

## Lab 15

### DEPLOY WORDPRESS USING CLOUDFORMATION



## 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 screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, and user information 'Antonio Ang', 'Oregon', and 'Support'. The main content area is titled 'Amazon Web Services' and is divided into several categories of services, each with an icon and a brief description:

- Compute**: EC2 (Virtual Servers in the Cloud), Lambda PREVIEW (Run Code in Response to Events).
- Storage & Content Delivery**: S3 (Scalable Storage in the Cloud), Storage Gateway (Integrates On-Premises IT Environments with Cloud Storage), Glacier (Archive Storage in the Cloud), CloudFront (Global Content Delivery Network).
- Database**: RDS (MySQL, Postgres, Oracle, SQL Server, and Amazon Aurora), DynamoDB (Predictable and Scalable NoSQL Data Store), ElastiCache (In-Memory Cache), Redshift (Managed Petabyte-Scale Data Warehouse Service).
- Networking**: VPC (Isolated Cloud Resources), Direct Connect (Dedicated Network Connection to AWS), Route 53 (Scalable DNS and Domain Name Registration).
- Administration & Security**: Directory Service (Managed Directories in the Cloud), Identity & Access Management (Access Control and Key Management), Trusted Advisor (AWS Cloud Optimization Expert), CloudTrail (User Activity and Change Tracking), Config PREVIEW (Resource Configurations and Inventory), CloudWatch (Resource and Application Monitoring).
- Deployment & Management**: Elastic Beanstalk (AWS Application Container), OpsWorks (DevOps Application Management Service), CloudFormation (Templated AWS Resource Creation), CodeDeploy (Automated Deployments).
- Analytics**: EMR (Managed Hadoop Framework), Kinesis (Real-time Processing of Streaming Big Data), Data Pipeline (Orchestration for Data-Driven Workflows).
- Application Services**: SQS (Message Queue Service), SWF (Workflow Service for Coordinating Application Components), AppStream (Low Latency Application Streaming), Elastic Transcoder (Easy-to-use Scalable Media Transcoding), SES (Email Sending Service), CloudSearch (Managed Search Service).
- Mobile Services**: Cognito (User Identity and App Data Synchronization), Mobile Analytics (Understand App Usage Data at Scale), SNS (Push Notification Service).
- Enterprise Applications**: WorkSpaces (Desktops in the Cloud), Zocalo (Secure Enterprise Storage and Sharing Service).

On the right side, there's a section titled 'Additional Resources' with links to 'Getting Started', 'AWS Console Mobile App', 'AWS Marketplace', 'Service Health', and 'Set Start Page'. The 'Service Health' section shows 'All services operating normally' and a timestamp 'Updated: Nov 20 2014 12:57:00 GMT-0800'.

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 **xxxxx** and the password **xxxxx**



Account:

User Name:

Password:

☐ I have an MFA Token ([more info](#))

Sign In

[Sign-in using root account credentials](#)

[Terms of Use](#) [Privacy Policy](#)  
© 1996-2014, Amazon Web Services, Inc. or its affiliates.

## 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.

Antonio Ang ▾ Oregon ▾ Support ▾

- US East (N. Virginia)
- | **US West (Oregon)**
- US West (N. California)
- EU (Ireland)
- EU (Frankfurt)
- Asia Pacific (Singapore)
- Asia Pacific (Tokyo)
- Asia Pacific (Sydney)
- South America (São Paulo)

## STEP 2: Create a CloudFormation Stack

CloudFormation is a great tool for tying together related AWS resources such as instances, DynamoDB tables, IAM users, and more. In this lab we'll deploy the WordPress CMS using a CloudFormation template.

A CloudFormation stack can have many types of resources, including EC2 instances, S3 buckets, IAM users or policies, and Route53 record sets. The template we're using will only contain one EC2 instance and its security group, but stacks can be as complex as you like, containing dozens of different resource types.

First, download the CloudFormation template that we are going to use.

Now open the CloudFormation console. From the dashboard, click on the logo.

### Management Tools



CloudWatch

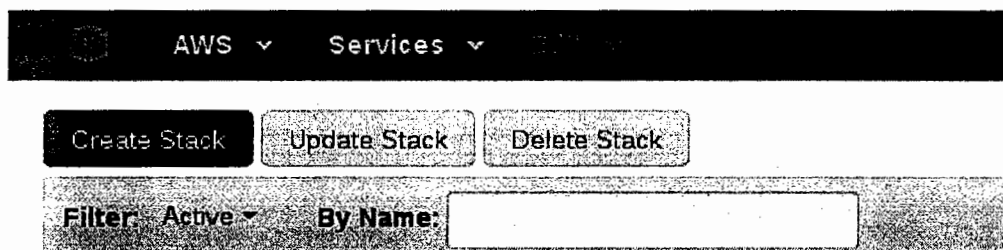
Monitor Resources and Applications



CloudFormation

Create and Manage Resources with Templates

You're now in the CloudFormation dashboard where we will be for the rest of the lab. Let's create a new stack by clicking "Create Stack".



Now click on **choose file** and select the template file that you just downloaded and click **next**.

## Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

**Design a template** Use AWS CloudFormation Designer to create or modify an existing template. [Learn more.](#)

[Design template](#)

**Choose a template** A template is a JSON-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

☐ Select a sample template

☒ Upload a template to Amazon S3

[Choose File](#)

launch-wordpress.template

☐ Specify an Amazon S3 template URL

In the Parameters page, you'll fill in the values for Stack name, Database password, Root password, DB username, instance type, Keypair name, SSH Location, SubnetID and VPC ID. Feel free to customize the settings or use the default ones. Just make sure to select a subnet that is inside the selected VPC otherwise you will have problems.

**Note: The default password is Start1234**

In the end, you should have something like this:

Stack name

## Parameters

DBName  The WordPress database name

DBPassword  The WordPress database admin account password

DBRootPassword  MySQL root password

DBUser  The WordPress database admin account username

InstanceType  WebServer EC2 instance type

KeyName  Name of an existing EC2 KeyPair to enable SSH access to the instances

SSHLocation  The IP address range that can be used to SSH to the EC2 instances

SubnetId  Subnet to use

VpcId  VPC to use

Click "Next" to advance to the tagging screen. It's not required to tag your stack, but in shared/organizational accounts tags help document who owns a resource.

For fun, let's tag the resource as a member of the testing environment. You won't be using this tag, so if you don't want to add it that's ok.

### Options

### Tags

You can specify tags (key-value pairs) for resources in your stack. You can add up to 10 unique key-value pairs for each stack. [Learn more.](#)

	Key (127 characters maximum)	Value (255 characters maximum)	
1	environment	test	+

### Advanced

You can set additional options for your stack, like notification options and a stack policy. [Learn more.](#)

Cancel Previous Next

Hit "Next" and you'll be taken to the last piece of the stack creation process, the review screen. You should be able to see all the parameters you selected (except passwords). Scroll to the bottom and click "Create" to get your stack started. You'll be taken back to the CloudFormation dashboard where we started the stack.

## STEP 3: Check Stack Status

Now that your stack has been created, we're going to watch it finish building your WordPress blog. Click on your new stack to see more information about it.

AWS Services			
Create Stack	Update Stack	Delete Stack	
Filter: Active	By Name:		
Stack Name	Created Time	Status	Description
<input checked="" type="checkbox"/> WordpressTestStack	2015-02-25 17:33:48 UTC-0500	CREATE_IN_PROGRESS	AWS CloudFormation Sample Template

The first thing you should notice is in the stack's status column. All CloudFormation stacks and resources have a status that reflects the last action done to them (CREATE in this case) and the status of that action (IN\_PROGRESS). As a user, these statuses are helpful for at-a-glance information about the condition of your deployment.

Down in the "Events" tab of the stack information, we can see which resources CloudFormation is working on. The stack itself contains all the other resources, so it needed to be created first. Next, the security group for our WordPress instance needed to be created, and now we'll wait a little bit for our actual instance to finish. This should take no more than 10 minutes.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy
2015-02-25		Status	Type	Logical ID	Status Reason		
▶ 17:34:18 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::Instance	WebServer	Resource creation Initiated			
▶ 17:34:16 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::Instance	WebServer				
▶ 17:34:14 UTC-0500	CREATE_COMPLETE	AWS::EC2::SecurityGroup	WebServerSecurityGroup	Resource creation Initiated			
▶ 17:34:12 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::SecurityGroup	WebServerSecurityGroup				
▶ 17:33:55 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::SecurityGroup	WebServerSecurityGroup	User Initiated			
▶ 17:33:48 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	WordpressTestStack				

Newer events are at the top of the list, and you can see how CloudFormation automatically handles dependencies between the resources in the stack. The EC2 instance can't be created without a security group, so CloudFormation waits for the security group to finish before moving on.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy
2015-02-25		Status	Type	Logical ID	Status Reason		
▶ 17:36:37 UTC-0500	CREATE_COMPLETE	AWS::CloudFormation::Stack	WordpressTestStack	Received SUCCESS signal with UniqueId: i-da670720			
▶ 17:36:35 UTC-0500	CREATE_COMPLETE	AWS::EC2::Instance	WebServer				
▶ 17:36:33 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::Instance	WebServer	Resource creation Initiated			
▶ 17:34:18 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::Instance	WebServer				
▶ 17:34:16 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::Instance	WebServer	Resource creation Initiated			
▶ 17:34:14 UTC-0500	CREATE_COMPLETE	AWS::EC2::SecurityGroup	WebServerSecurityGroup				
▶ 17:34:12 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::SecurityGroup	WebServerSecurityGroup	User Initiated			
▶ 17:33:55 UTC-0500	CREATE_IN_PROGRESS	AWS::EC2::SecurityGroup	WebServerSecurityGroup				
▶ 17:33:48 UTC-0500	CREATE_IN_PROGRESS	AWS::CloudFormation::Stack	WordpressTestStack				

After waiting a few minutes, we can see that the instance was created and then CloudFormation set the stack's status to CREATE\_COMPLETE.

Now let's see if WordPress works for us.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy
Key		Value					
WebsiteURL		http://ec2-54-83-104-76.compute-1.amazonaws.com/wordpress					

Go to the "Outputs" tab for the URL of the WordPress instance. CloudFormation used the EC2 instance's DNS name to build the URL you see here. From the console, click the link for your new WordPress installation. You can log in using the username and password credentials you selected when creating the stack.

#### STEP 4: View Stack Resources

You've successfully created a CloudFormation stack, now let's take a look at what's inside. From the CloudFormation dashboard click on your stack, then go to the "Resources" panel.



Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy
Logical ID	Physical ID	Type	Status				
WebServerSecurityGroup	WordPressTestStack-WebServerSecurityGroup-81FG56KSA7H-W	AWS::EC2::SecurityGroup	CREATE_COMPLETE				
WebServer	da670720	AWS::EC2::Instance	CREATE_IN_PROGRESS				

CloudFormation is easy to integrate into other workflows because all it does is orchestrate other AWS products and make it easy to repeat patterns. In your stack, the resources are the security group for the EC2 instance and the EC2 instance itself. You can see that the resource has a name in the template (WebServer) that maps to a "physical ID" (i-da670720) which is the real resource. Here is an excerpt of the template to give you an idea of what resources look like in that form. The full template is also available for you to review.

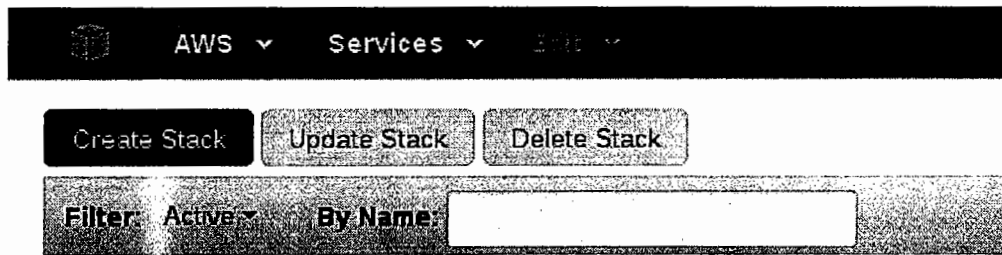
```
...
"Resources" : {
  "WebServerSecurityGroup" : {
    "Type" : "AWS::EC2::SecurityGroup",
    "Properties" : {
      "GroupDescription" : "Enable HTTP access via port 80 locked down to the load balancer + S
SH access",
      "SecurityGroupIngress" : [
        { "IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0" },
        { "IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22", "CidrIp" : { "Ref" : "SSHLocation" } }
      ]
    }
  },
  ...
}
```

You can see that we actually didn't specify a name for the security group itself because CloudFormation generates one automatically. This is common for CloudFormation resources like IAM users, security groups, and DynamoDB tables if their name is left unspecified.

Now that you understand the resources underlying your WordPress installation, it's time to delete your stack. This will also destroy all its resources.

## STEP 5: Delete the Stack

Now that you've tested CloudFormation's built-in WordPress template and made sure your installation works, it's time to clean up the resources that CloudFormation created for you. Click on your stack and then hit the **"Delete Stack"** button at the top of the console.



Now go to the "Events" tab for your stack to see what is happening to your resources.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy
2015-02-25		Status	Type	Logical ID		Status Reason	
▶ 17:38:49 UTC-0500		DELETE_IN_PROGRESS	AWS::EC2::Instance	WebServer			
▶ 17:38:43 UTC-0500		DELETE_IN_PROGRESS	AWS::CloudFormation::Stack	WordpressTestStack		User Initiated	
▶ 17:36:37 UTC-0500		CREATE_COMPLETE	AWS::CloudFormation::Stack	WordpressTestStack			

CloudFormation begins by deleting its resources and transitioning the stack to the DELETE\_IN\_PROGRESS state. Since there isn't a resource that depends on the EC2 instance, that is deleted first.

After a few minutes, all the resources will be deleted and your Events console will look like this.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy
2015-02-25		Status	Type	Logical ID		Status Reason	
▶ 17:40:07 UTC-0500		DELETE_COMPLETE	AWS::CloudFormation::Stack	WordpressTestStack			
▶ 17:40:06 UTC-0500		DELETE_COMPLETE	AWS::EC2::SecurityGroup	WebServerSecurityGroup			
▶ 17:40:05 UTC-0500		DELETE_IN_PROGRESS	AWS::EC2::SecurityGroup	WebServerSecurityGroup			
▶ 17:40:03 UTC-0500		DELETE_COMPLETE	AWS::EC2::Instance	WebServer			
▶ 17:38:49 UTC-0500		DELETE_IN_PROGRESS	AWS::EC2::Instance	WebServer			
▶ 17:38:43 UTC-0500		DELETE_IN_PROGRESS	AWS::CloudFormation::Stack	WordpressTestStack		User Initiated	
▶ 17:36:37 UTC-0500		CREATE_COMPLETE	AWS::CloudFormation::Stack	WordpressTestStack			

You can see that the order in which resources are deleted is the exact reverse of the order in which they were created. In a stack with only one "dependency chain" like this one, things always get deleted in the same order. In larger stacks, CloudFormation can delete resources in any order as long as all their dependents are deleted first. Since the security group was in use by the EC2 instance, it could not be deleted until **after** the instance.

Congratulations! You've successfully created, tested, and deleted your first stack.