Develop a CloudFormation Template

The development team at your company creates custom themes and plugins for Wordpress websites. Each theme or plugin they develop needs to be tested and reviewed on a live Wordpress website. Their current practice is to run a virtual machine on their desktops, but using that method prevents them from sharing their work with other developers. To allow sharing, they've started setting up servers in AWS, but this is causing chaos; dozens of developers are creating resources in the company AWS account without following guidelines causing naming conflicts and abandoned resources. In addition, the servers are using different operating systems and versions of software, causing inconsistencies in the test environments.

You've been asked to create a CloudFormation template that will allow the development team to build Wordpress environments quickly, easily, and in a consistent manner.

1. Open a text or code editor on your local system and create a plain-text file named wordpress_cloudformation_template.json. This document will hold the CloudFormation template in JSON format. While creating the document, pay special attention to punctuation including commas, colons, quotes, and brackets.
2. You'll begin the document by entering a left bracket, the template format version, and the template description. Paste the following code into the document:

```
{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "CloudFormation Template for a Stand Alone Wordpress Site",
```

Next, you'll continue the document by adding the parameters. Particularly, this template will need parameters for:

- An SSH key that developers can user to connect to their server.
- The type of instance they need to run, including a default selection of "t2.small."
- A name, password, username, and root password for the MySQL database that will be installed on the server. Defaults need to be included for these as well.

3. Paste the following code into the document, on the line immediately following the previous entry:

```
"Parameters" : {

"KeyName": {

"Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instances",
```

```
"Type": "AWS::EC2::KeyPair::KeyName",

"ConstraintDescription" : "must be the name of an existing EC2 KeyPair."

},

"InstanceType" : {

"Description" : "WebServer EC2 instance type",

"Type" : "String",

"Default" : "t2.small",

"AllowedValues" : [ "t2.small" ],

"ConstraintDescription" : "must be a valid EC2 instance type."

},

"DBName" : {

"Default": "wordpressdb",

"Description" : "The WordPress database name",

"Type": "String",

"MinLength": "1",

"MaxLength": "64",

"AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",

"ConstraintDescription" : "must begin with a letter and contain only alphanumeric characters."

},

"DBUser" : {

"Default": "wordpress",

"Description" : "The WordPress database admin account username",

"Type": "String",

"MinLength": "1",

"MaxLength": "16",
```

```
"AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",

"ConstraintDescription" : "must begin with a letter and contain only alphanumeric characters."

},

"DBPassword" : {

"Default": "wordpress",

"Description" : "The WordPress database admin account password",

"Type": "String",

"MinLength": "8",

"MaxLength": "41",

"AllowedPattern" : "[a-zA-Z0-9]*",

"ConstraintDescription" : "must contain only alphanumeric characters."

},

"DBRootPassword" : {

"Default": "wordpress",

"Description" : "MySQL root password",

"Type": "String",

"MinLength": "8",

"MaxLength": "41",

"AllowedPattern" : "[a-zA-Z0-9]*",

"ConstraintDescription" : "must contain only alphanumeric characters."

}

},
```

With the parameters in place, you'll now add a mappings section that will allow the template to dynamically select the correct AMI when creating an EC2 instance. The mapping should be based on the instance type and the region where the stack is being created.

4. Paste the following code into your document, again after the previously entered code:

```
"Mappings" : {

"AWSInstanceType2Arch" : {

"t2.small" : { "Arch" : "HVM64" }

},

"AWSRegionArch2AMI" : {

"us-east-1" : {"HVM64" : "ami-032930428bf1abbff"},

"us-east-2" : {"HVM64" : "ami-027cab9a7bf0155df"},

"us-west-1" : {"HVM64" : "ami-088c153f74339f34c"},

"us-west-2" : {"HVM64" : "ami-01fee56b22f308154"}

}

},
```

After the mappings section you'll next add a section for resources. Your template needs to include resources for an EC2 Instance and Security Group. The Security Group needs to allow SSH access on port 22 and HTTP access on port 80.

5. Paste the following code into your document:

```
"Resources" : {

"WebServerSecurityGroup" : {

"Type" : "AWS::EC2::SecurityGroup",

"Properties" : {

"GroupDescription" : "Enable HTTP access via port 80, SSH access via port 22",

"SecurityGroupIngress" : [

{"IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0"},

{"IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22", "CidrIp" : "0.0.0.0/0"}

]
```

```
}

},
```

Next is a resource for an EC2 Instance. This resource will configure all aspects of the server, including:

- Setting instance properties like SSH key and security group.
- Installing PHP, Apache, and MySQL.
- Creating a database for the Wordpress site using the parameters passed into the template.
- Installing Wordpress and configuring it to use the database.

6. Paste the following code into your document:

```
"WebServer": {

"Type" : "AWS::EC2::Instance",

"Metadata" : {

"AWS::CloudFormation::Init" : {

"configSets" : {

"wordpress_install" : ["install_cfn", "install_wordpress", "configure_wordpress" ]

},

"install_cfn" : {

"files": {

"/etc/cfn/cfn-hup.conf": {

"content": { "Fn::Join": [ "", [

"[main]\n",

"stack=", { "Ref": "AWS::StackId" }, "\n",

"region=", { "Ref": "AWS::Region" }, "\n"

]]},

"mode" : "000400",

"owner" : "root",

"group" : "root"
```

```
},

"/etc/cfn/hooks.d/cfn-auto-reloader.conf": {

"content": { "Fn::Join": [ "", [

"[cfn-auto-reloader-hook]\n",

"triggers=post.update\n",

"path=Resources.WebServer.Metadata.AWS::CloudFormation::Init\n",

"action=/opt/aws/bin/cfn-init -v ",

" --stack ", { "Ref" : "AWS::StackName" },

" --resource WebServer ",

" --configsets wordpress_install ",

" --region ", { "Ref" : "AWS::Region" }, "\n"

]]},

"mode" : "000400",

"owner" : "root",

"group" : "root"

}

},

"services" : {

"sysvinit" : {

"cfn-hup" : { "enabled" : "true", "ensureRunning" : "true",

"files" : ["/etc/cfn/cfn-hup.conf", "/etc/cfn/hooks.d/cfn-auto-reloader.conf"] }

}

}

},

"install_wordpress" : {
```

```
"packages" : {

"yum" : {

"php73" : [],

"php73-mysqlnd" : [],

"mysql57" : [],

"mysql57-server" : [],

"mysql57-devel" : [],

"mysql57-libs" : [],

"httpd24" : []

}

},

"sources" : {

"/var/www/html" : "http://wordpress.org/latest.tar.gz"

},

"files" : {

"/tmp/setup.mysql" : {

"content" : { "Fn::Join" : ["", [

"CREATE DATABASE ", { "Ref" : "DBName" }, ";\n",

"CREATE USER '", { "Ref" : "DBUser" }, "'@'localhost' IDENTIFIED BY '", { "Ref" : "DBPassword" },
"';\n",

"GRANT ALL ON ", { "Ref" : "DBName" }, ".* TO '", { "Ref" : "DBUser" }, "'@'localhost';\n",

"FLUSH PRIVILEGES;\n"

]]},

"mode" : "000400",

"owner" : "root",

"group" : "root"
```

```
},

"/tmp/create-wp-config" : {

"content" : { "Fn::Join" : [ "", [

"#!/bin/bash -xe\n",

"cp /var/www/html/wordpress/wp-config-sample.php /var/www/html/wordpress/wp-config.php\n",

"sed -i \"s/'database_name_here'/'",{ "Ref" : "DBName" }, "'/g\" wp-config.php\n",

"sed -i \"s/'username_here'/'",{ "Ref" : "DBUser" }, "'/g\" wp-config.php\n",

"sed -i \"s/'password_here'/'",{ "Ref" : "DBPassword" }, "'/g\" wp-config.php\n"

]]},

"mode" : "000500",

"owner" : "root",

"group" : "root"

}

},

"services" : {

"sysvinit" : {

"httpd" : { "enabled" : "true", "ensureRunning" : "true" },

"mysqld" : { "enabled" : "true", "ensureRunning" : "true" }

}

}

},

"configure_wordpress" : {

"commands" : {

"01_set_mysql_root_password" : {

"command" : { "Fn::Join" : ["", ["mysqladmin -u root password '", { "Ref" : "DBRootPassword" }, "'"]]},
```

```
"test" : { "Fn::Join" : ["", ["$(mysql ", { "Ref" : "DBName" }, " -u root --password='", { "Ref" :
"DBRootPassword" }, "' >/dev/null 2>&1 </dev/null); (( $? != 0 ))"]]}

},

"02_create_database" : {

"command" : { "Fn::Join" : ["", ["mysql -u root --password='", { "Ref" : "DBRootPassword" }, "' <
/tmp/setup.mysql"]]},

"test" : { "Fn::Join" : ["", ["$(mysql ", { "Ref" : "DBName" }, " -u root --password='", { "Ref" :
"DBRootPassword" }, "' >/dev/null 2>&1 </dev/null); (( $? != 0 ))"]]}

},

"03_configure_wordpress" : {

"command" : "/tmp/create-wp-config",

"cwd" : "/var/www/html/wordpress"

}

}

}

}

},

"Properties": {

"ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region" },

{ "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "InstanceType" }, "Arch" ] } ] },

"InstanceType" : { "Ref" : "InstanceType" },

"SecurityGroups" : [ {"Ref" : "WebServerSecurityGroup"} ],

"KeyName" : { "Ref" : "KeyName" },

"UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [

"#!/bin/bash -xe\n",

"yum update -y aws-cfn-bootstrap\n",

"/opt/aws/bin/cfn-init -v ",
```

```
" --stack ", { "Ref" : "AWS::StackName" },

" --resource WebServer ",

" --configsets wordpress_install ",

" --region ", { "Ref" : "AWS::Region" }, "\n",

"/opt/aws/bin/cfn-signal -e $? ",

" --stack ", { "Ref" : "AWS::StackName" },

" --resource WebServer ",

" --region ", { "Ref" : "AWS::Region" }, "\n"

]]}}

},

"CreationPolicy" : {

"ResourceSignal" : {

"Timeout" : "PT15M"

}

}

}

},
```

The final section of the template is used to generate outputs using references to the resources that were created. You'll use the PublicDNS property of the WebServer resource to add an output that prints a URL for the Wordpress website.

7. Paste the following code into the document. Note that this code includes right-side curly brackets that close the Resource section as well as the left-side curly bracket that opened the JSON document:

```
"Outputs" : {

"WebsiteURL" : {

"Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" : [ "WebServer", "PublicDnsName" ]},
"/wordpress" ]]},
```

```
"Description" : "WordPress Website"

}

}

}
```

8. Save your document.

You now have a JSON document containing a CloudFormation template. Move on to the next challenge where you will create a stack in AWS and test the template.

Use the CloudFormation Template to Create a Wordpress Stack

You've drafted a template to create the resources needed to create a stand-alone Wordpress site. Before handing it off to the development team, you need to test it by creating a stack with the CloudFormation interface in the AWS console.

1. If you haven't yet logged in to the AWS console, click on the **Open AWS console** button to the right of these instructions, then use the credentials provided to log in to AWS.

   Note: The **Open AWS console** button will automatically place you in the **US West (Oregon)** region after logging in; if you did not use this button to log in to the AWS console, ensure that you are in the **US West (Oregon)** region by using the region selector in the top right corner of the console.

2. Use the **Services** dropdown at the top of the page to navigate to the **CloudFormation** service, then click the button labeled **Create stack**.
3. In the **Prerequisite** section, select **Template is ready**. In the **Specify template** section, select **Upload a template file**. Click the button labeled **Chose file**, then in the file selector that opens, choose your local copy of the

wordpress_cloudformation_template.json file you created in the previous challenge. Select the file and click **Open**.
4. Click **Next**.
5. Under the section labeled **Stack name**, enter a name for your stack. Note that the name is restricted to letters (A-Z and a-z), numbers (0-9), and dashes (-).
6. Under the **Parameters** section, take note of the default entries for **DBName**, **DBPassword**, **DBRootPassword**, **DBUser**, and **InstanceType**. For the **KeyName** parameter, use the dropdown to select **default-cloudformation-ssh-key**.
7. Click **Next**.
8. On the **Configure stack options** screen under the **Tags** section, add a tag using Name for the **Key** and demo for the **Value**. Leave all other options as-is and click **Next**.
9. On the **Review** screen, review your selections, then click **Create stack**.
10. While the stack is being created, monitor the **Events** tab using the refresh button.

The CloudFormation console will indicate **CREATE_COMPLETE** in green letters once the resources for the stack have been successfully created. If your stack creation fails, check your JSON template for errors and try the stack creation process again.

Click the **Resources** tab and confirm that two resources have been created: an EC2 instance and a Security Group.

Click the Outputs tab. Confirm that a URL labeled “WebsiteURL” is present. You’ll use that URL in the next challenge to complete the Wordpress installation.

Install Wordpress

The CloudFormation stack generates a URL for the Wordpress site as output. To complete your testing, you need to open the URL and enter the values needed to complete the Wordpress installation.

1. View the **Outputs** tab of the CloudFormation stack you just created, and open the **WebsiteURL** in a new tab.
2. In that new tab, under the **Information needed** section, enter the following for the specified fields:
    - Site Title: Demo Site
    - Username: demo
    - Password: developerdemosite!!
    - Your Email: demo@example.com
    - Search engine visibility: selected
3. Click the button labeled **Install Wordpress**.
4. On the **Success!** screen, click the button labeled **Log in**. Enter the username and password for the demo user and click the button labeled **Log in**. This opens the administrative dashboard for the site.

The Wordpress admin dashboard is the interface that the developers will use to install their themes and plugins. By completing the Wordpress installation, you've confirmed that the CloudFormation template is complete and ready for use by the development team. Now that you are ready to share the template with them, you need to delete the demo stack you created. You'll

do that in the next challenge.

Delete the CloudFormation Stack

Now that your CloudFormation template has been developed and tested, you can remove the stack and its resources.

1. Go back to the CloudFormation tab and open the **Events** tab of your stack.
2. Click the button labeled **Delete**.
3. Confirm that you are deleting the stack by clicking the button labeled **Delete stack**.
4. While the stack is being deleted, monitor the **Events** tab by occasionally clicking the refresh button to update the screen.
5. Note the **DELETE_IN_PROGRESS** and **DELETE_COMPLETE** status messages for each of the stack's resources.

Once the stack has been deleted, you can view it by going to the **Stacks** page of the CloudFormation console and filtering on **Deleted**. Click the stack's name to view the stack and any of its details.

Congratulations! You have authored a CloudFormation template and used it to deploy a stack. By using this template, the development team will have a consistent deployment method for their Wordpress sites and an easy way to manage all of the site's resources.

Develop a CloudFormation Template

The development team at your company creates custom themes and plugins for Wordpress websites. Each theme or plugin they develop needs to be tested and reviewed on a live Wordpress website. Their current practice is to run a virtual machine on their desktops, but using that method prevents them from sharing their work with other developers. To allow sharing, they've started setting up servers in AWS, but this is causing chaos; dozens of developers are creating resources in the company AWS account without following guidelines causing naming conflicts and abandoned resources. In addition, the servers are using different operating systems and versions of software, causing inconsistencies in the test environments.

You've been asked to create a CloudFormation template that will allow the development team to build Wordpress environments quickly, easily, and in a consistent manner.

1. Open a text or code editor on your local system and create a plain-text file named `wordpress_cloudformation_template.json`. This document will hold the CloudFormation template in JSON format. While creating the document, pay special attention to punctuation including commas, colons, quotes, and brackets.

2. You'll begin the document by entering a left bracket, the template format version, and the template description. Paste the following code into the document:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
 "Description" : "CloudFormation Template for a Stand Alone Wordpress Site",
```

Next, you'll continue the document by adding the parameters. Particularly, this template will need parameters for:

- An SSH key that developers can user to connect to their server.
- The type of instance they need to run, including a default selection of "t2.small."
- A name, password, username, and root password for the MySQL database that will be installed on the server. Defaults need to be included for these as well.

3. Paste the following code into the document, on the line immediately following the previous entry:

```
  "Parameters" : {

    "KeyName": {

      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instances",

      "Type": "AWS::EC2::KeyPair::KeyName",

      "ConstraintDescription" : "must be the name of an existing EC2 KeyPair."

    },

    "InstanceType" : {

      "Description" : "WebServer EC2 instance type",

      "Type" : "String",

      "Default" : "t2.small",

      "AllowedValues" : [ "t2.small" ],

      "ConstraintDescription" : "must be a valid EC2 instance type."

    },

    "DBName" : {
```

```
    "DBName" : {
      "Default": "wordpressdb",
      "Description" : "The WordPress database name",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "64",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",
      "ConstraintDescription" : "must begin with a letter and contain
only alphanumeric characters."
    },
    "DBUser" : {
      "Default": "wordpress",
      "Description" : "The WordPress database admin account username",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",
```

```
        "ConstraintDescription" : "must begin with a letter and contain
only alphanumeric characters."
      },
      "DBPassword" : {
        "Default": "wordpress",
        "Description" : "The WordPress database admin account password",
        "Type": "String",
        "MinLength": "8",
        "MaxLength": "41",
        "AllowedPattern" : "[a-zA-Z0-9]*",
        "ConstraintDescription" : "must contain only alphanumeric
characters."
      },
      "DBRootPassword" : {
        "Default": "wordpress",
        "Description" : "MySQL root password",
        "Type": "String",
        "MinLength": "8"
```

```
      "Description" : "MySQL root password",

      "Type": "String",

      "MinLength": "8",

      "MaxLength": "41",

      "AllowedPattern" : "[a-zA-Z0-9]*",

      "ConstraintDescription" : "must contain only alphanumeric
characters."

    }

  },
```

With the parameters in place, you'll now add a mappings section that will allow the template to dynamically select the correct AMI when creating an EC2 instance. The mapping should be based on the instance type and the region where the stack is being created.

4. Paste the following code into your document, again after the previously entered code:

```
  "Mappings" : {

    "AWSInstanceType2Arch" : {

      "t2.small" : { "Arch" : "HVM64"  }

    },
```

```
        "AWSRegionArch2AMI" : {

          "us-east-1" : {"HVM64" : "ami-032930428bf1abbff"},

          "us-east-2" : {"HVM64" : "ami-027cab9a7bf0155df"},

          "us-west-1" : {"HVM64" : "ami-088c153f74339f34c"},

          "us-west-2" : {"HVM64" : "ami-01fee56b22f308154"}

        }

    },
```

After the mappings section you'll next add a section for resources. Your template needs to include resources for an EC2 Instance and Security Group. The Security Group needs to allow SSH access on port 22 and HTTP access on port 80.

5. Paste the following code into your document:

```
      "Resources" : {

"WebServerSecurityGroup" : {

      "Type" : "AWS::EC2::SecurityGroup",

      "Properties" : {

        "GroupDescription" : "Enable HTTP access via port 80, SSH
access via port 22",

        "SecurityGroupIngress" : [
```

```
        "SecurityGroupIngress" : [

          {"IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80",
"CidrIp" : "0.0.0.0/0"},

          {"IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22",
"CidrIp" : "0.0.0.0/0"}

        ]

      }

    },
```

Next is a resource for an EC2 Instance. This resource will configure all aspects of the server, including:

- Setting instance properties like SSH key and security group.
- Installing PHP, Apache, and MySQL.
- Creating a database for the Wordpress site using the parameters passed into the template.
- Installing Wordpress and configuring it to use the database.

6. Paste the following code into your document:

```
    "WebServer": {

      "Type" : "AWS::EC2::Instance",
```

```
      "Metadata" : {

        "AWS::CloudFormation::Init" : {

          "configSets" : {

            "wordpress_install" : ["install_cfn",  "install_wordpress",
"configure_wordpress" ]

          },

          "install_cfn" : {

            "files": {

              "/etc/cfn/cfn-hup.conf": {

                "content": { "Fn::Join": [ "", [

                  "[main]\n",

                  "stack=", { "Ref": "AWS::StackId" }, "\n",

                  "region=", { "Ref": "AWS::Region" }, "\n"

                ]]},

                "mode"  : "000400",

                "owner" : "root",

                "group" : "root"
```

```
            },
            "/etc/cfn/hooks.d/cfn-auto-reloader.conf": {
              "content": { "Fn::Join": [ "", [
                "[cfn-auto-reloader-hook]\n",
                "triggers=post.update\n",
                "path=Resources.WebServer.Metadata.AWS::CloudFormation::Init\n",
                "action=/opt/aws/bin/cfn-init -v ",
                "         --stack ", { "Ref" : "AWS::StackName" },
                "         --resource WebServer ",
                "         --configsets wordpress_install ",
                "         --region ", { "Ref" : "AWS::Region" }, "\n"
              ]]},
              "mode"  : "000400",
              "owner" : "root",
              "group" : "root"
```

```
                "group" : "root"
            }
        },
        "services" : {
            "sysvinit" : {
                "cfn-hup" : { "enabled" : "true", "ensureRunning" :
"true",
                              "files" : ["/etc/cfn/cfn-hup.conf",
"/etc/cfn/hooks.d/cfn-auto-reloader.conf"] }
            }
        }
    },
    "install_wordpress" : {
        "packages" : {
            "yum" : {
                "php73"         : [],
                "php73-mysqlnd"  : [],
```

```
                "php73-mysqlnd"  : [],
                "mysql57"        : [],
                "mysql57-server" : [],
                "mysql57-devel"  : [],
                "mysql57-libs"   : [],
                "httpd24"        : []
              }
            },
            "sources" : {
              "/var/www/html" : "http://wordpress.org/latest.tar.gz"
            },
            "files" : {
              "/tmp/setup.mysql" : {
                "content" : { "Fn::Join" : ["", [
                  "CREATE DATABASE ", { "Ref" : "DBName" }, ";\n",
                  "CREATE USER '", { "Ref" : "DBUser" }, "'@'localhost'
IDENTIFIED BY '", { "Ref" : "DBPassword" }, "';\n",
```

```
                    "GRANT ALL ON ", { "Ref" : "DBName" }, ".* TO '", {
"Ref" : "DBUser" }, "'@'localhost';\n",

                    "FLUSH PRIVILEGES;\n"

                ]]},

                "mode"   : "000400",

                "owner" : "root",

                "group" : "root"

            },

            "/tmp/create-wp-config" : {

                "content" : { "Fn::Join" : [ "", [

                    "#!/bin/bash -xe\n",

                    "cp /var/www/html/wordpress/wp-config-sample.php
/var/www/html/wordpress/wp-config.php\n",

                    "sed -i \"s/'database_name_here'/'",{ "Ref" :
"DBName" }, "'/g\" wp-config.php\n",

                    "sed -i \"s/'username_here'/'",{ "Ref" : "DBUser" },
"'/g\" wp-config.php\n",

                    "sed -i \"s/'password_here'/'",{ "Ref" : "DBPassword"
}, "'/g\" wp-config.php\n"
```

```
                ]]},
                "mode" : "000500",
                "owner" : "root",
                "group" : "root"
              }
            },
          "services" : {
            "sysvinit" : {
              "httpd"  : { "enabled" : "true", "ensureRunning" :
"true" },
              "mysqld" : { "enabled" : "true", "ensureRunning" :
"true" }
            }
          }
        },
        "configure_wordpress" : {
          "commands" : {
            "01_set_mysql_root_password" : {
```

```
                "command" : { "Fn::Join" : ["", ["mysqladmin -u root
password '", { "Ref" : "DBRootPassword" }, "'"]]},

                "test" : { "Fn::Join" : ["", ["$(mysql ", { "Ref" :
"DBName" }, " -u root --password='", { "Ref" : "DBRootPassword" }, "'
>/dev/null 2>&1 </dev/null); (( $? != 0 ))"]]}

            },

            "02_create_database" : {

                "command" : { "Fn::Join" : ["", ["mysql -u root --
password='", { "Ref" : "DBRootPassword" }, "' < /tmp/setup.mysql"]]},

                "test" : { "Fn::Join" : ["", ["$(mysql ", { "Ref" :
"DBName" }, " -u root --password='", { "Ref" : "DBRootPassword" }, "'
>/dev/null 2>&1 </dev/null); (( $? != 0 ))"]]}

            },

            "03_configure_wordpress" : {

                "command" : "/tmp/create-wp-config",

                "cwd" : "/var/www/html/wordpress"

            }

        }

    }
```

```
            }

          }

        }

      },

      "Properties": {

        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref"
: "AWS::Region" },

                          { "Fn::FindInMap" : [ "AWSInstanceType2Arch",
{ "Ref" : "InstanceType" }, "Arch" ] } ] },

        "InstanceType"   : { "Ref" : "InstanceType" },

        "SecurityGroups" : [ {"Ref" : "WebServerSecurityGroup"} ],

        "KeyName"        : { "Ref" : "KeyName" },

        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [

                        "#!/bin/bash -xe\n",

                        "yum update -y aws-cfn-bootstrap\n",

                        "/opt/aws/bin/cfn-init -v ",

                        "         --stack ", { "Ref" : "AWS::StackName"
},
```

```
                        "          --resource WebServer ",
                        "          --configsets wordpress_install ",
                        "          --region ", { "Ref" : "AWS::Region" },
"\n",
                        "/opt/aws/bin/cfn-signal -e $? ",
                        "          --stack ", { "Ref" : "AWS::StackName"
},
                        "          --resource WebServer ",
                        "          --region ", { "Ref" : "AWS::Region" },
"\n"
        ]]}}
      },
      "CreationPolicy" : {
        "ResourceSignal" : {
          "Timeout" : "PT15M"
        }
      }
    }
```

```
    }
  },
```

The final section of the template is used to generate outputs using references to the resources that were created. You'll use the PublicDNS property of the WebServer resource to add an output that prints a URL for the Wordpress website.

7. Paste the following code into the document. Note that this code includes right-side curly brackets that close the Resource section as well as the left-side curly bracket that opened the JSON document:

```
  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" : [ "WebServer", "PublicDnsName" ]}, "/wordpress" ]]},
      "Description" : "WordPress Website"
    }
  }
}
```

8. Save your document.

You now have a JSON document containing a CloudFormation template. Move on to the next challenge where you will create a stack in AWS and test the template.

Use the CloudFormation Template to Create a Wordpress Stack

You've drafted a template to create the resources needed to create a stand-alone Wordpress site. Before handing it off to the development team, you need to test it by creating a stack with the CloudFormation interface in the AWS console.

1. If you haven't yet logged in to the AWS console, click on the **Open AWS console** button to the right of these instructions, then use the credentials provided to log in to AWS.

   Note: The **Open AWS console** button will automatically place you in the **US West (Oregon)** region after logging in; if you did not use this button to log in to the AWS console, ensure that you are in the **US West (Oregon)** region by using the region selector in the top right corner of the console.

2. Use the **Services** dropdown at the top of the page to navigate to the **CloudFormation** service, then click the button labeled **Create stack**.

3. In the **Prerequisite** section, select **Template is ready**. In the **Specify template** section, select **Upload a template file**. Click the button labeled **Chose file**, then in the file selector that opens, choose your local copy of the `wordpress_cloudformation_template.json` file you created in the previous challenge. Select the file and click **Open**.

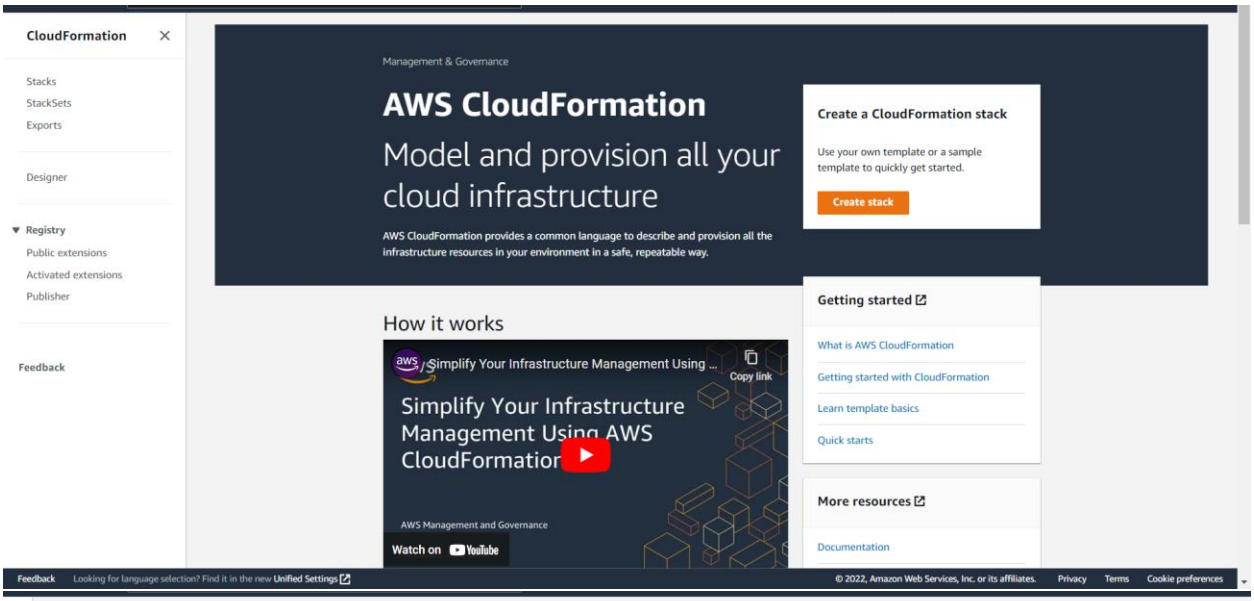
4. Click **Next**.

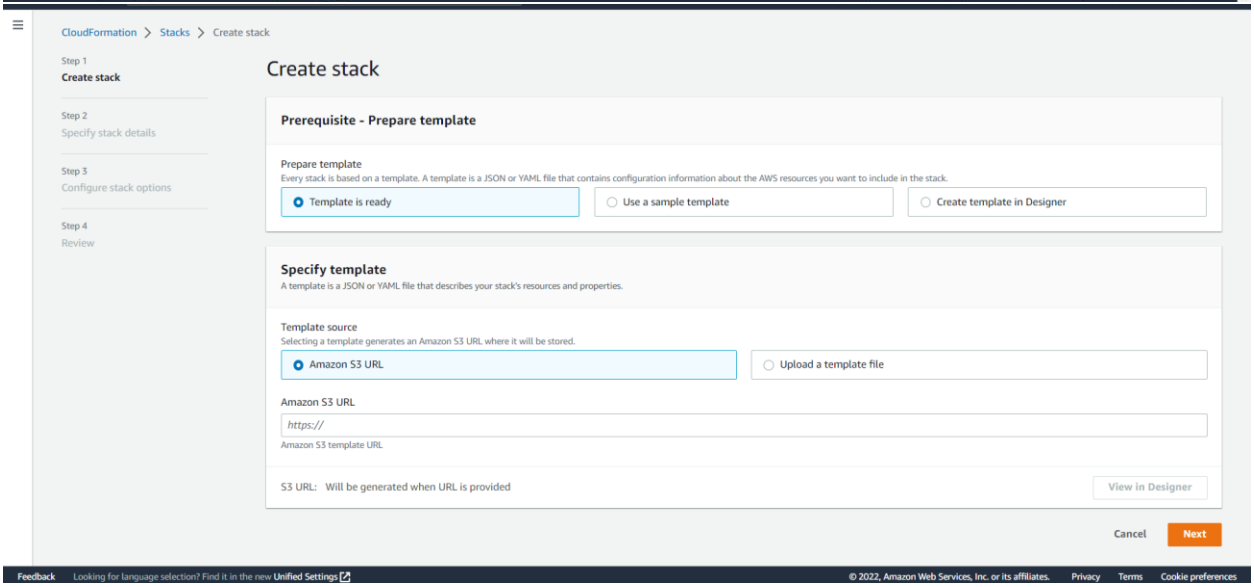
4. Click **Next**.
5. Under the section labeled **Stack name**, enter a name for your stack. Note that the name is restricted to letters (A-Z and a-z), numbers (0-9), and dashes (-).
6. Under the **Parameters** section, take note of the default entries for **DBName**, **DBPassword**, **DBRootPassword**, **DBUser**, and **InstanceType**. For the **KeyName** parameter, use the dropdown to select **default-cloudformation-ssh-key**.
7. Click **Next**.
8. On the **Configure stack options** screen under the **Tags** section, add a tag using `Name` for the **Key** and `demo` for the **Value**. Leave all other options as-is and click **Next**.
9. On the **Review** screen, review your selections, then click **Create stack**.
10. While the stack is being created, monitor the **Events** tab using the refresh button.

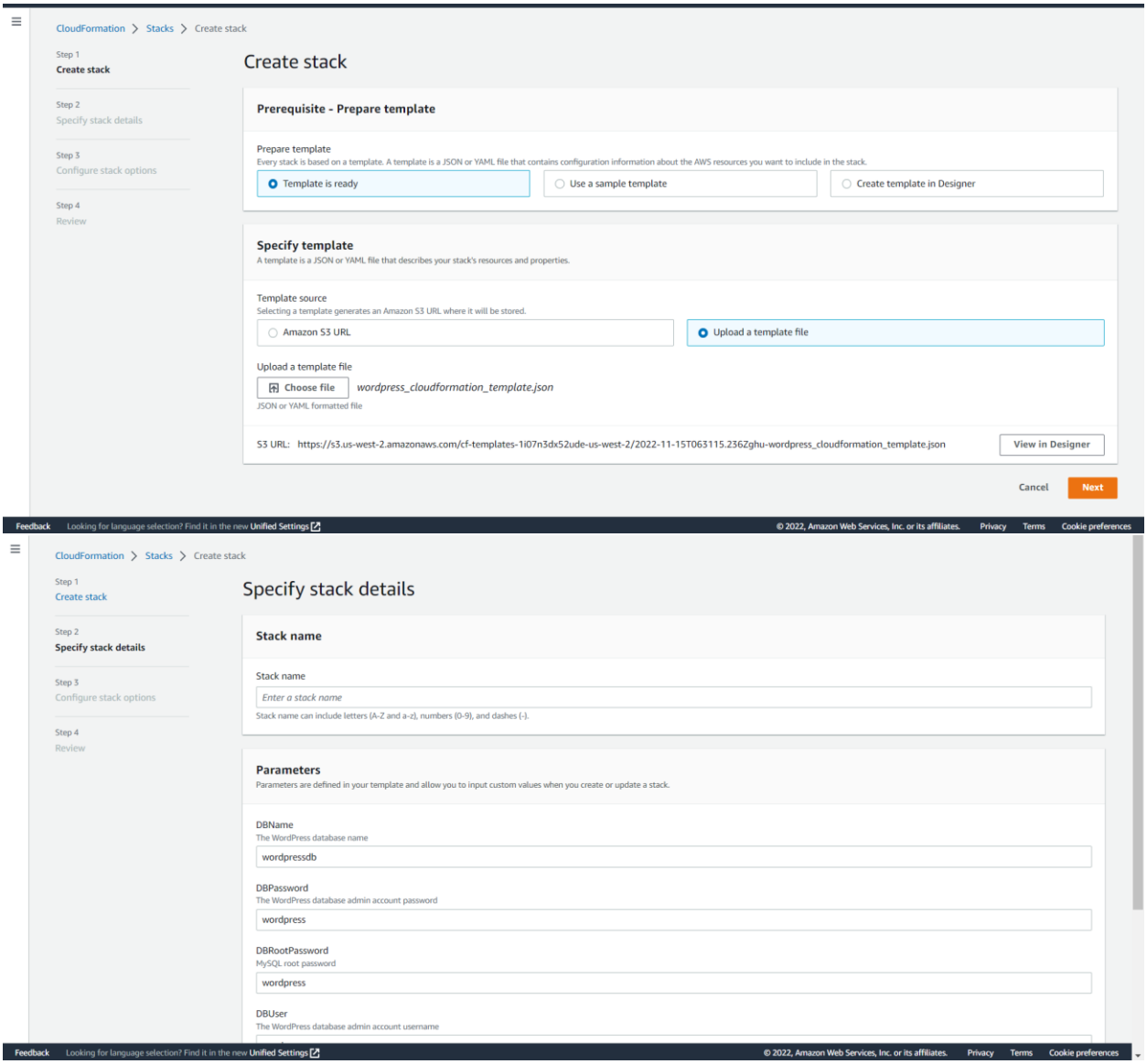
The CloudFormation console will indicate **CREATE_COMPLETE** in green letters once the resources for the stack have been successfully created. If your stack creation fails, check your JSON template for errors and try the stack creation process again.

Click the **Resources** tab and confirm that two resources have been created: an EC2 instance and a Security Group.

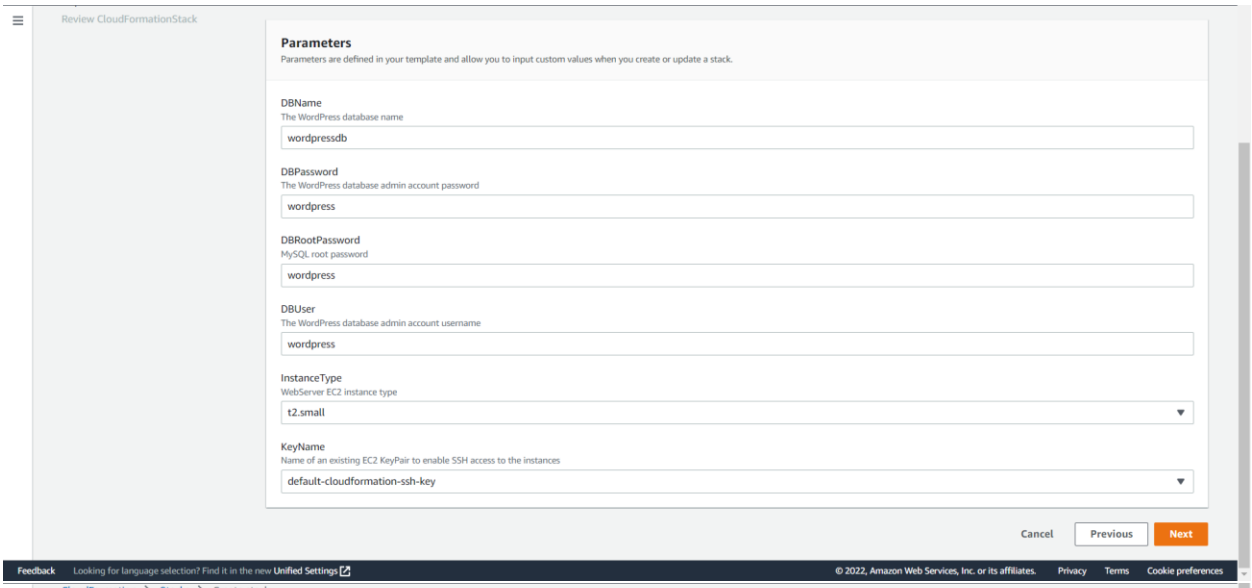
Click the **Outputs** tab. Confirm that a URL labeled "WebsiteURL" is present. You'll use that URL in the next challenge to complete the Wordpress installation.

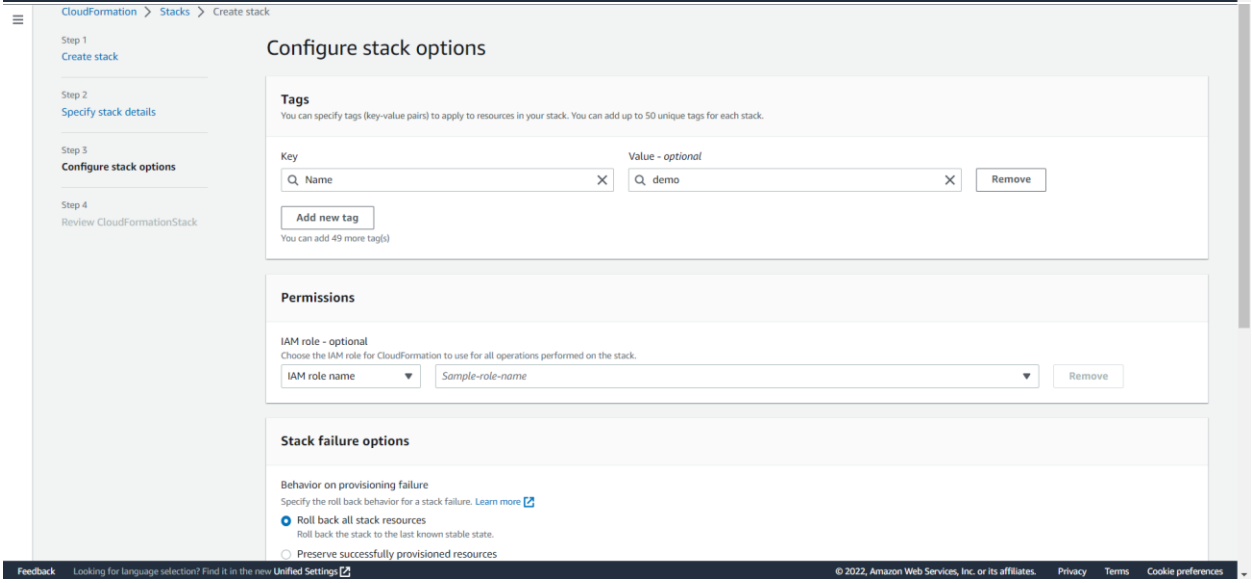

CloudFormation
Stacks
StackSets
Exports
Designer
Registry
Public extensions
Activated extensions
Publisher
Feedback
Management & Governance
AWS CloudFormation
Model and provision all your cloud infrastructure
AWS CloudFormation provides a common language to describe and provision all the infrastructure resources in your environment in a safe, repeatable way.
Create a CloudFormation stack
Use your own template or a sample template to quickly get started.
Create stack
How it works
Simplify Your Infrastructure Management Using ...
Copy link
Simplify Your Infrastructure Management Using AWS CloudFormation
AWS Management and Governance
Watch on YouTube
Getting started
What is AWS CloudFormation
Getting started with CloudFormation
Learn template basics
Quick starts
More resources
Documentation
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

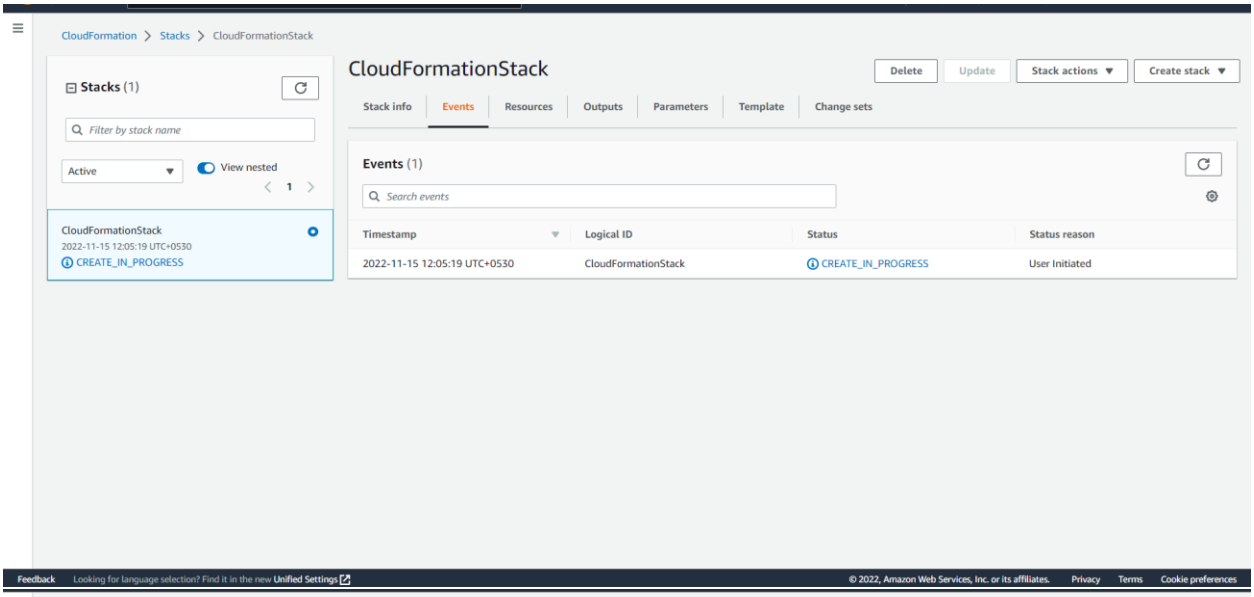

CloudFormation > Stacks > Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review
Create stack
Prerequisite - Prepare template
Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.
Template is ready
Use a sample template
Create template in Designer
Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.
Template source
Selecting a template generates an Amazon S3 URL where it will be stored.
Amazon S3 URL
Upload a template file
Amazon S3 URL
https://
Amazon S3 template URL
S3 URL: Will be generated when URL is provided
View in Designer
Cancel
Next
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

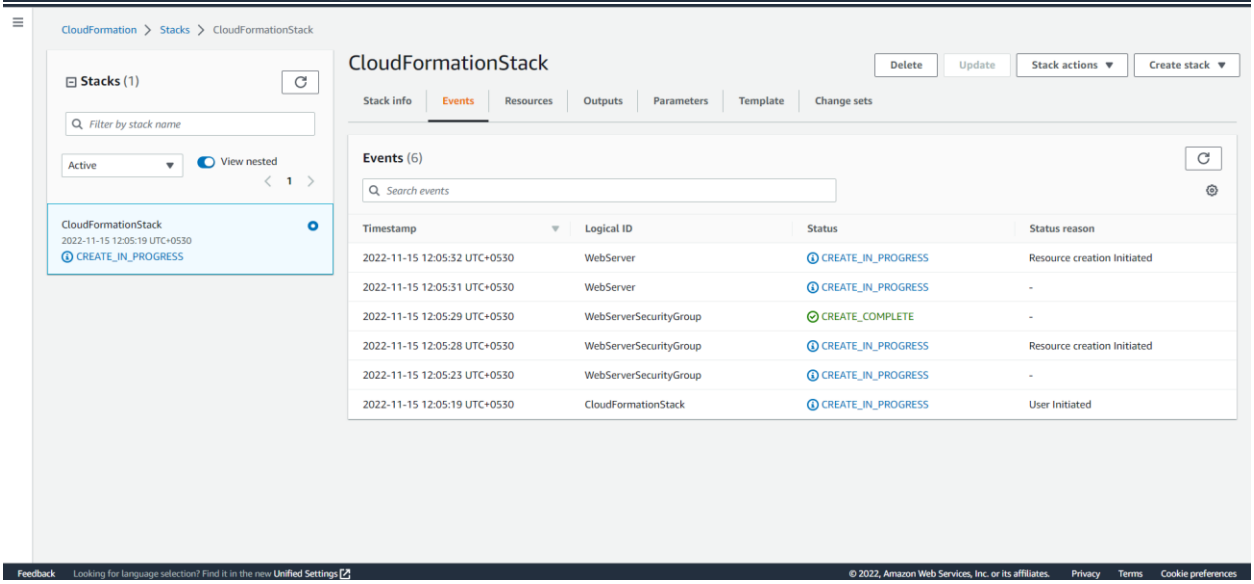

CloudFormation > Stacks > Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review
Create stack
Prerequisite - Prepare template
Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.
Template is ready
Use a sample template
Create template in Designer
Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.
Template source
Selecting a template generates an Amazon S3 URL where it will be stored.
Amazon S3 URL
Upload a template file
Upload a template file
Choose file
wordpress_cloudformation_template.json
JSON or YAML formatted file
S3 URL: https://s3.us-west-2.amazonaws.com/cf-templates-1i07n3dx52ude-us-west-2/2022-11-15T063115.236Zghu-wordpress_cloudformation_template.json
View in Designer
Cancel
Next
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

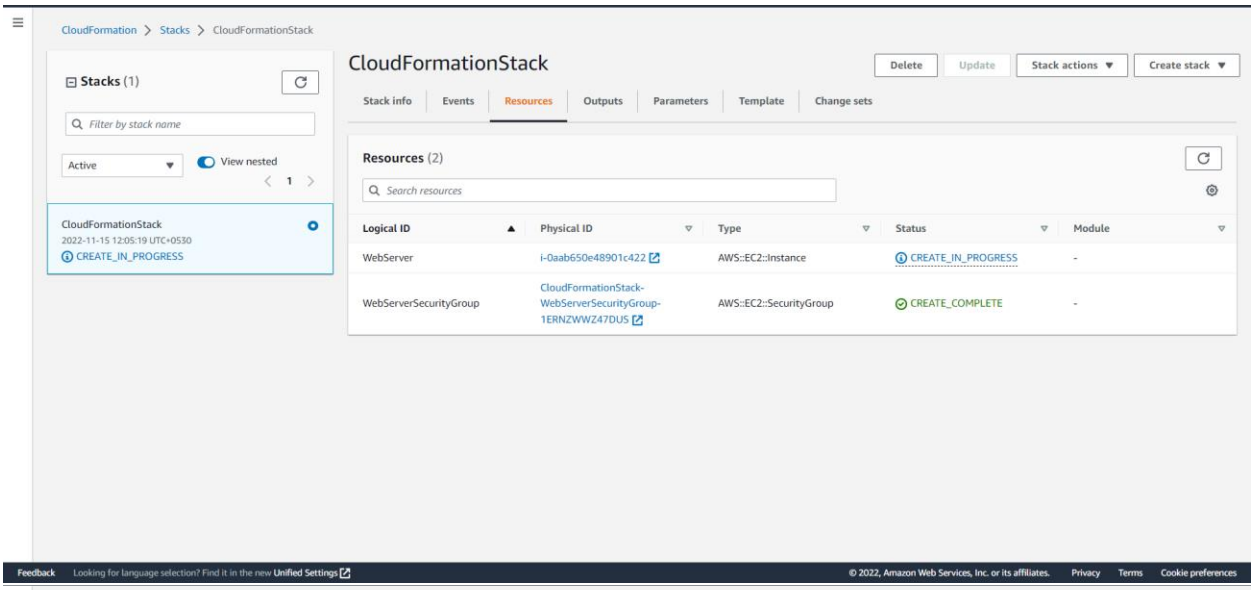
CloudFormation > Stacks > Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review
Specify stack details
Stack name
Stack name
Enter a stack name
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
DBName
The WordPress database name
wordpressdb
DBPassword
The WordPress database admin account password
wordpress
DBRootPassword
MySQL root password
wordpress
DBUser
The WordPress database admin account username
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

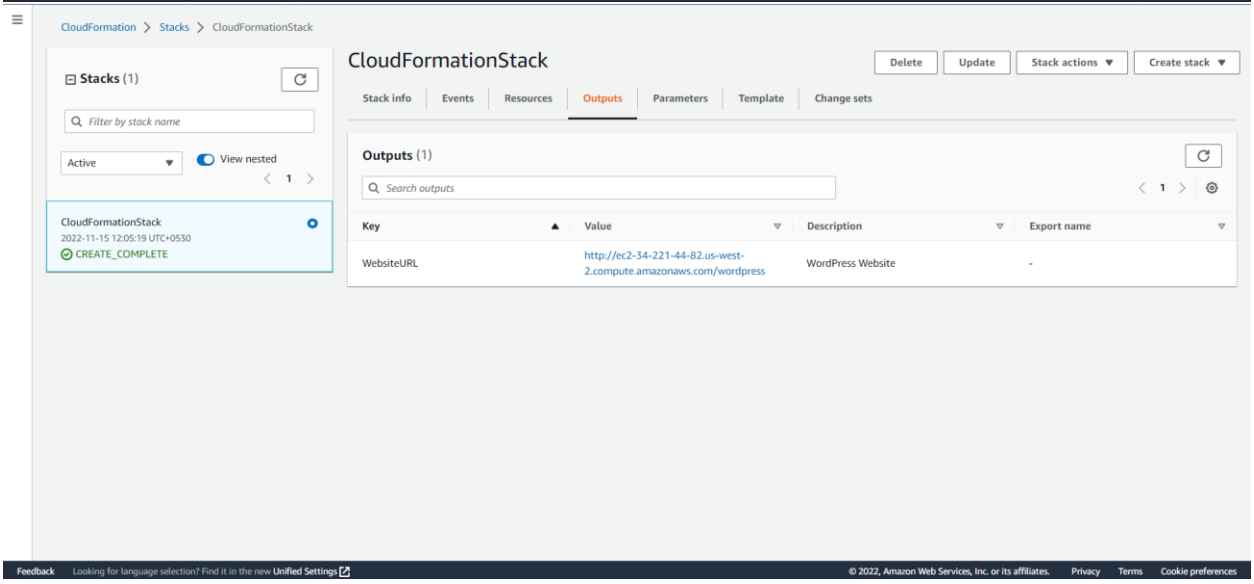

Review CloudFormationStack
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
DBName
The WordPress database name
wordpressdb
DBPassword
The WordPress database admin account password
wordpress
DBRootPassword
MySQL root password
wordpress
DBUser
The WordPress database admin account username
wordpress
InstanceType
WebServer EC2 instance type
t2.small
KeyName
Name of an existing EC2 KeyPair to enable SSH access to the instances
default-cloudformation-ssh-key
Cancel
Previous
Next
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

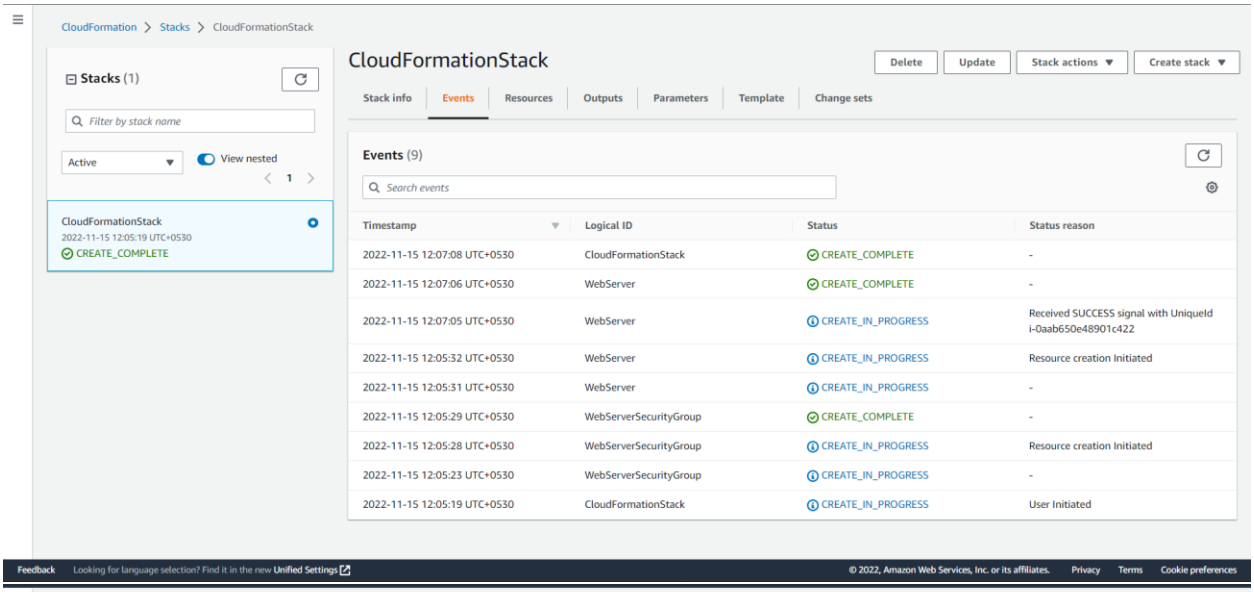

CloudFormation > Stacks > Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review CloudFormationStack
Configure stack options
Tags
You can specify tags (key-value pairs) to apply to resources in your stack. You can add up to 50 unique tags for each stack.
Key
Name
Value - optional
demo
Remove
Add new tag
You can add 49 more tag(s)
Permissions
IAM role - optional
Choose the IAM role for CloudFormation to use for all operations performed on the stack.
IAM role name
Sample-role-name
Remove
Stack failure options
Behavior on provisioning failure
Specify the roll back behavior for a stack failure. Learn more
Roll back all stack resources
Roll back the stack to the last known stable state.
Preserve successfully provisioned resources
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

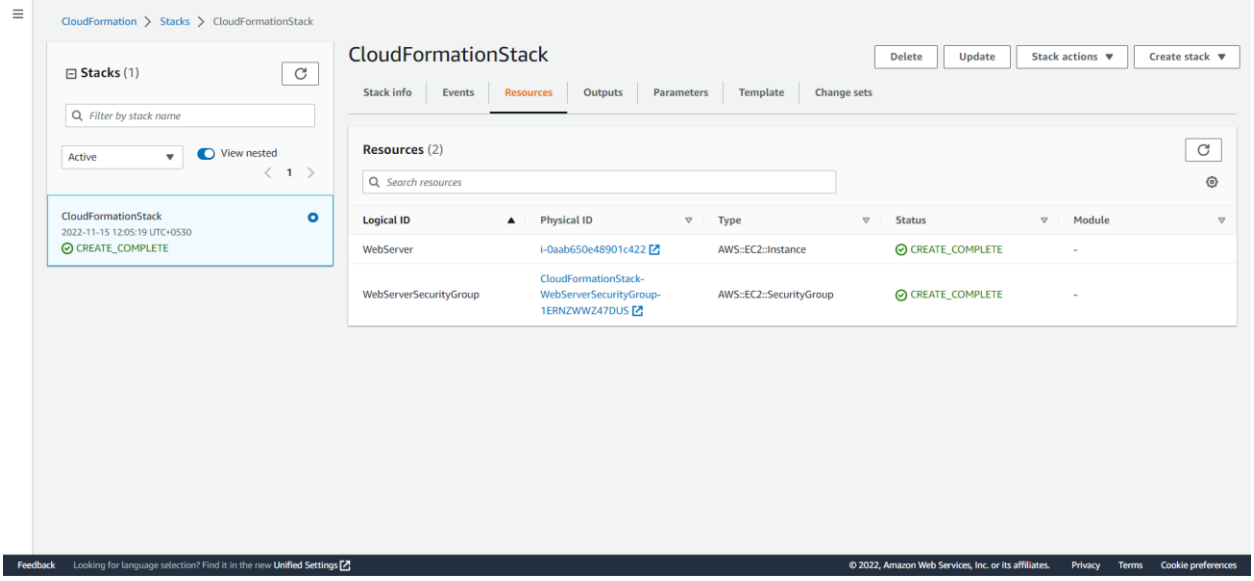

CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_IN_PROGRESS
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Events (1)
Search events
Timestamp
Logical ID
Status
Status reason
2022-11-15 12:05:19 UTC+0530
CloudFormationStack
CREATE_IN_PROGRESS
User Initiated
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_IN_PROGRESS
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Events (6)
Search events
Timestamp
Logical ID
Status
Status reason
2022-11-15 12:05:32 UTC+0530
WebServer
CREATE_IN_PROGRESS
Resource creation Initiated
2022-11-15 12:05:31 UTC+0530
WebServer
CREATE_IN_PROGRESS
-
2022-11-15 12:05:29 UTC+0530
WebServerSecurityGroup
CREATE_COMPLETE
-
2022-11-15 12:05:28 UTC+0530
WebServerSecurityGroup
CREATE_IN_PROGRESS
Resource creation Initiated
2022-11-15 12:05:23 UTC+0530
WebServerSecurityGroup
CREATE_IN_PROGRESS
-
2022-11-15 12:05:19 UTC+0530
CloudFormationStack
CREATE_IN_PROGRESS
User Initiated
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_IN_PROGRESS
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Resources (2)
Search resources
Logical ID
Physical ID
Type
Status
Module
WebServer
i-0aab650e48901c422
AWS::EC2::Instance
CREATE_IN_PROGRESS
-
WebServerSecurityGroup
CloudFormationStack-WebServerSecurityGroup-1ERNZWWZ47DUS
AWS::EC2::SecurityGroup
CREATE_COMPLETE
-
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_COMPLETE
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Outputs (1)
Search outputs
1
Key
Value
Description
Export name
WebsiteURL
http://ec2-34-221-44-82.us-west-2.compute.amazonaws.com/wordpress
WordPress Website
-
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_COMPLETE
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Events (9)
Search events
Timestamp
Logical ID
Status
Status reason
2022-11-15 12:07:08 UTC+0530 CloudFormationStack CREATE_COMPLETE -
2022-11-15 12:07:06 UTC+0530 WebServer CREATE_COMPLETE -
2022-11-15 12:07:05 UTC+0530 WebServer CREATE_IN_PROGRESS Received SUCCESS signal with UniqueId i-0aab650e48901c422
2022-11-15 12:05:32 UTC+0530 WebServer CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:31 UTC+0530 WebServer CREATE_IN_PROGRESS -
2022-11-15 12:05:29 UTC+0530 WebServerSecurityGroup CREATE_COMPLETE -
2022-11-15 12:05:28 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:23 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS -
2022-11-15 12:05:19 UTC+0530 CloudFormationStack CREATE_IN_PROGRESS User Initiated
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_COMPLETE
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Resources (2)
Search resources
Logical ID
Physical ID
Type
Status
Module
WebServer i-0aab650e48901c422 AWS::EC2::Instance CREATE_COMPLETE -
WebServerSecurityGroup CloudFormationStack-WebServerSecurityGroup-1ERNZWWZ47DUS AWS::EC2::SecurityGroup CREATE_COMPLETE -
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Install Wordpress

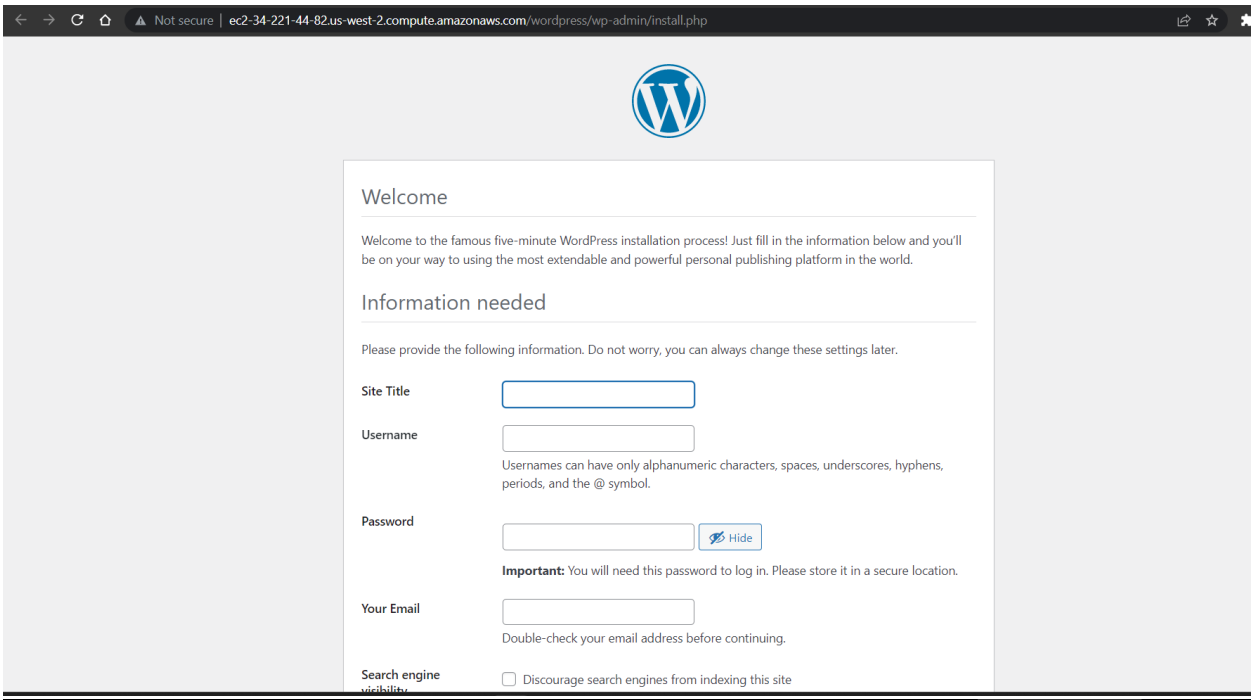
The CloudFormation stack generates a URL for the Wordpress site as output. To complete your testing, you need to open the URL and enter the values needed to complete the Wordpress installation.

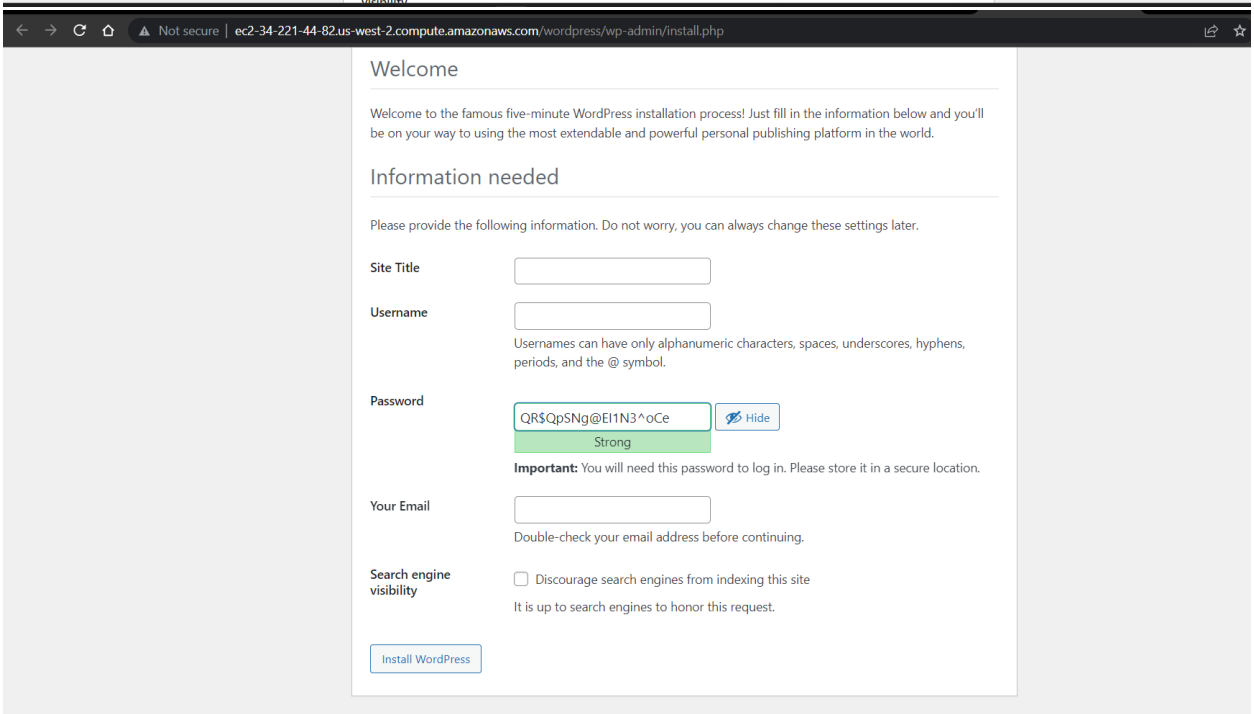
1. View the **Outputs** tab of the CloudFormation stack you just created, and open the **WebsiteURL** in a new tab.
2. In that new tab, under the **Information needed** section, enter the following for the specified fields:
    - Site Title: `Demo Site`
    - Username: `demo`
    - Password: `developerdemosite!!`
    - Your Email: `demo@example.com`
    - Search engine visibility: `selected`
3. Click the button labeled **Install Wordpress**.
4. On the **Success!** screen, click the button labeled **Log in**. Enter the username and password for the demo user and click the button labeled **Log in**. This opens the administrative dashboard for the site.

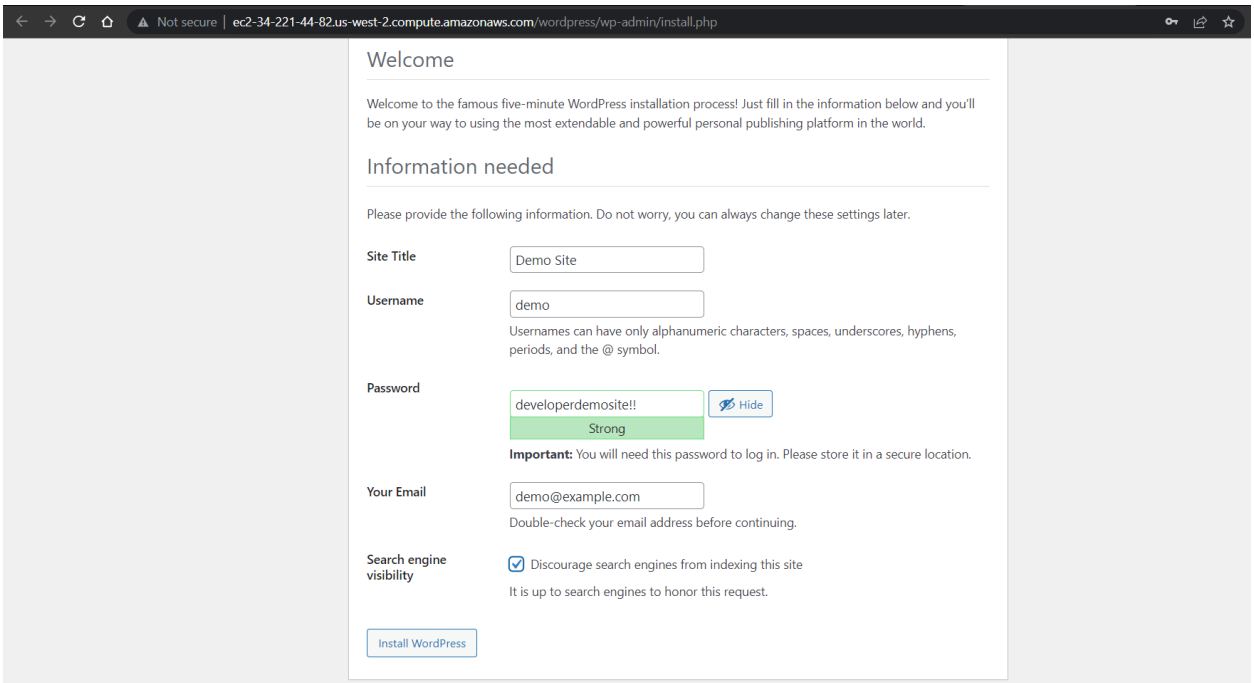
The CloudFormation stack generates a URL for the Wordpress site as output. To complete your testing, you need to open the URL and enter the values needed to complete the Wordpress installation.

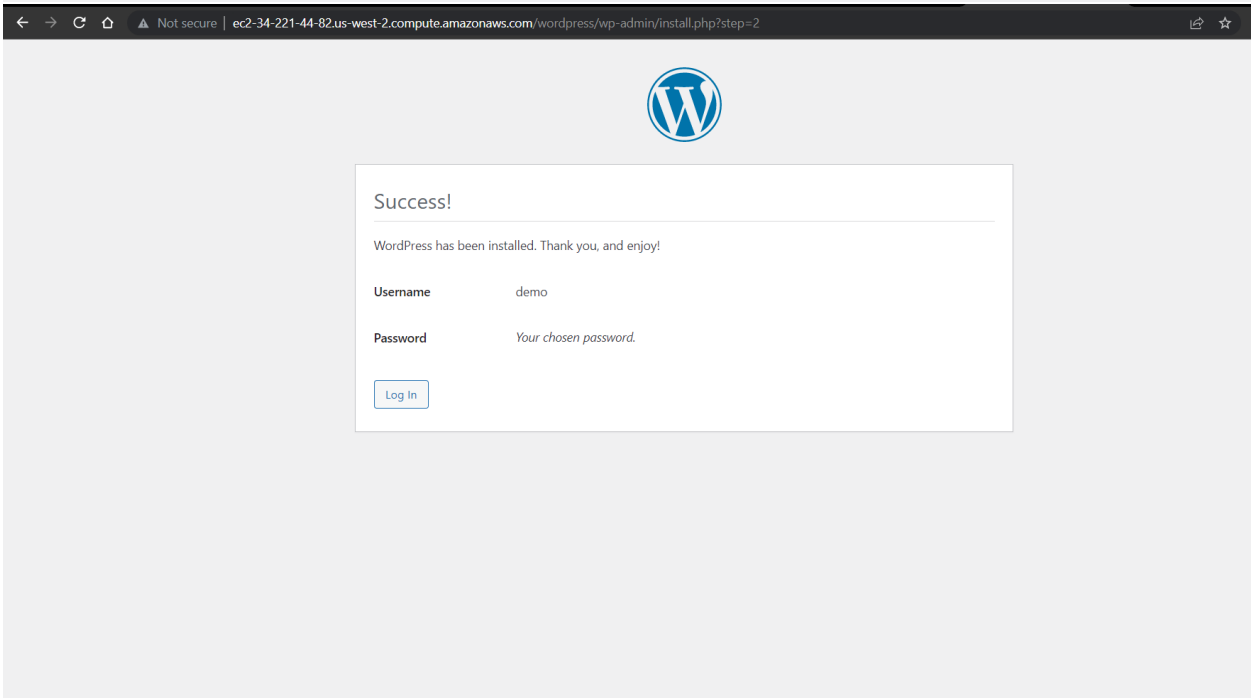
1. View the **Outputs** tab of the CloudFormation stack you just created, and open the **WebsiteURL** in a new tab.
2. In that new tab, under the **Information needed** section, enter the following for the specified fields:
    - Site Title: `Demo Site`
    - Username: `demo`
    - Password: `developerdemosite!!`
    - Your Email: `demo@example.com`
    - Search engine visibility: `selected`
3. Click the button labeled **Install Wordpress**.
4. On the **Success!** screen, click the button labeled **Log in**. Enter the username and password for the demo user and click the button labeled **Log in**. This opens the administrative dashboard for the site.

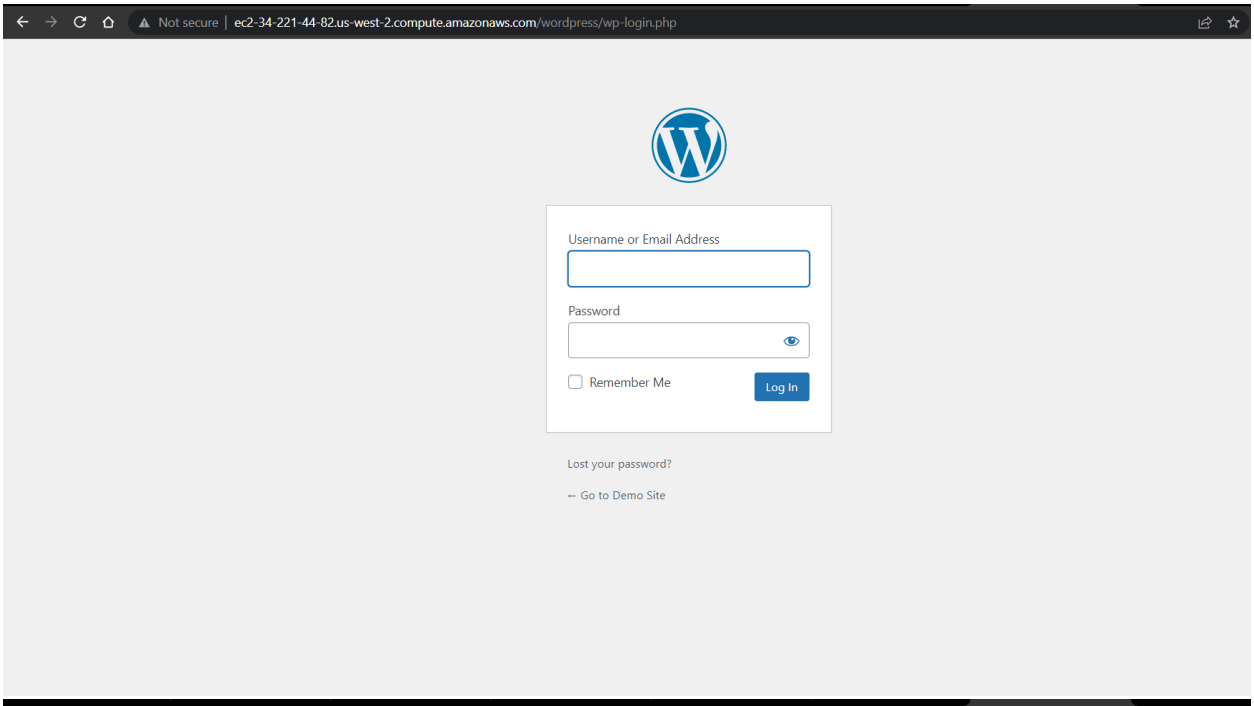
The Wordpress admin dashboard is the interface that the developers will use to install their themes and plugins. By completing the Wordpress installation, you've confirmed that the CloudFormation template is complete and ready for use by the development team. Now that you are ready to share the template with them, you need to delete the demo stack you created. You'll do that in the next challenge.

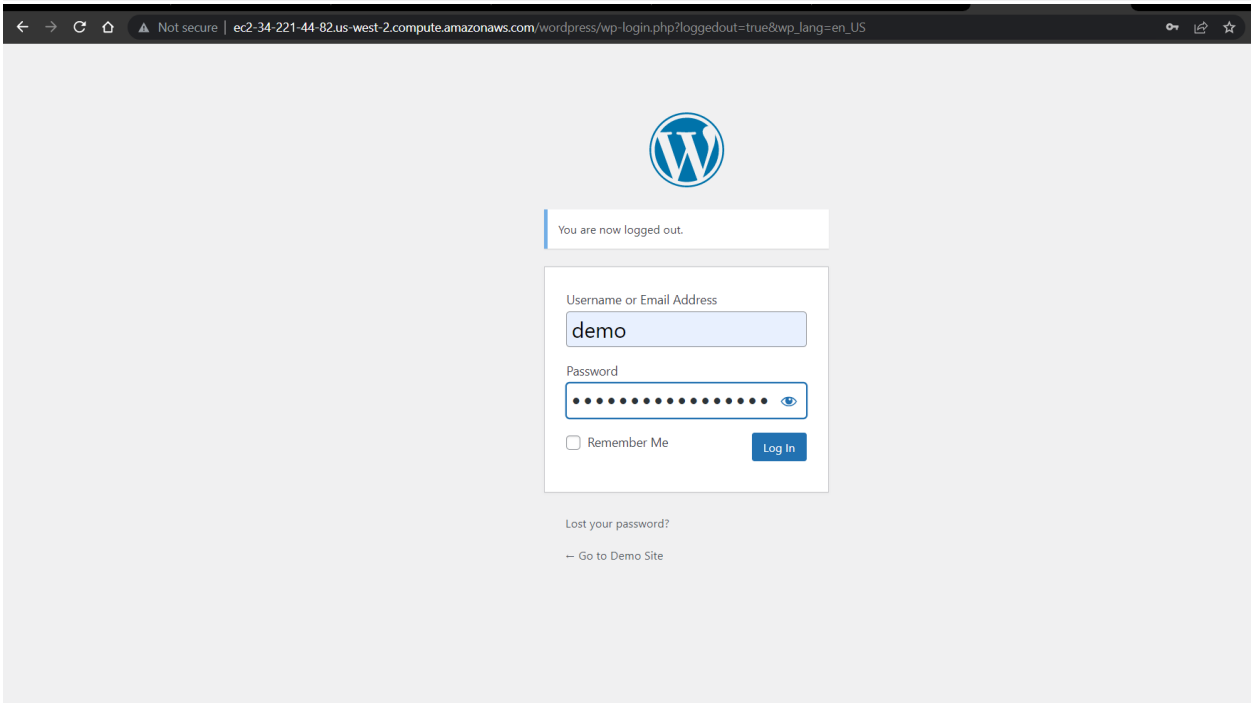

Not secure | ec2-34-221-44-82.us-west-2.compute.amazonaws.com/wordpress/wp-admin/install.php
Welcome
Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.
Information needed
Please provide the following information. Do not worry, you can always change these settings later.
Site Title
Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password
Hide
Important: You will need this password to log in. Please store it in a secure location.
Your Email
Double-check your email address before continuing.
Search engine visibility
Discourage search engines from indexing this site

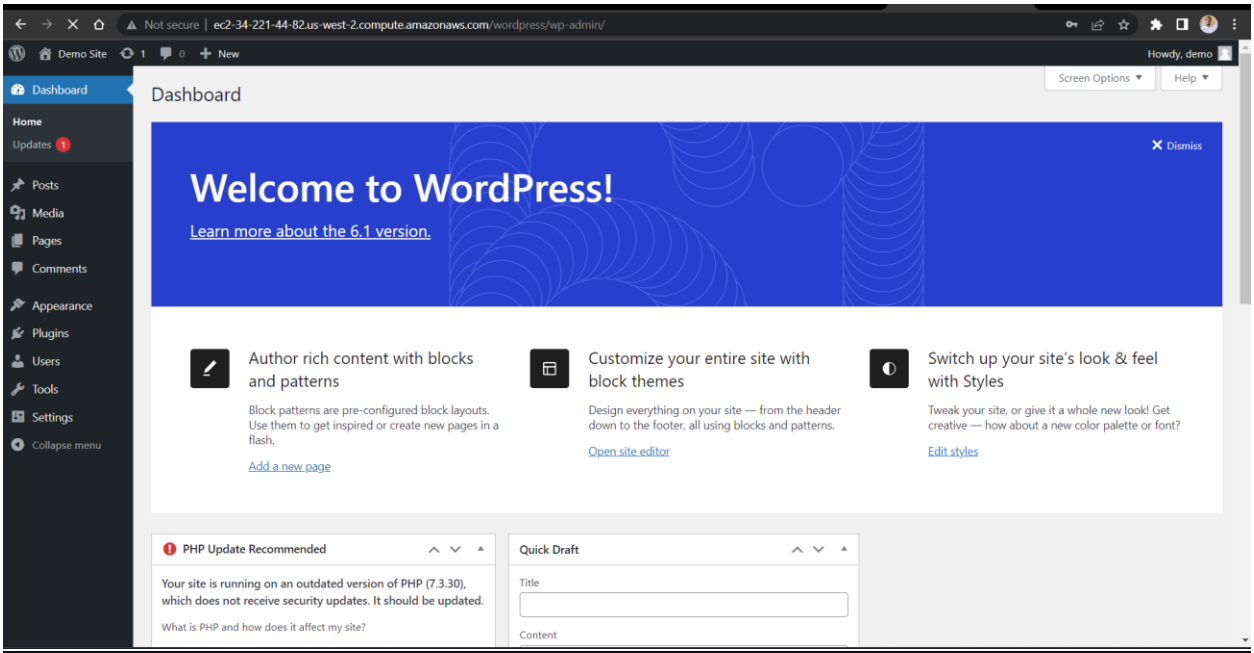

Not secure | ec2-34-221-44-82.us-west-2.compute.amazonaws.com/wordpress/wp-admin/install.php
Welcome
Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.
Information needed
Please provide the following information. Do not worry, you can always change these settings later.
Site Title
Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password
QR$QpSNg@El1N3^oCe
Hide
Strong
Important: You will need this password to log in. Please store it in a secure location.
Your Email
Double-check your email address before continuing.
Search engine visibility
Discourage search engines from indexing this site
It is up to search engines to honor this request.
Install WordPress


Not secure | ec2-34-221-44-82.us-west-2.compute.amazonaws.com/wordpress/wp-admin/install.php
Welcome
Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.
Information needed
Please provide the following information. Do not worry, you can always change these settings later.
Site Title
Demo Site
Username
demo
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password
developerdemosite!!
Hide
Strong
Important: You will need this password to log in. Please store it in a secure location.
Your Email
demo@example.com
Double-check your email address before continuing.
Search engine visibility
Discourage search engines from indexing this site
It is up to search engines to honor this request.
Install WordPress


Not secure | ec2-34-221-44-82.us-west-2.compute.amazonaws.com/wordpress/wp-admin/install.php?step=2
Success!
WordPress has been installed. Thank you, and enjoy!
Username
demo
Password
Your chosen password.
Log In


Not secure | ec2-34-221-44-82.us-west-2.compute.amazonaws.com/wordpress/wp-login.php
Username or Email Address
Password
Remember Me
Log In
Lost your password?
← Go to Demo Site


Not secure | ec2-34-221-44-82.us-west-2.compute.amazonaws.com/wordpress/wp-login.php?loggedout=true&wp_lang=en_US
You are now logged out.
Username or Email Address
demo
Password
Remember Me
Log In
Lost your password?
← Go to Demo Site



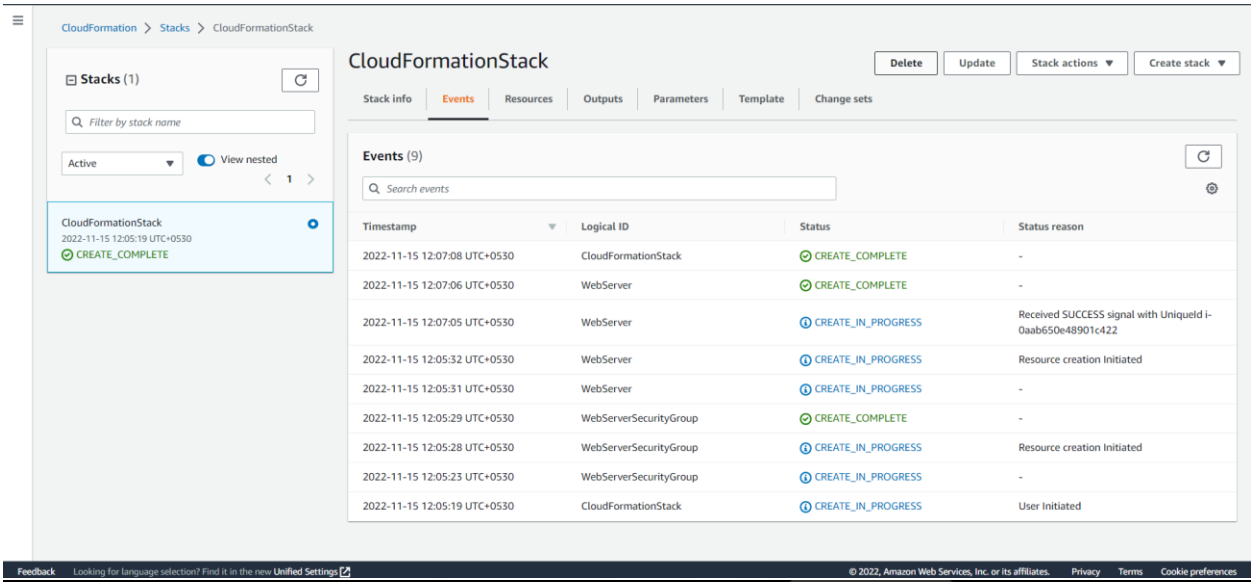
Delete the CloudFormation Stack

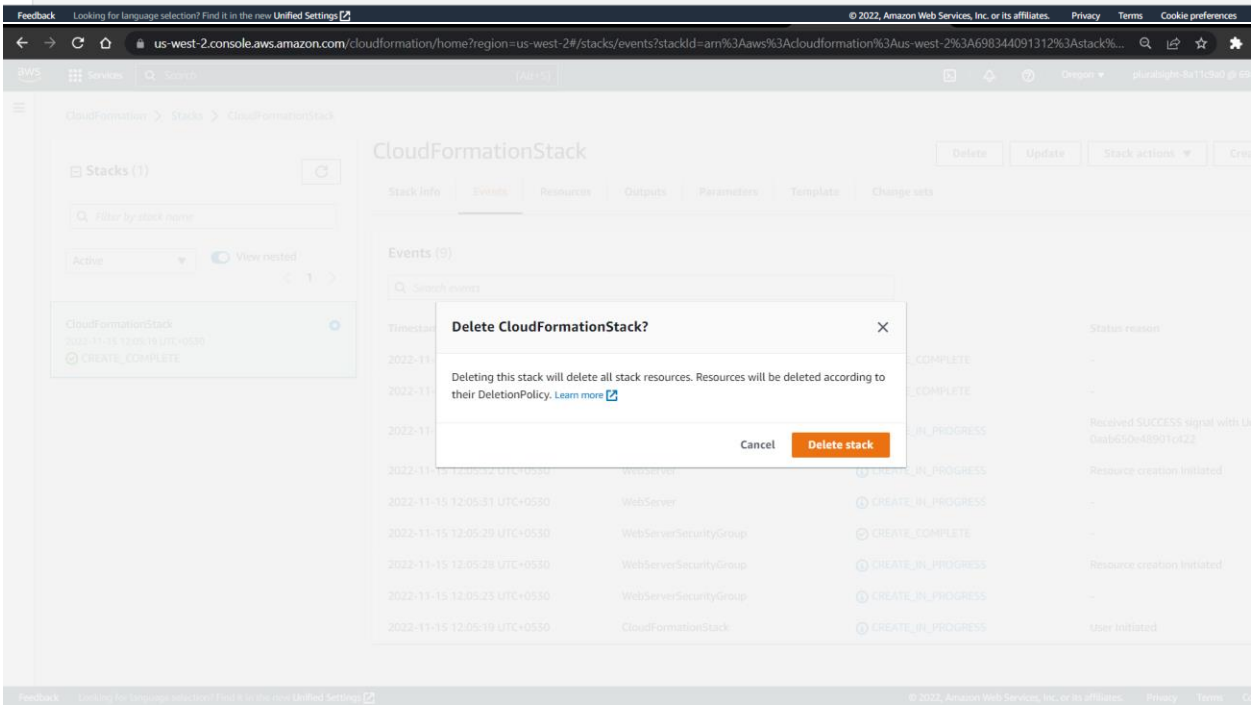
Now that your CloudFormation template has been developed and tested, you can remove the stack and its resources.

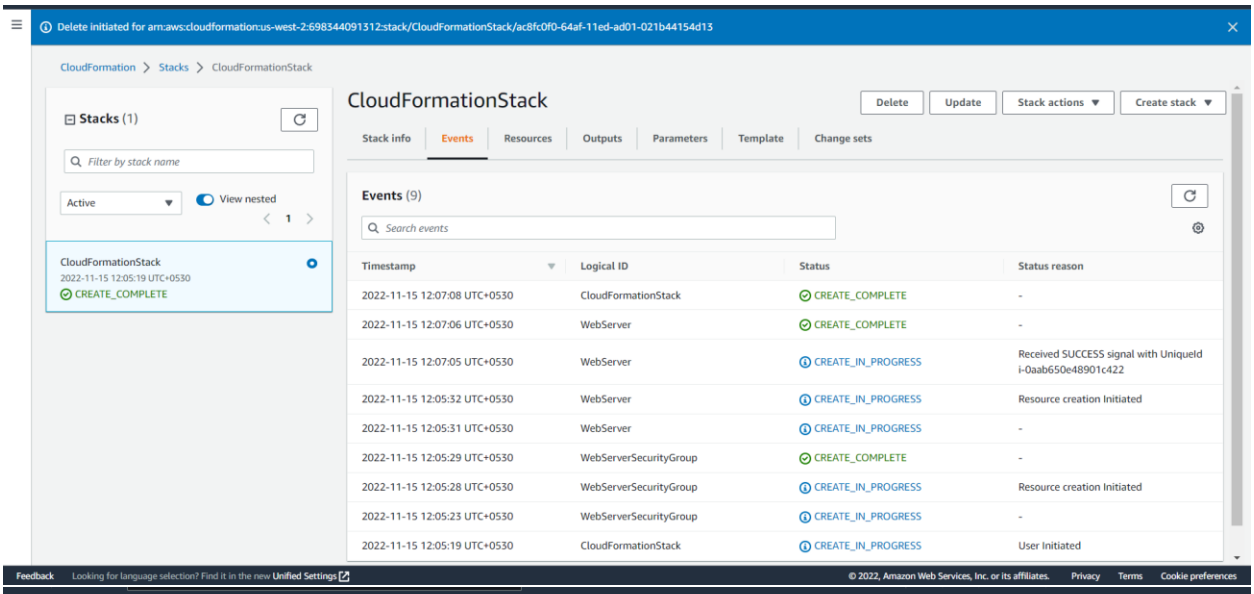
1. Go back to the CloudFormation tab and open the **Events** tab of your stack.
2. Click the button labeled **Delete**.
3. Confirm that you are deleting the stack by clicking the button labeled **Delete stack**.
4. While the stack is being deleted, monitor the **Events** tab by occasionally clicking the refresh button to update the screen.
5. Note the **DELETE_IN_PROGRESS** and **DELETE_COMPLETE** status messages for each of the stack's resources.

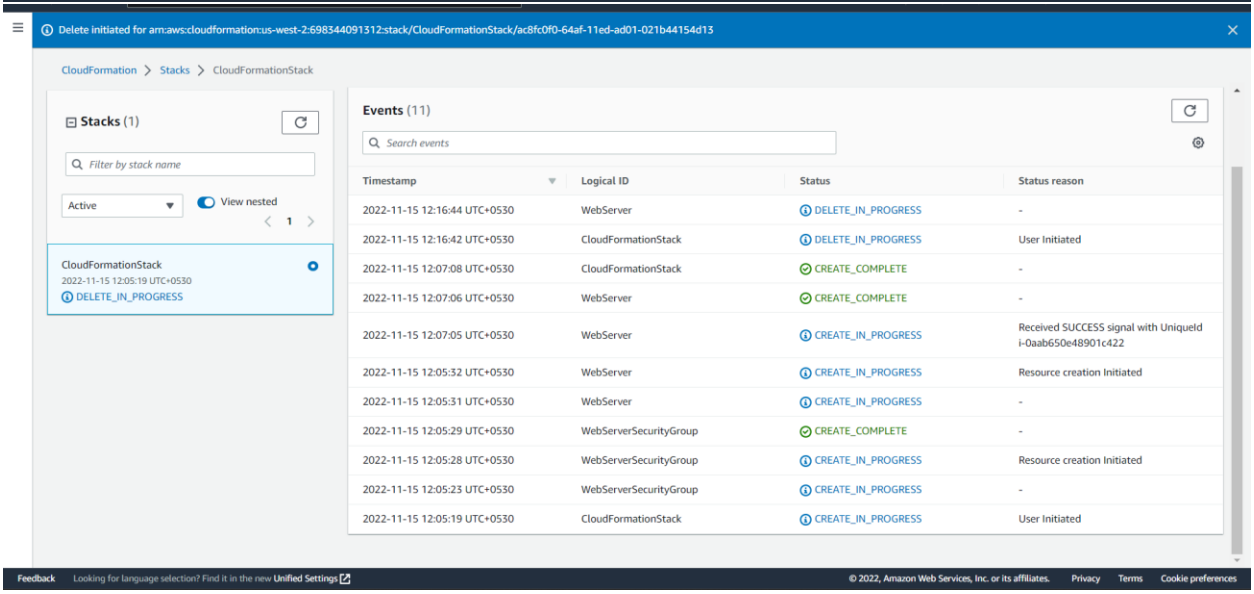
Once the stack has been deleted, you can view it by going to the **Stacks** page of the CloudFormation console and filtering on **Deleted**. Click the stack's name to view the stack and any of its details.

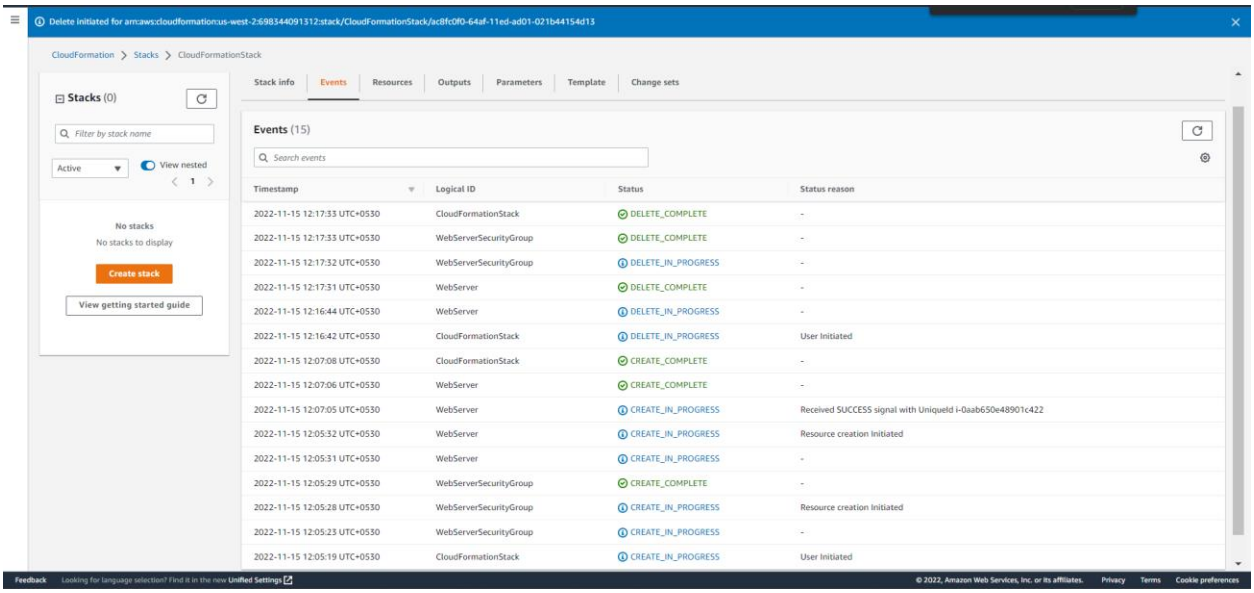
Congratulations! You have authored a CloudFormation template and used it to deploy a stack. By using this template, the development team will have a consistent deployment method for their Wordpress sites and an easy way to manage all of the site's resources.


CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_COMPLETE
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Events (9)
Search events
Timestamp
Logical ID
Status
Status reason
2022-11-15 12:07:08 UTC+0530 CloudFormationStack CREATE_COMPLETE -
2022-11-15 12:07:06 UTC+0530 WebServer CREATE_COMPLETE -
2022-11-15 12:07:05 UTC+0530 WebServer CREATE_IN_PROGRESS Received SUCCESS signal with UniqueId i-0aab650e48901c422
2022-11-15 12:05:32 UTC+0530 WebServer CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:31 UTC+0530 WebServer CREATE_IN_PROGRESS -
2022-11-15 12:05:29 UTC+0530 WebServerSecurityGroup CREATE_COMPLETE -
2022-11-15 12:05:28 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:23 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS -
2022-11-15 12:05:19 UTC+0530 CloudFormationStack CREATE_IN_PROGRESS User Initiated


Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
us-west-2.console.aws.amazon.com/cloudformation/home?region=us-west-2#/stacks/events?stackId=arn%3Aaws%3Acloudformation%3Aus-west-2%3A698344091312%3Astack%...
Delete CloudFormationStack?
Deleting this stack will delete all stack resources. Resources will be deleted according to their DeletionPolicy. Learn more
Cancel
Delete stack


Delete initiated for arn:aws:cloudformation:us-west-2:698344091312:stack/CloudFormationStack/ac8fc0f0-64af-11ed-ad01-021b44154d13
CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
CREATE_COMPLETE
CloudFormationStack
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Events (9)
Search events
Timestamp
Logical ID
Status
Status reason
2022-11-15 12:07:08 UTC+0530 CloudFormationStack CREATE_COMPLETE -
2022-11-15 12:07:06 UTC+0530 WebServer CREATE_COMPLETE -
2022-11-15 12:07:05 UTC+0530 WebServer CREATE_IN_PROGRESS Received SUCCESS signal with UniqueId i-0aab650e48901c422
2022-11-15 12:05:32 UTC+0530 WebServer CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:31 UTC+0530 WebServer CREATE_IN_PROGRESS -
2022-11-15 12:05:29 UTC+0530 WebServerSecurityGroup CREATE_COMPLETE -
2022-11-15 12:05:28 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:23 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS -
2022-11-15 12:05:19 UTC+0530 CloudFormationStack CREATE_IN_PROGRESS User Initiated
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Delete initiated for arn:aws:cloudformation:us-west-2:698344091312:stack/CloudFormationStack/ac8fc0f0-64af-11ed-ad01-021b44154d13
CloudFormation > Stacks > CloudFormationStack
Stacks (1)
Filter by stack name
Active
View nested
1
CloudFormationStack
2022-11-15 12:05:19 UTC+0530
DELETE_IN_PROGRESS
Events (11)
Search events
Timestamp
Logical ID
Status
Status reason
2022-11-15 12:16:44 UTC+0530 WebServer DELETE_IN_PROGRESS -
2022-11-15 12:16:42 UTC+0530 CloudFormationStack DELETE_IN_PROGRESS User Initiated
2022-11-15 12:07:08 UTC+0530 CloudFormationStack CREATE_COMPLETE -
2022-11-15 12:07:06 UTC+0530 WebServer CREATE_COMPLETE -
2022-11-15 12:07:05 UTC+0530 WebServer CREATE_IN_PROGRESS Received SUCCESS signal with UniqueId i-0aab650e48901c422
2022-11-15 12:05:32 UTC+0530 WebServer CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:31 UTC+0530 WebServer CREATE_IN_PROGRESS -
2022-11-15 12:05:29 UTC+0530 WebServerSecurityGroup CREATE_COMPLETE -
2022-11-15 12:05:28 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:23 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS -
2022-11-15 12:05:19 UTC+0530 CloudFormationStack CREATE_IN_PROGRESS User Initiated
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Delete initiated for arn:aws:cloudformation:us-west-2:698344091312:stack/CloudFormationStack/ac8fc0f0-64af-11ed-ad01-021b44154d13
CloudFormation > Stacks > CloudFormationStack
Stacks (0)
Filter by stack name
Active
View nested
1
No stacks
No stacks to display
Create stack
View getting started guide
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Events (15)
Search events
Timestamp
Logical ID
Status
Status reason
2022-11-15 12:17:33 UTC+0530 CloudFormationStack DELETE_COMPLETE -
2022-11-15 12:17:33 UTC+0530 WebServerSecurityGroup DELETE_COMPLETE -
2022-11-15 12:17:32 UTC+0530 WebServerSecurityGroup DELETE_IN_PROGRESS -
2022-11-15 12:17:31 UTC+0530 WebServer DELETE_COMPLETE -
2022-11-15 12:16:44 UTC+0530 WebServer DELETE_IN_PROGRESS -
2022-11-15 12:16:42 UTC+0530 CloudFormationStack DELETE_IN_PROGRESS User Initiated
2022-11-15 12:07:08 UTC+0530 CloudFormationStack CREATE_COMPLETE -
2022-11-15 12:07:06 UTC+0530 WebServer CREATE_COMPLETE -
2022-11-15 12:07:05 UTC+0530 WebServer CREATE_IN_PROGRESS Received SUCCESS signal with UniqueId i-0aab650e48901c422
2022-11-15 12:05:32 UTC+0530 WebServer CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:31 UTC+0530 WebServer CREATE_IN_PROGRESS -
2022-11-15 12:05:29 UTC+0530 WebServerSecurityGroup CREATE_COMPLETE -
2022-11-15 12:05:28 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS Resource creation Initiated
2022-11-15 12:05:23 UTC+0530 WebServerSecurityGroup CREATE_IN_PROGRESS -
2022-11-15 12:05:19 UTC+0530 CloudFormationStack CREATE_IN_PROGRESS User Initiated
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences