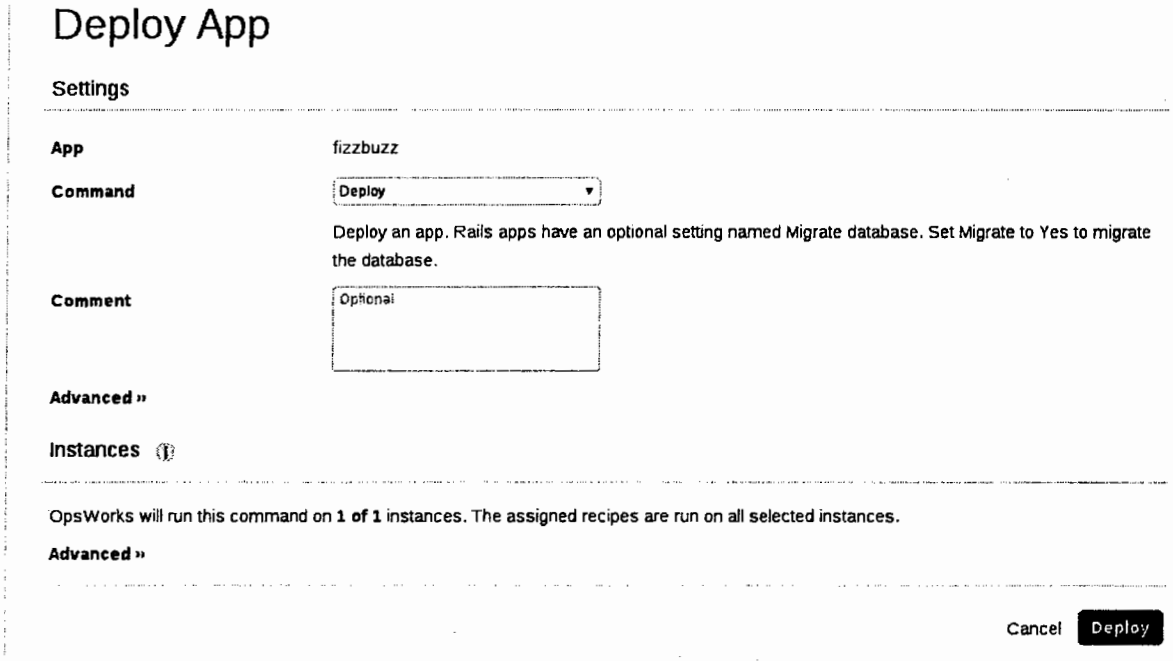
Now we've told OpsWorks where our code is and what servers to run it on, it's time to deploy. In the next step, we'll deploy and test the demo app.

STEP 9: Deploy fizzbuzz-app

Now that you've told AWS OpsWorks where the code for your application can be found, it's time to send the command to your instance(s) to download and run your app.

Name	Type	Data Source	Last Deployment	Actions
fizzbuzz	PHP			deploy edit delete

Click "Deploy" to create a new deployment for **fizzbuzz**. You can also use the deployment workflow to send arbitrary commands to your instances.



Once you hit deploy, it should only take a couple minutes for your instance to receive the new application and restart its web server. To check that it worked, go back to the "Instances" tab and click on the public IP address.

You should see a page like the above. In the next step, we'll see how to clean up the resources that comprise your stack.

STEP 10: Delete the OpsWorks Stack

Before a stack can be deleted, all the Apps and instances it contains must be deleted.

First, let's delete the application. Go to the Apps tab and find the "Delete" button on fizzbuzz.

Apps

An app represents code stored in a repository that you want to install on application server instances. When you deploy the app, OpsWorks downloads the code from the repository to the specified server instances. Learn more.

Name	Type	Data Source	Last Deployment	Actions
fizzbuzz	PHP		2015-03-29 02:10:27 UTC	deploy edit delete

Are you sure that you want to delete fizzbuzz?

If you delete this app, all your configuration settings will be lost. In order to remove the app an undeploy event will be triggered on all running instances.

Cancel **Delete**

After the app is deleted, go to the Instances tab to stop the php-app1 instance.

PHP App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	online	t2.micro	24/7	us-east-1e	52.4.143.231	stop ssh

Are you sure you want to stop php-app1?

All data not stored on EBS volumes will be lost.

Cancel **Stop**

The instance will take a couple minutes to stop, when it does you can delete it.

PHP App Server

Hostname	Status	Size	Type	AZ	Public IP	Actions
php-app1	stopped	t2.micro	24/7	us-east-1e	52.4.143.231	start delete

Are you sure that you want to delete php-app1?

If you delete this instance all logs and data associated with it will be lost.

Cancel **Delete**

Now that all the resources have been deleted, it's safe to delete the stack itself.

fizzbuzzapp

Run Command **Stack Settings** **Delete Stack**

Are you sure that you want to delete fizzbuzzapp?

If you delete this stack, all your settings will be lost.

Cancel **Delete**

All our resources are now tidy. In this lab, we've seen how OpsWorks divides and manages resources based on where they fit in your stack, and we've learned how OpWorks can ease tasks like deployment and scaling. In a future lab, you'll learn about time-based instance scaling, using database (RDS) layers, and handling traffic with Elastic Load Balancers.

*****