



slalom

Slalom AWS Introduction Workshop

Lab 2 Instructions

Introduction

Welcome to Lab 2 of the Slalom AWS Introduction Workshop! This document outlines 13 steps, walking you through the creation of the AWS resources required to extend and improve upon the architecture created in Lab 1.

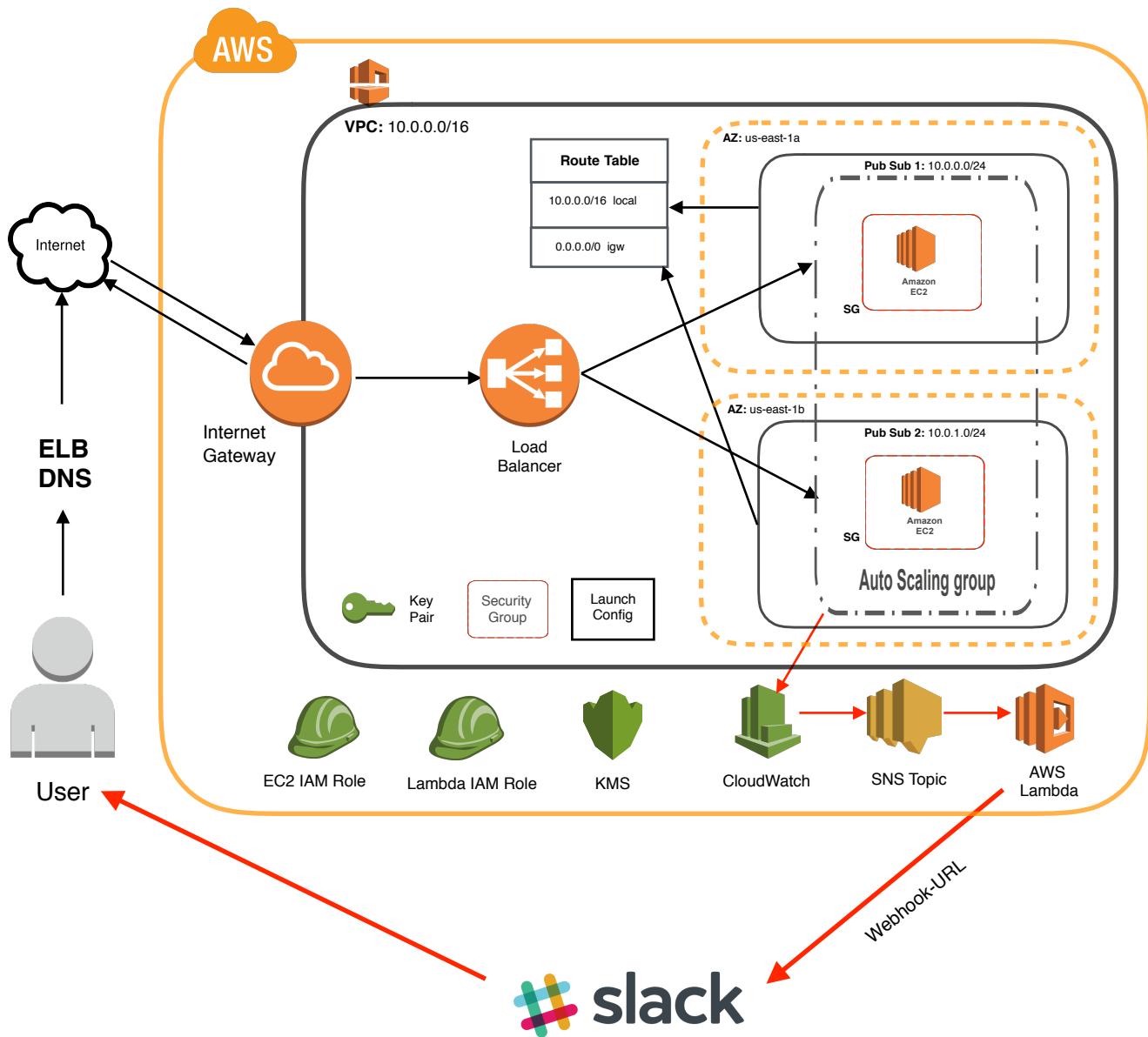
In this lab we will be using services such as AWS Simple Notification Service (SNS), AWS Key Management Service (KMS), AWS CloudWatch and AWS Lambda to create an alert from your autoscaling group that will post to your own personal Slack channel.

At the end of each step (excluding Step 1) you will find an architectural summary of the AWS resources that you have built so far.

The next two pages contain a brief summary of what is achieved within each step of the process. Please reach out to Bruce, Jeff or Todd for clarification on any of the content covered within this document!

The image at the top of the following page outlines the final architecture of our system following the completion of this lab.

Lab 2 - Final Architecture



Step Overview

Step 1: Logging in to the AWS Console

This step simply directs you to log in to your account and select the appropriate AWS Region

Step 2: Create your own private Slack channel

Slack is a cloud-based team collaboration tool. Although on the surface it may seem similar to any other messaging platform, its capabilities extend far beyond that. In this lab we will see that Slack has deep integrations with a number of technologies, including AWS. In this step, you create your own private Slack channel.

Step 3: Add the Webhooks-Integration to your private Slack channel

As mentioned above, Slack has a number of deep integrations with third party tools. Within this step, you will install the Webhooks-Integration within your Slack channel. The resulting webhook URL (a URL that is unique to your Slack channel) can be used by these third party tools to post messages and statuses to Slack.

Step 4: Create an SNS Topic

Amazon Simple Notification Service (SNS) is one of AWS's oldest services. SNS is a fast, flexible messaging service that allows you to push notifications to a number of different AWS services, email users or third party services.

Step 5: Create an encrypted key using AWS Key Management Service (KMS)

AWS Key Management Service (KMS) is a service that makes it easy to create and manage encryption keys that you can use to encrypt your data. In this step, you will create a new encrypted key that will be used to encrypt the webhook URL generated in Step 3

Step 6: Create an IAM Role for running AWS Lambda functions

Remembering back to Lab 1, we used AWS Identity and Access Management (IAM) to create an IAM Role that we assigned to our EC2 instances in the autoscaling group. As a refresher, AWS IAM Roles and Policies can be used to help restrict how an AWS Resource can communicate with other AWS Resources. In this Step, we will create a

new IAM Role that we will eventually assign to an AWS Lambda function that we will create later on in the Lab.

Step 7: Use the AWS IAM Policy Generator to create and assign an IAM Policy

We will use the IAM Policy generator to generate an IAM policy that we will apply to the IAM Role we just created in Step 6. This policy will contain a permission which will allow it to decrypt the key we created in Step 5.

Step 8: Update your Auto Scaling Group Instance Count

Step 9 will require us to create CloudWatch alarms for our Auto Scaling Group. In order to set up this alarm, we must have some active EC2 instances in our group. This step will increase the Min and Desired capacities to allow for alarm creation to occur in Step 9.

Step 9: Set up an AWS CloudWatch Alarm

AWS CloudWatch is a service that provides monitoring capabilities within AWS. You can monitor important statistics about a number of AWS services. In our case, we are going to create a CloudWatch alarm that will be monitoring the CPU utilization levels on the EC2 instances within your autoscaling group.

Step 10: Create and configure an AWS Lambda function using provided blueprints

AWS Lambda is a service provided to allow serverless compute capabilities. Instead of having an EC2 instance that runs all of the time, AWS Lambda allows you to create a code function that will be executed at certain times. In our case, we are using a Lambda function blueprint provided by AWS that contains code to send a notification to Slack.

Step 11: Recreate the ELB from Lab 1

As part of the cleanup for Lab 1, you were asked to delete the ELB. In this step, you will recreate it again so that we can test the functionality of our webpage once again.

Step 12: Increase the min and desired capacity of your AutoScaling Group

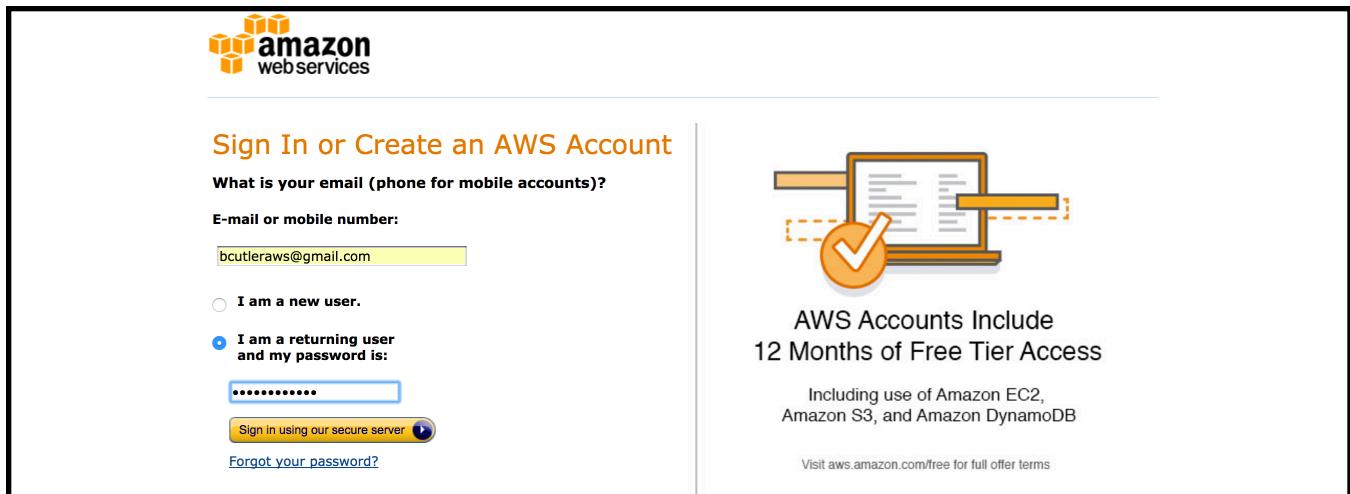
Also as part of the cleanup for Lab 1, you were asked to decrease the min and desired capacity values to 0. This ensured that any EC2 instances in your autoscaling group were terminated. By turning the value back up to 2, this will create 2 brand new EC2 instances as part of your autoscaling group.

Step 13: Test it out!

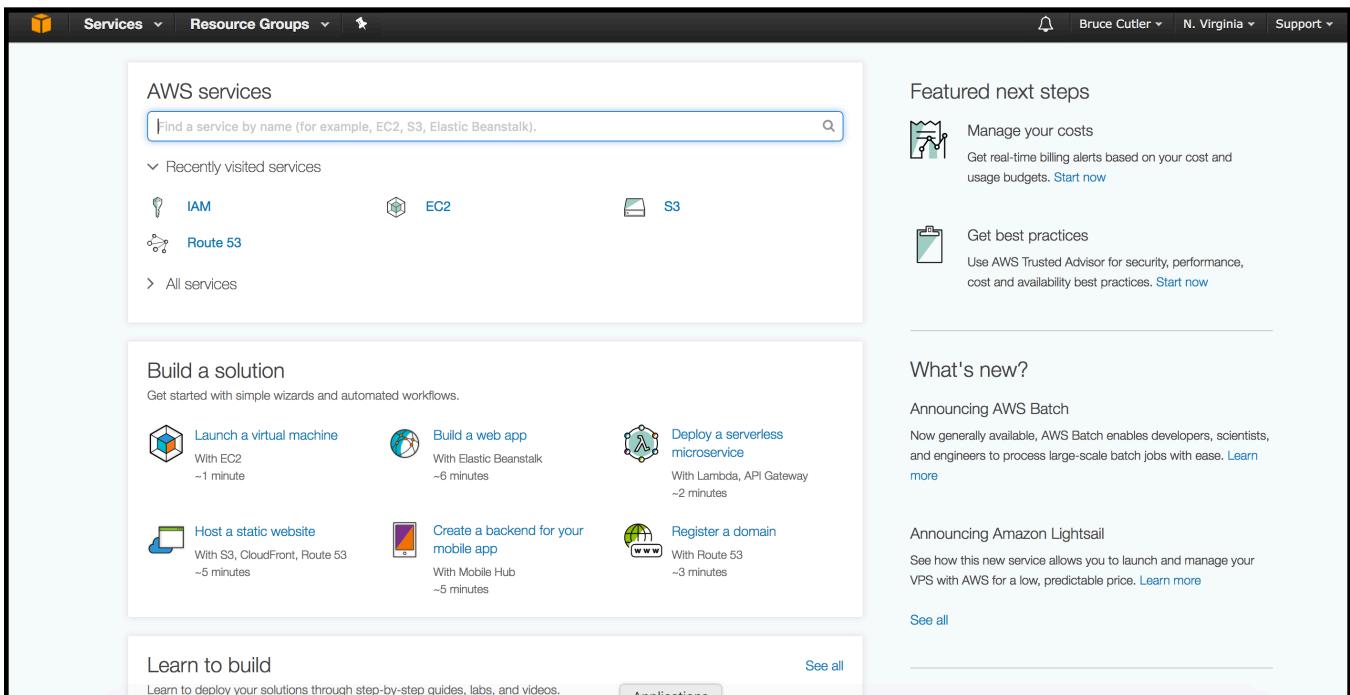
Once we've build all of the above components, we can use the DNS value from the load balancer in a browser to view our simple webpage.

Step 1: Logging in to the AWS Console

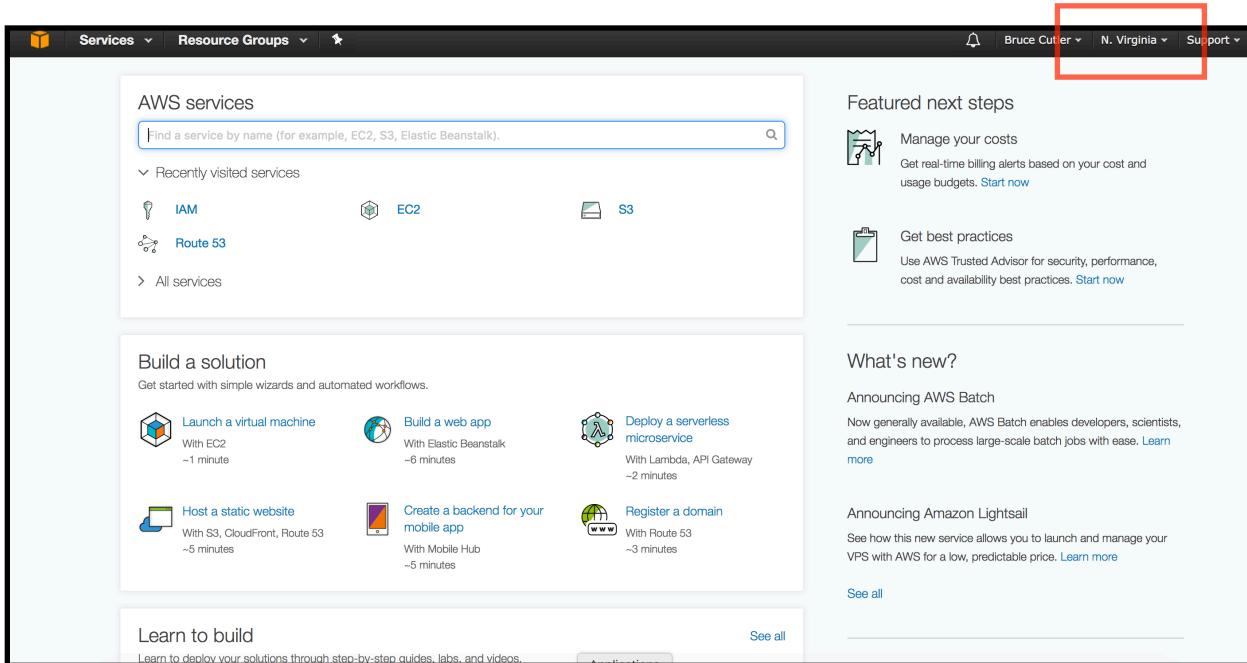
1. In a web browser, navigate to <https://console.aws.amazon.com/console/home>



2. Log in with the email / password combination you signed up for your account with. This will bring you to the AWS Console home page:

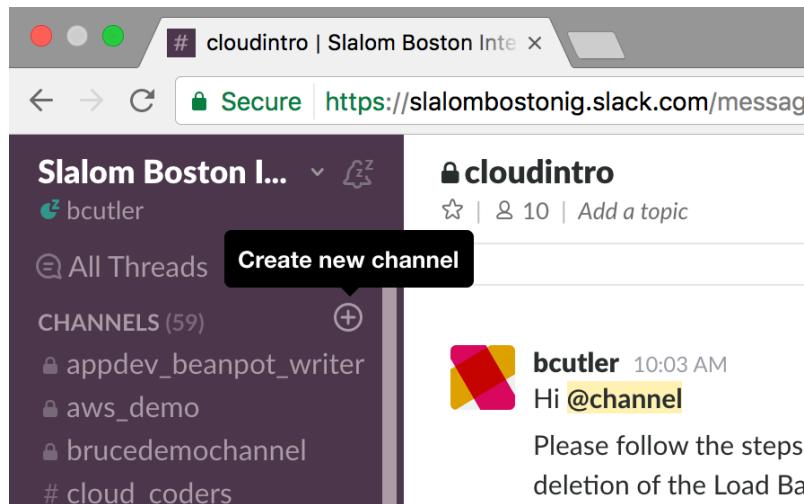


3. At the top right of the AWS Console screen, ensure that the **N. Virginia** is selected as your AWS Region



Step 2: Create your own private Slack channel

1. Whether you use Slack from your browser or using the Slack Desktop app, click on the (+) button beside Channels:



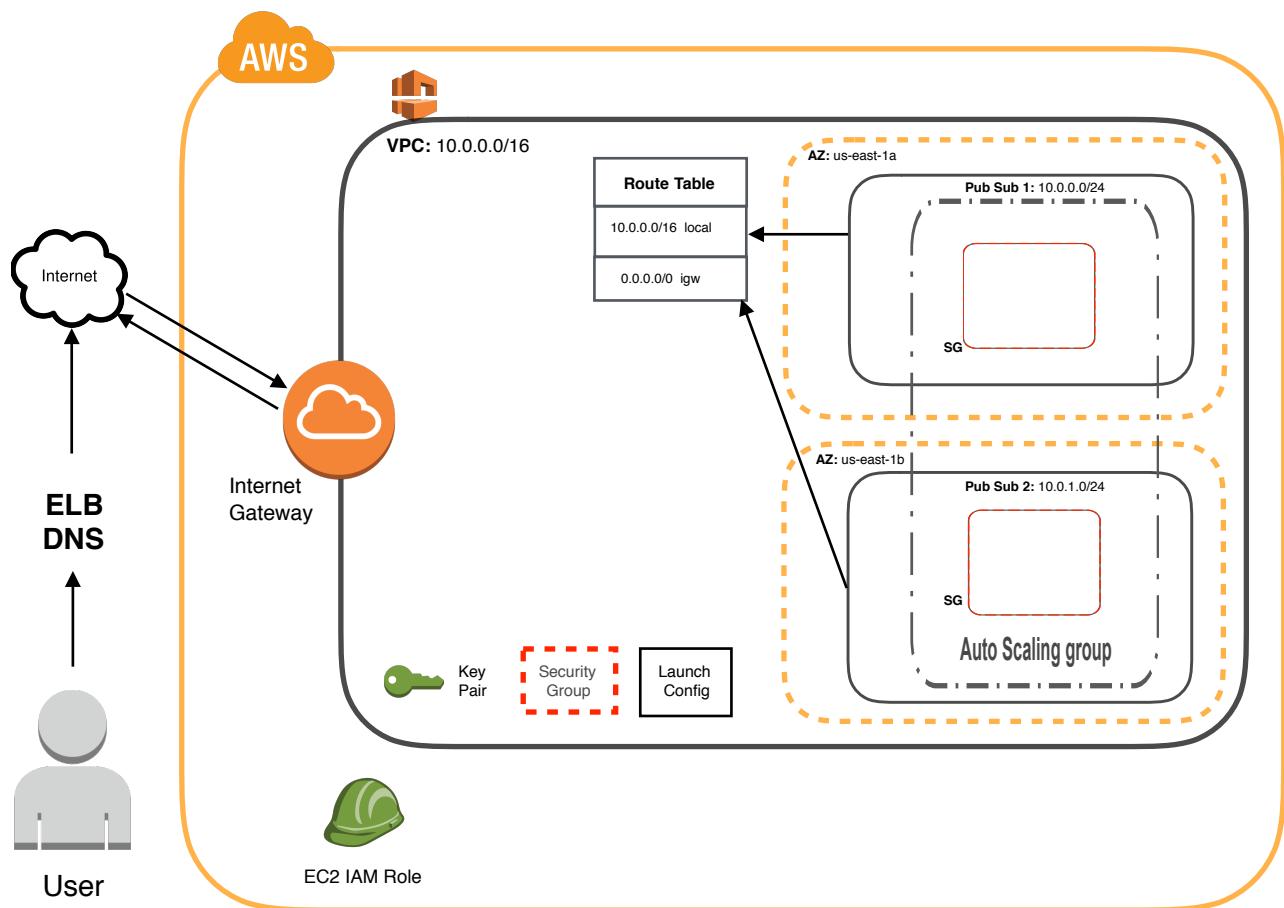
2. In the Create Channel screen that appears, ensure that you:
 - Change the channel type to **Private**
 - Give it a unique **Name**
 - Provide a **Purpose** description for the channel

The dialog box has a title 'Create a private channel'. It contains the following fields:

- A radio button labeled 'Private' is selected, with a note: 'This channel can only be joined or viewed by invite.'
- A 'Name' input field containing 'brucedemochannel1'. A note below it says: 'Names must be lowercase, without spaces or periods, and shorter than 22 characters.'
- A 'Purpose (optional)' input field containing 'slalom-aws-intro Lab2'.
- A 'What's this channel about?' input field.
- An optional 'Send invites to:' section with a search bar and a note: 'Search by name'.
- At the bottom are two buttons: 'Cancel' and 'Create Channel'.

3. Proceed to click **Create Channel** once the above 3 conditions have been satisfied

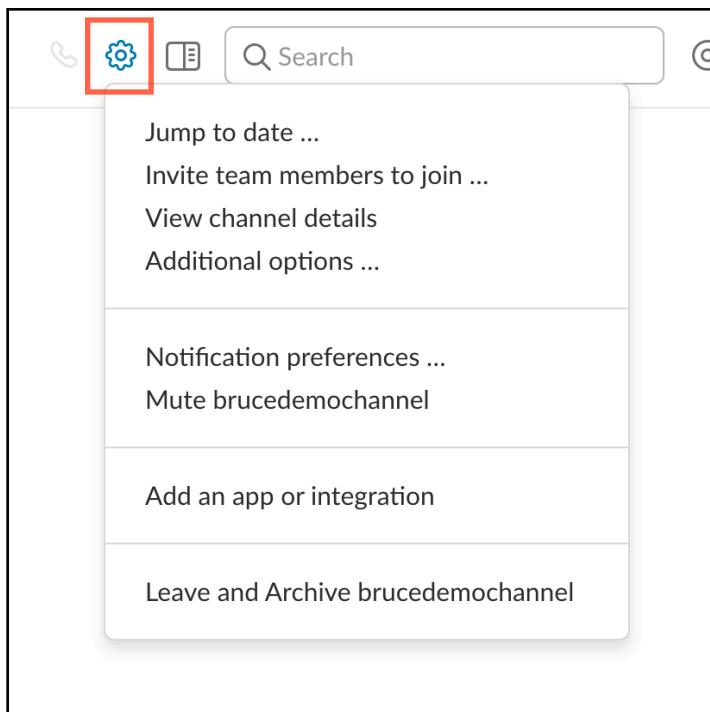
System Architecture: Step 2



Step 3: Add the Incoming WebHooks integration to your Slack Channel

Adding in the Incoming WebHooks integration will allow you to generate a URL that uniquely identifies your private Slack channel. This URL can then be used by other services, such as AWS, to push messages to that channel.

1. Ensure you have your new Slack channel selected from the list of available channels
2. Navigate to the top right of the Window and select the cog symbol, indicating settings



3. Select the **Add an app or integration** menu item

4. A new page will load with a search bar. Enter “**webhook**” and you should see the following:

The screenshot shows the Slack App Directory search results for "webhook". The search bar at the top contains "webhook". Below it, three results are listed:

- Incoming WebHooks**: Send data into Slack in real-time. It features a red and black icon of two interlocking circles.
- Outgoing WebHooks**: Get data out of Slack in real-time. It features a red and black icon of two interlocking circles.
- Amazon SQS**: A distributed queue messaging service. It features a yellow icon of a building.

5. Click on **Incoming WebHooks** in the search results
6. In the page that loads, select the green “**Request**” button. This will alert our Slalom Slack Team Owners that we would like to add this new functionality.

The screenshot shows the "Incoming WebHooks" configuration page. At the top left is a red and black icon of two interlocking circles. To its right is the title "Incoming WebHooks". Below the title is a brief description: "Incoming Webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later." Further down is a section titled "Message Attachments" with the subtext: "Message Attachments can also be used in Incoming Webhooks to display richly-formatted messages that stand out from regular chat messages." On the left side of the main content area is a green button labeled "Request". Below the "Request" button is a "Learn More" link. At the bottom left are links for "Help and support" and "Privacy policy". At the bottom center is a configuration card showing a log entry: "Posts to brucedemochannel as incoming-webhook" by "butter" on Apr 10, 2017, with an edit icon to the right.

7. Once the Slack team owner has approved your request (I've alerted our Slack team owners, but this could take a few minutes), you will be notified by the Slackbot channel. Within that notification message there will be a link to Install the integration. Click on the **Install** link

The screenshot shows two windows side-by-side. On the left is a 'DIRECT MESSAGES' list with a message from 'slackbot'. A red box highlights this message. On the right is a browser window titled 'APM - slack.com' showing the 'Incoming WebHooks' configuration page. A red box highlights the 'install it.' button in the message from slackbot.

DIRECT MESSAGES

slackbot 2:05 PM ☆

@Chris Allen approved your request to install this app 🙌. Go ahead and [install it.](#)

Incoming WebHooks

Incoming Webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later.

[Message Attachments](#) can also be used in Incoming Webhooks to display richly-formatted messages that stand out from regular chat messages.

Permissions & Scopes

- Confirm user's identity ([identify](#))
- Post to specific channels in Slack ([incoming-webhook](#))

8. The following page will load. Click the green **Add Configuration** button

The screenshot shows the 'Incoming WebHooks' configuration page. It features a logo icon, a title 'Incoming WebHooks', a description of what it does, and a large green 'Add Configuration' button. Below the button are links for 'Learn More', 'Help and support', and 'Privacy policy'.

Incoming WebHooks

Incoming Webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later.

[Message Attachments](#) can also be used in Incoming Webhooks to display richly-formatted messages that stand out from regular chat messages.

Add Configuration

Learn More

Requires approval from a Team Owner. [Browse approved apps](#)

Help and support >

Privacy policy >

9. In the page that loads, ensure that you select your new channel created in **Step 2**. Private channels can be found towards the bottom of the dropdown list.

Click the green **Add Incoming WebHooks Integration** button once you have your channel selected.

The screenshot shows the 'Incoming WebHooks' configuration page. At the top, there's a logo and the text 'Incoming WebHooks' followed by 'Send data into Slack in real-time.' Below this, a note says: 'Incoming Webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later.' Another note below it says: 'Message Attachments can also be used in Incoming Webhooks to display richly-formatted messages that stand out from regular chat messages.' A yellow warning box titled 'New to Slack integrations?' contains a note: 'Check out our [Getting Started](#) guide to familiarize yourself with the most common types of integrations, and tips to keep in mind while building your own. You can also [register as a developer](#) to let us know what you're working on, and to receive future updates to our APIs.' The main section is titled 'Post to Channel' with the sub-instruction: 'Start by choosing a channel where your Incoming Webhook will post messages to.' A dropdown menu shows 'brucedemochannel' selected, with an option to 'create a new channel'. A large green button at the bottom right says 'Add Incoming WebHooks integration'. At the very bottom, a small note says: 'By creating an incoming webhook, you agree to the [Slack API Terms of Service](#)'.

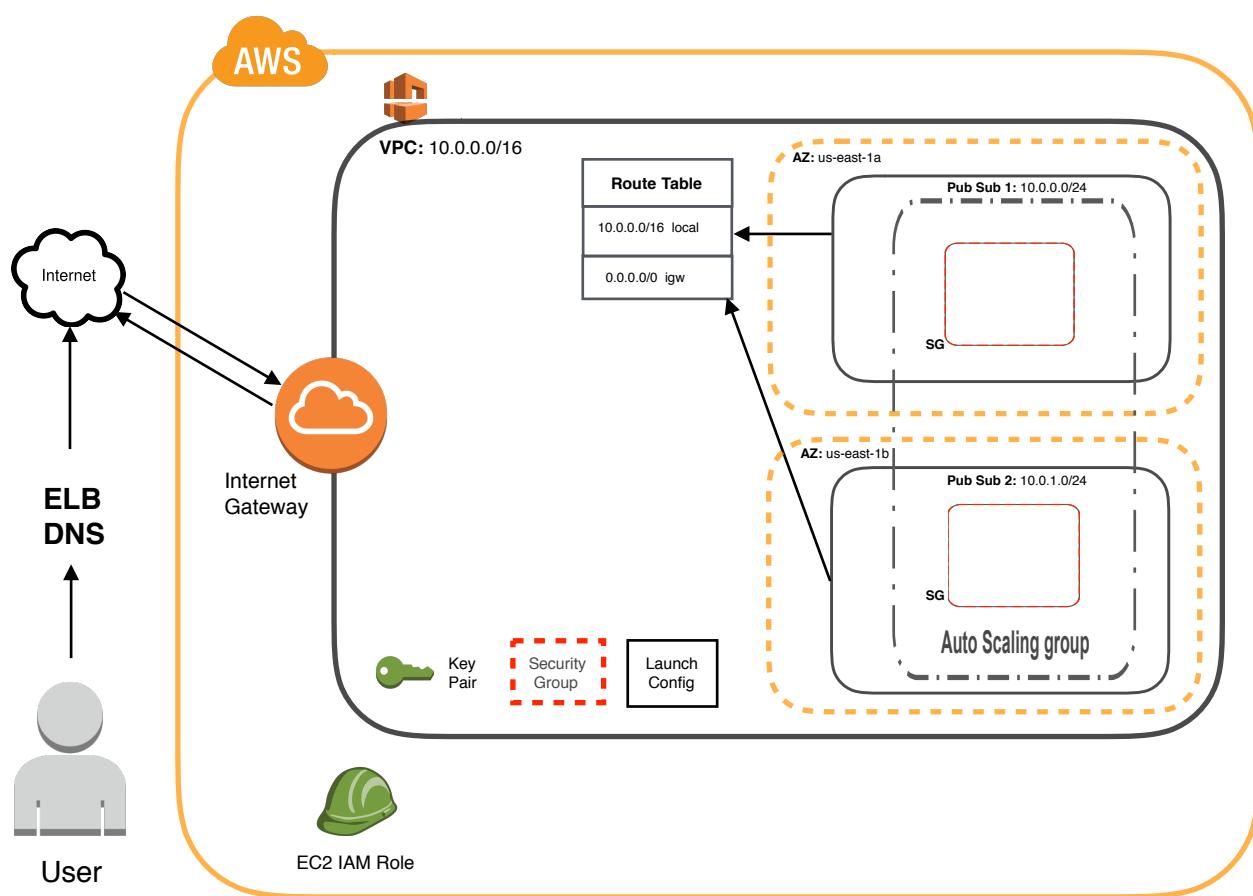
10. A configuration page will load which allows you to customize a number of different aspects of the WebHook integration you have just added.

On this page, a particularly important piece of information is the **Webhook URL**, as highlighted below. This is URL uniquely identifies your Slack channel, allowing external services to post messages and statuses to Slack

The screenshot shows the 'Incoming WebHooks' configuration page again. At the top, it says 'Added by bcutler on April 28th, 2017' with options to 'Disable' or 'Remove'. Below the note about Incoming Webhooks, there's another note about Message Attachments. A yellow warning box titled 'New to Slack integrations?' is present. The main section is titled 'Setup Instructions' with the sub-instruction: 'We'll guide you through the steps necessary to configure an Incoming Webhook so you can start sending data to Slack.' At the bottom, a red box highlights the 'Webhook URL' field, which contains the value '<https://hooks.slack.com/services/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>'. There's also a 'close' button next to the URL field.

11. Open up a blank text document (Notepad, Microsoft Word etc.) and copy the **Webhook URL** to it so that we can use it easily at a later stage of the lab.

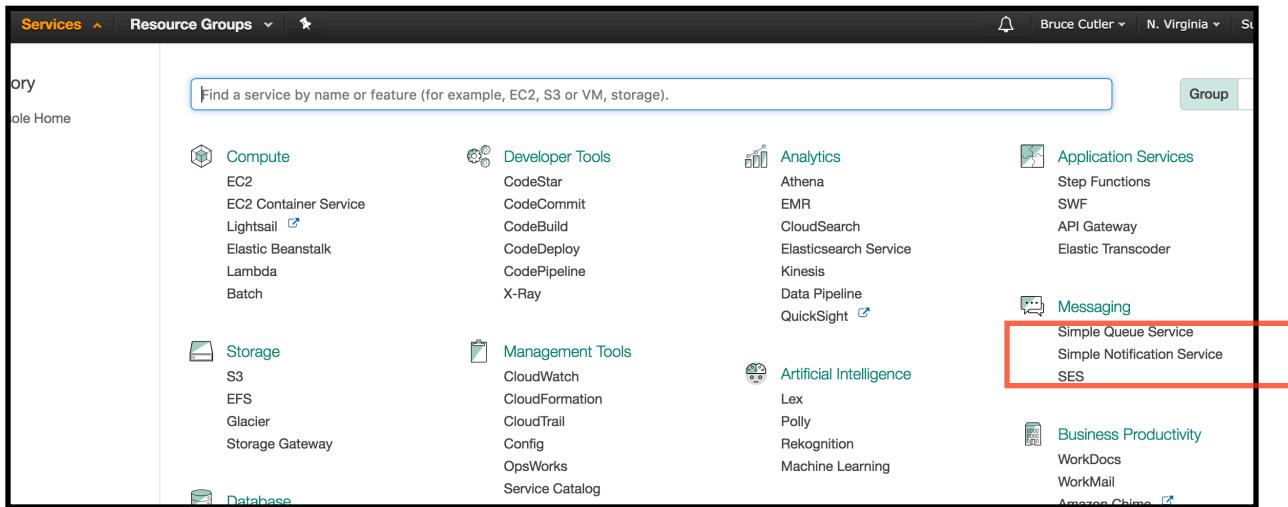
System Architecture: Step 3



Webhook URL: [https://hooks.slack.com/....](https://hooks.slack.com/)

Step 4: Create an SNS Topic

1. Navigate to **Services** → **Simple Notification Service**



2. Click the blue **Get Started** button
3. Select the **Create Topic** link
3. In the Create new topic popup that appears, enter the following and click the blue **Create Topic** button:

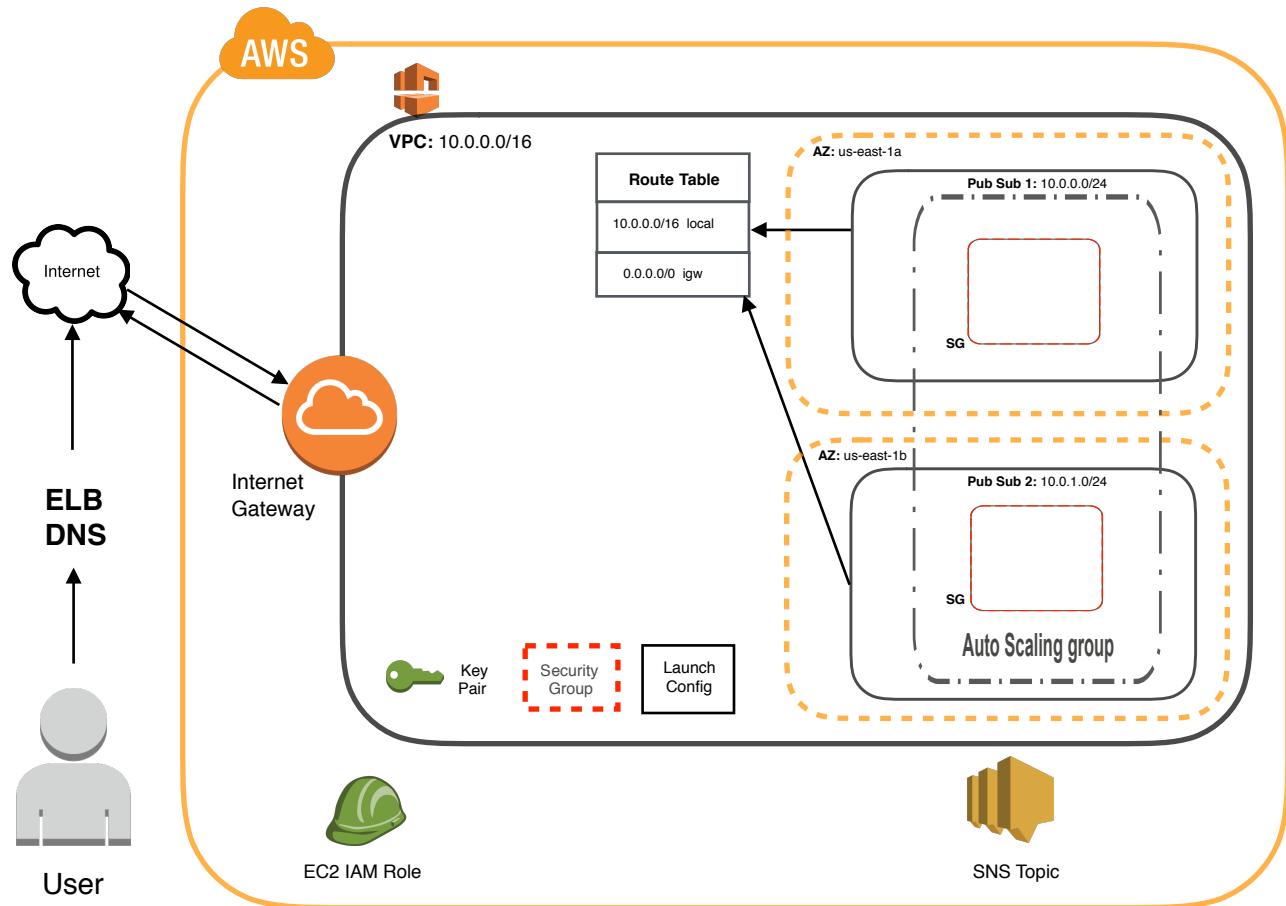
Topic Name: high_cpu_notif

Display name: <Leave as default value>

The dialog box has a title bar 'Create new topic'. The instructions below the title state: 'A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN.)'. There are two input fields: 'Topic name' containing 'high_cpu_notif' and 'Display name' containing '<Leave as default value>'. At the bottom right are 'Cancel' and 'Create topic' buttons, with 'Create topic' being the active button.

4. After the topic has been created, click on **Topics** in the left hand menu. You should see your new SNS Topic, **high_cpu_notif** listed

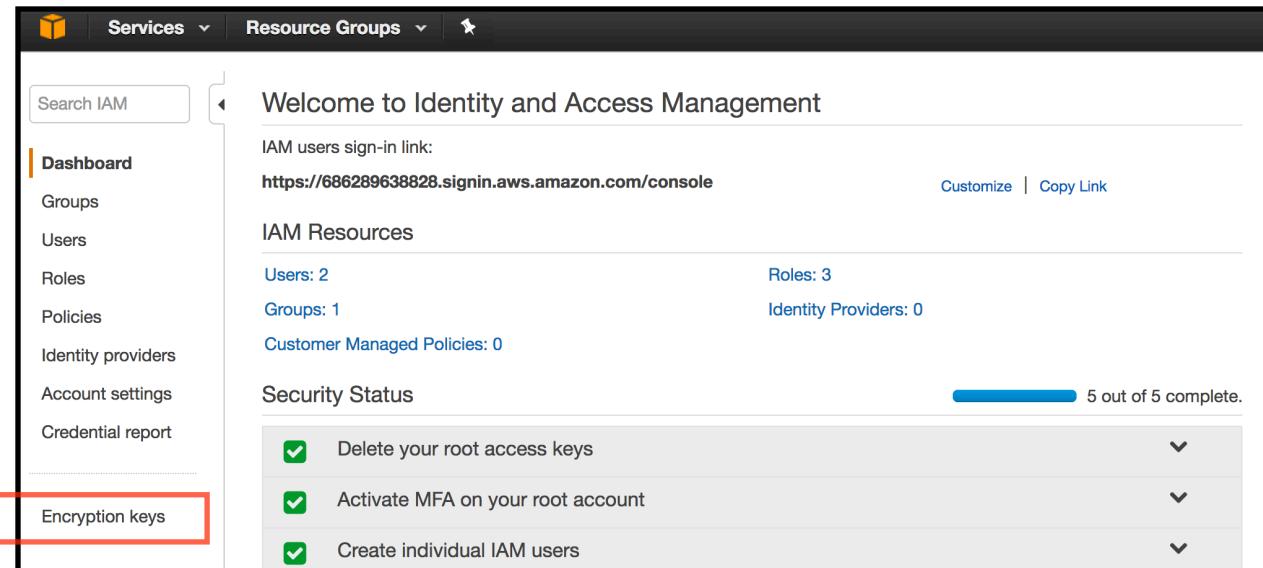
System Architecture: Step 4



Webhook URL: <https://hooks.slack.com/....>

Step 5: Create an encryption key using Amazon Key Management Service (KMS)

1. Navigate to **Services** → **IAM**
2. Select the **Encryption keys** link in the left menu

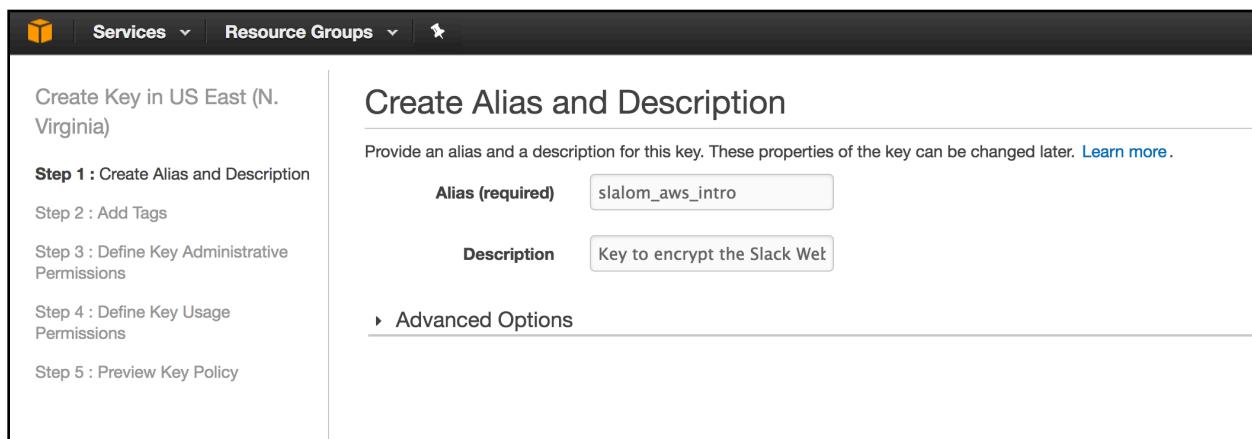


The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, there's a navigation sidebar with links like 'Dashboard', 'Groups', 'Users', 'Roles', 'Policies', 'Identity providers', 'Account settings', 'Credential report', and 'Encryption keys'. The 'Encryption keys' link is highlighted with a red box. The main content area has a heading 'Welcome to Identity and Access Management'. It displays 'IAM users sign-in link' and a URL. Below that is a section for 'IAM Resources' showing 'Users: 2', 'Groups: 1', 'Roles: 3', and 'Identity Providers: 0'. There's also a 'Customer Managed Policies: 0' section. Under 'Security Status', there are three items with checkboxes: 'Delete your root access keys', 'Activate MFA on your root account', and 'Create individual IAM users'. A progress bar indicates '5 out of 5 complete.'

3. Select the blue **Get Started Now** button
4. Select the blue **Create Key** button
5. In **Step 1** of the wizard that appears , enter the following Alias and Description:

Alias: slalom_aws_intro

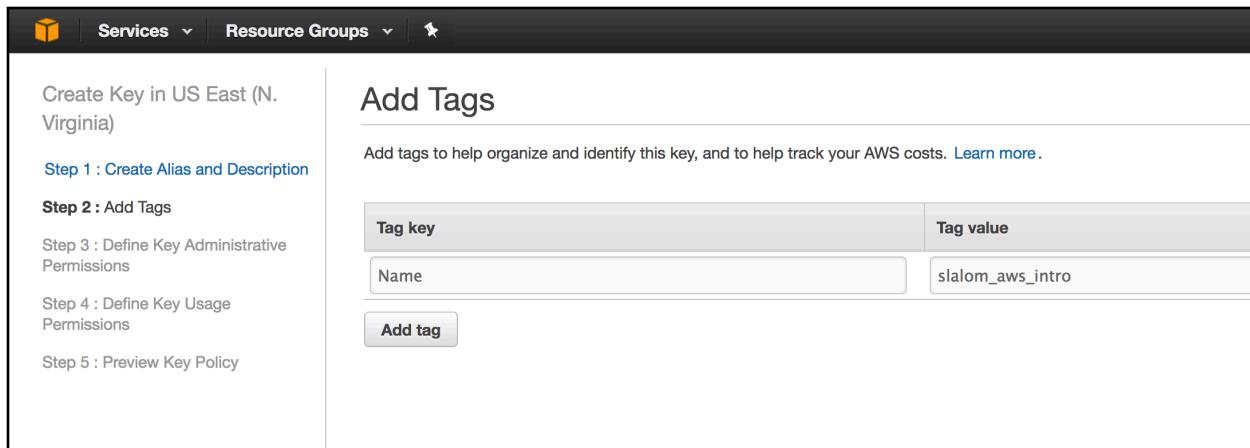
Description: Key to encrypt the Slack Webhook URL



The screenshot shows the 'Create Key in US East (N. Virginia)' wizard. On the left, a sidebar lists steps: 'Step 1 : Create Alias and Description', 'Step 2 : Add Tags', 'Step 3 : Define Key Administrative Permissions', 'Step 4 : Define Key Usage Permissions', and 'Step 5 : Preview Key Policy'. The main area is titled 'Create Alias and Description' with the sub-instruction 'Provide an alias and a description for this key. These properties of the key can be changed later. [Learn more](#)'. It has two input fields: 'Alias (required)' containing 'slalom_aws_intro' and 'Description' containing 'Key to encrypt the Slack Webhook URL'. Below these is a link 'Advanced Options'.

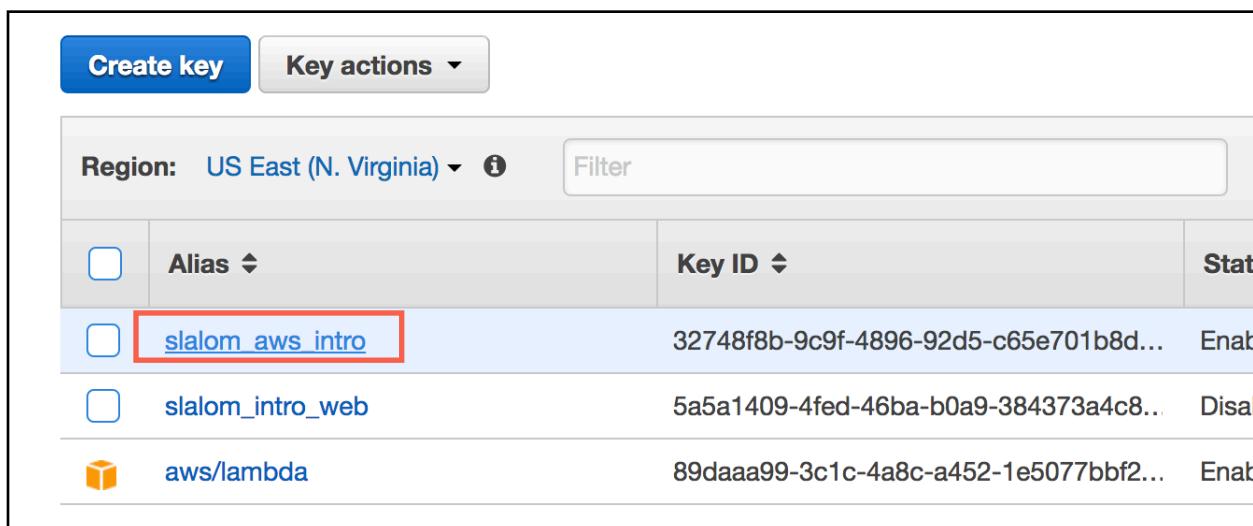
6. In **Step 2**, enter the following tag:

key = Name
value = slalom_aws_intro



The screenshot shows the 'Add Tags' step of a key creation wizard. On the left, a sidebar lists steps: 'Create Key in US East (N. Virginia)', 'Step 1 : Create Alias and Description', 'Step 2 : Add Tags' (which is selected), 'Step 3 : Define Key Administrative Permissions', 'Step 4 : Define Key Usage Permissions', and 'Step 5 : Preview Key Policy'. The main area is titled 'Add Tags' with the sub-instruction 'Add tags to help organize and identify this key, and to help track your AWS costs. [Learn more.](#)'. It contains a table with two columns: 'Tag key' and 'Tag value'. A single row is present with 'Name' in the key column and 'slalom_aws_intro' in the value column. A 'Add tag' button is at the bottom of the table.

7. In **Step 3**, click the blue **Next Step** button
8. In **Step 4**, click the blue **Next Step** button
9. **Step 5** offers you a preview of the Key's attached policy. This simply defines how the key can be used by various users and AWS services. No action is required here, so click the blue **Finish** button to complete key creation.
10. You should see your newly created **slalom_aws_intro** key. Click on the key name in the Alias column



The screenshot shows the 'Keys' section of the AWS IAM console. At the top, there are tabs for 'Create key' and 'Key actions'. Below that, a search bar has 'Region: US East (N. Virginia)' and a 'Filter' input field. The main area is a table with columns: 'Alias' (checkbox), 'Key ID' (checkbox), and 'Status'. Three rows are listed: 'slalom aws intro' (selected, highlighted with a red border), 'slalom_intro_web', and 'aws/lambda'. The 'slalom aws intro' row has its 'Alias' cell highlighted with a red border.

Alias	Key ID	Status
<input type="checkbox"/> slalom aws intro	32748f8b-9c9f-4896-92d5-c65e701b8d...	Enabled
<input type="checkbox"/> slalom_intro_web	5a5a1409-4fed-46ba-b0a9-384373a4c8...	Disabled
<input type="checkbox"/> aws/lambda	89daaa99-3c1c-4a8c-a452-1e5077bbf2...	Enabled

11. In the page that loads, **copy the value in the ARN field** “arn:aws:kms:us-east.....” to the text file that you were asked to create earlier (See Step 2 #11 for reference).

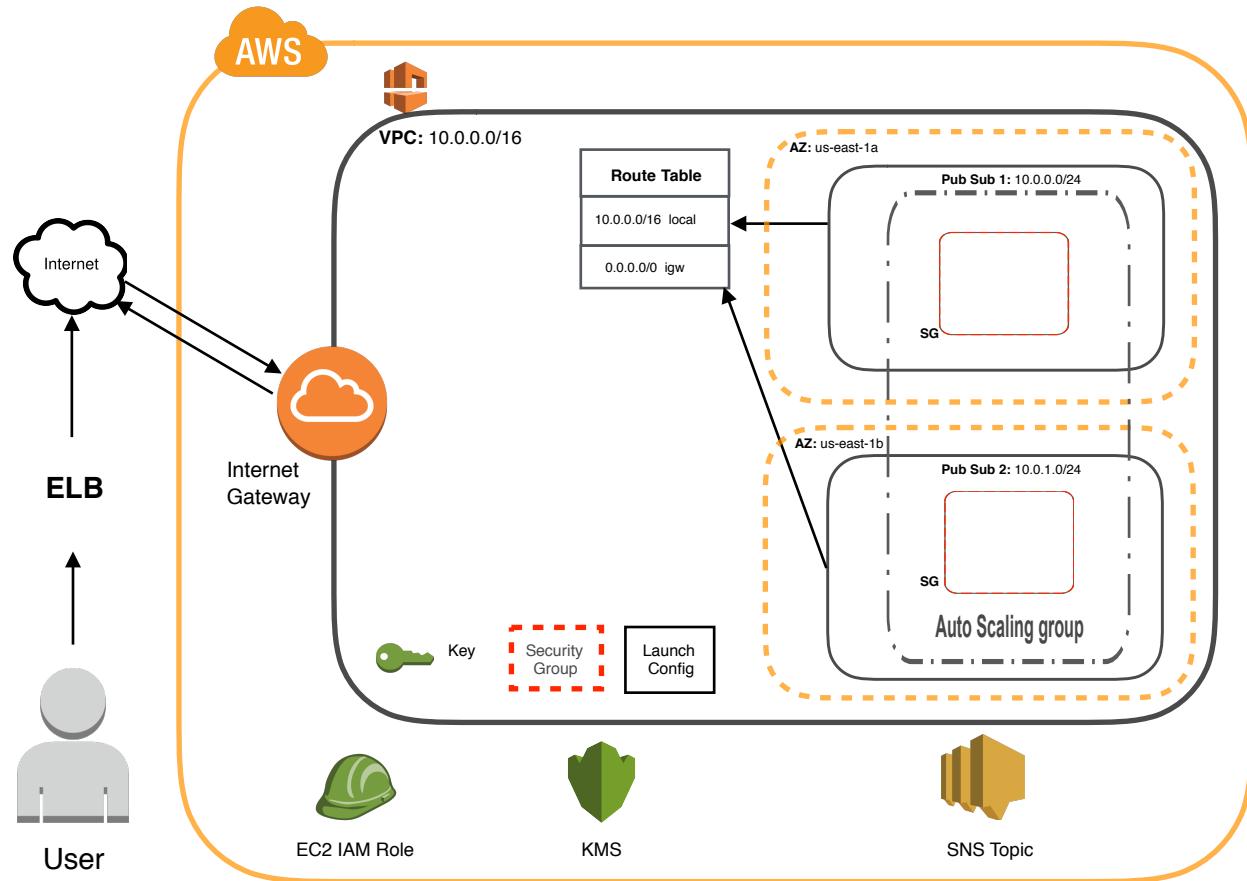
Again, this is an important piece of text that will be used to configure another service later within the lab.

IAM > Encryption Keys > slalom_aws_intro

▼ Summary

Region	us-east-1
ARN	arn:aws:kms:us-east-1:686289638828:key/32748f8b-9c9f-4896-92d5-c65e701b8d31
Status	Enabled
Alias	slalom_aws_intro
Description	[Empty]

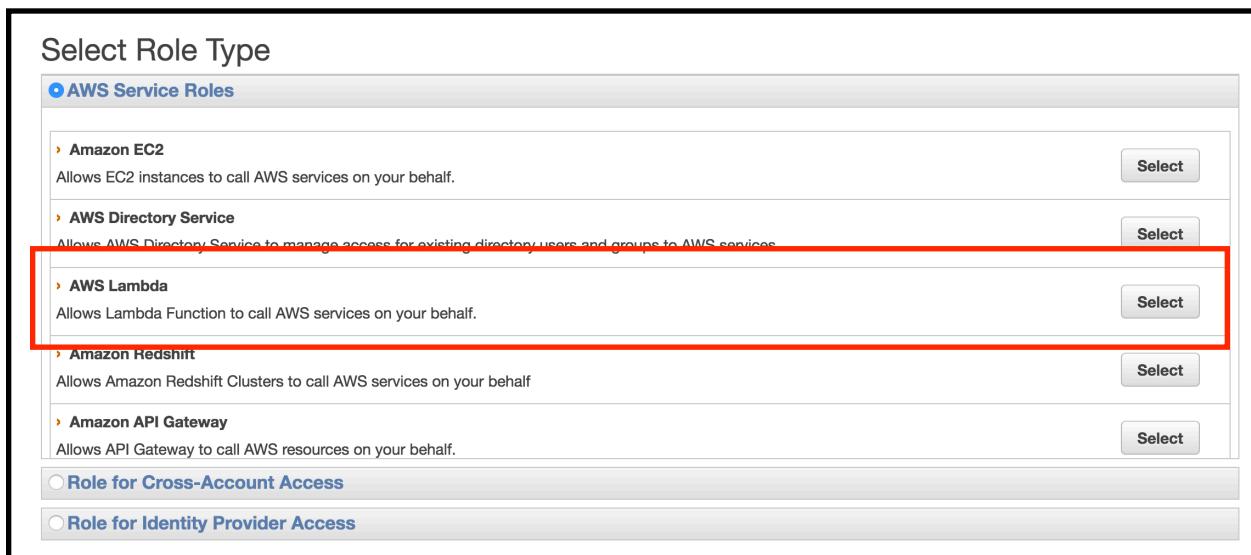
System Architecture: Step 5



Webhook URL: <https://hooks.slack.com/....>

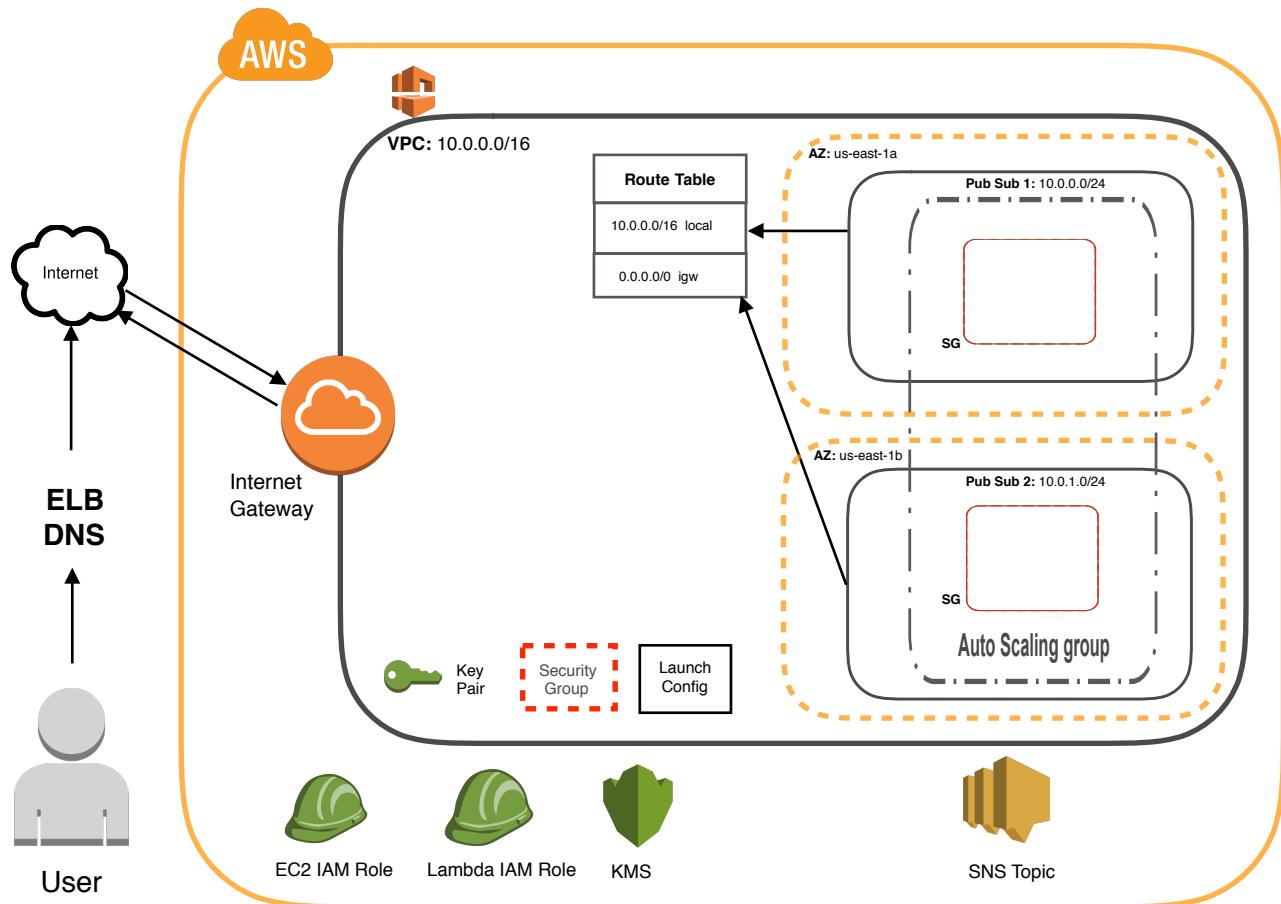
Step 6: Create An IAM Role for Running AWS Lambda Functions

1. Navigate to **Services** → **IAM**
2. In the left menu, select **Roles**. You should see the slalom_web_server role that we created in Lab1
3. Select the blue **Create New Role** button
4. In the Select Role Type screen, click the **Select** button in the AWS Lambda section:



5. In the Attach Policy screen, don't select any policy. We will add this in ourselves in the following Step. Click the blue **Next Step** button:
6. Enter **slalom_lambda_role** as the Role Name and click **Next Step**
7. Click the **Create Role** button to create the **slalom_lambda_role** IAM Role

System Architecture: Step 6



Webhook URL: [https://hooks.slack.com/....](https://hooks.slack.com/)

Step 7: Create and assign an IAM Policy to your new IAM Role

1. Navigate to **Services** —> **IAM**
2. In the left menu, click the **Roles** link
3. Click on the name of your new role, **slalom_lambda_role**

The screenshot shows the AWS IAM service interface. The left sidebar has a highlighted 'Roles' section. The main area displays a list of roles with a search bar and filter options. A red box highlights the 'slalom_lambda_role' in the list.

Role name	Description	Last updated
chef_node		2018-01-12
chef_server		2018-01-12
slalom_lambda_role		2018-01-12
test_server		2018-01-12

4. The following page should open. Click on the **Inline Policies** section as shown:

The screenshot shows the AWS IAM Roles page. The left sidebar has 'Roles' selected. The main content shows a role named 'slalom_lambda_role'. Under the 'Permissions' tab, there is a section for 'Managed Policies' which is empty. Below it is a section for 'Inline Policies' which is also empty. A blue button labeled 'Attach Policy' is visible above the 'Inline Policies' section. A red box highlights the 'Inline Policies' section, and a red arrow points from the text in the next step towards this box.

5. The message lets us know that we don't currently have any inline policies attached to the new Role you created in Step 6. Let's use the **click here** link to create one.

The screenshot shows a page titled 'Inline Policies'. It displays the message: 'There are no inline policies to show. To create one, [click here](#)'. A red arrow points from the text in the previous step towards this 'click here' link.

6. Keep the **Policy Generator** selected and click the grey **Select** button
7. In the next page, ensure the following are set:

Effect: Allow
AWS Service: AWS Key Management Service
Actions: Decrypt
Amazon Resource Name: < Paste in the ARN for your Key from Step 5 #11 >

Edit Permissions

The policy generator enables you to create policies that control access to Amazon Web Services (AWS) products and see [Overview of Policies](#) in Using AWS Identity and Access Management.

Effect Allow Deny

AWS Service AWS Key Management Service

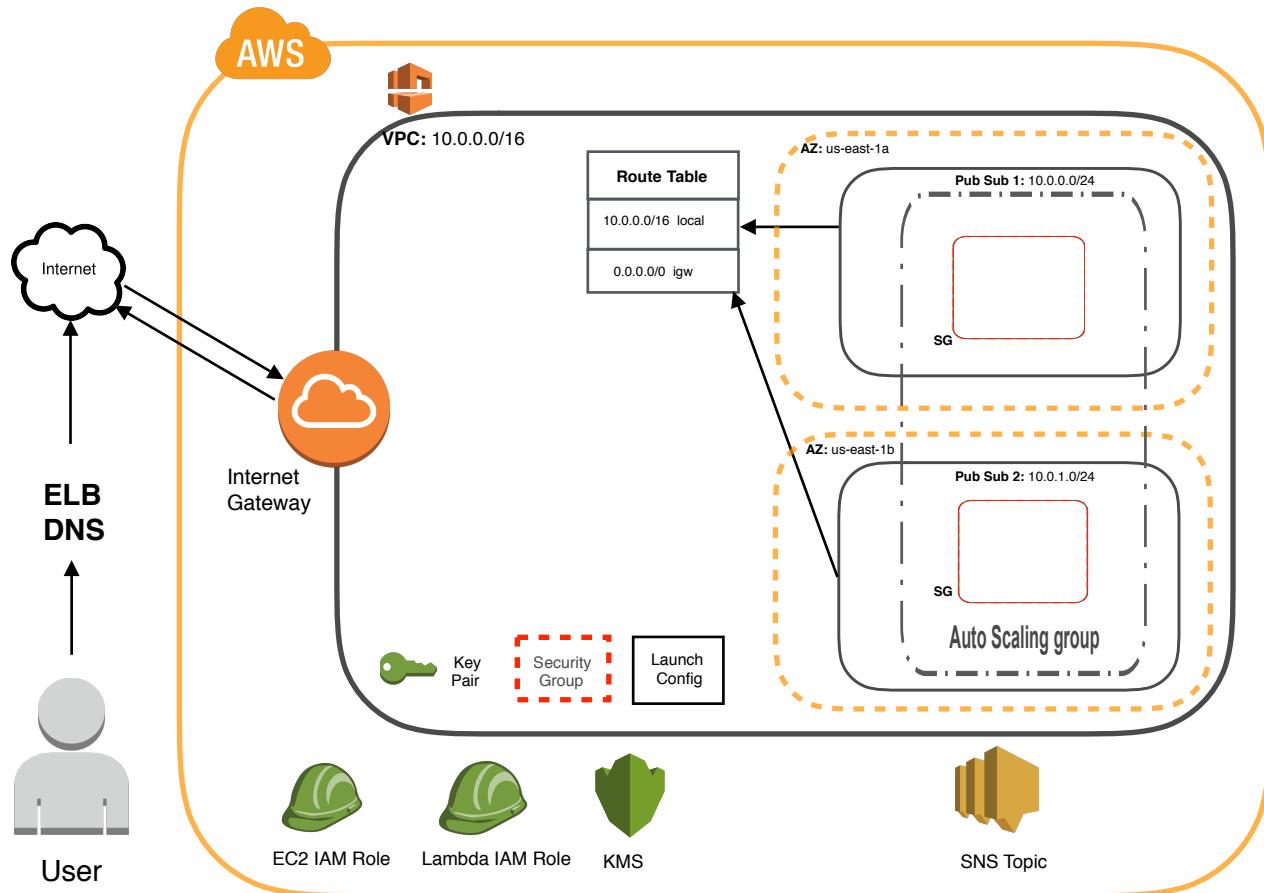
Actions 1 Action(s) Selected

Amazon Resource Name (ARN) arn:aws:kms:us-east-1:6862896

[Add Conditions \(optional\)](#)

[Add Statement](#)
8. Once the above have been added, click the grey **Add Statement** button
9. Click the blue **Next Step** button
10. Click the blue **Apply Policy** button. This will add your new custom policy to the IAM Role

System Architecture: Step 7



Webhook URL: [https://hooks.slack.com/....](https://hooks.slack.com/)

Step 8: Update your Auto Scaling Group Instance Count

1. Navigate to **Services** → **EC2**
2. In the left menu, select **Auto Scaling Groups**
3. Select your **slalom_aws_intro_web_asg** group
4. In the box at the bottom, select the grey **Edit** button

The screenshot shows the AWS Auto Scaling Groups page. At the top, there is a table listing five Auto Scaling Groups:

Auto Scaling Group	Launch Configuration	Desired	Min	Max	Health Check Type	Last Check Time
slalom_aws_intro_web_asg	Dev-web-LaunchConfig-SD88HE5HPRO0	0	0	2	CloudWatch Metrics	2023-09-12 13:45:00 UTC
slalom-simple-asg	slalom-simple-lc	0	0	1	CloudWatch Metrics	2023-09-12 13:45:00 UTC
slalom-interval-asg	slalom-interval-lc	0	0	1	CloudWatch Metrics	2023-09-12 13:45:00 UTC
slalom-stg-asg	slalom_stg_lc	0	0	1	CloudWatch Metrics	2023-09-12 13:45:00 UTC
slalom-dev-asg	slalom-dev-lc	0	0	1	CloudWatch Metrics	2023-09-12 13:45:00 UTC

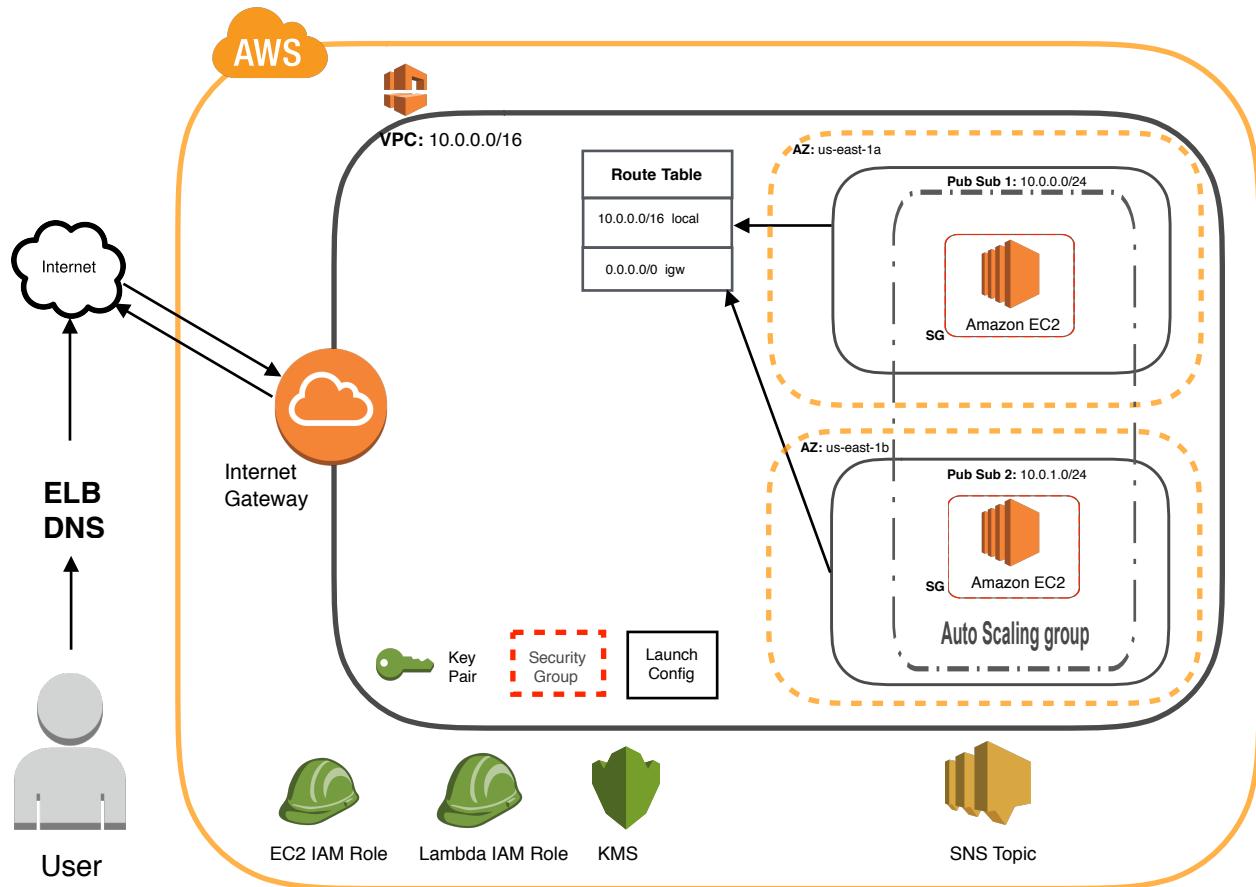
Below the table, there is a navigation bar with tabs: Details (highlighted in orange), Activity History, Scaling Policies, Instances, Monitoring, Notifications, Tags, and Scheduled Actions. To the right of the navigation bar is a grey 'Edit' button, which is also highlighted with a red box.

Under the 'Details' tab, the configuration details for the selected Auto Scaling Group are shown:

- Launch Configuration:** Dev-web-LaunchConfig-SD88HE5HPRO0
- Load Balancers:** slalom-aws-intro-web-elb
- Target Groups:**
- Desired:** 0
- Min:** 0
- Availability Zone(s):** us-east-1a, us-east-1b
- Subnet(s):** subnet-bb7d5cde, subnet-ccde3696

5. Change the value of **Desired** and **Min** to 2. This will create two EC2 instances within your ASG. We need active EC2 instances in our ASG to allow us to create our CloudWatch alarm in Step 8.
6. Click the blue **Save** button

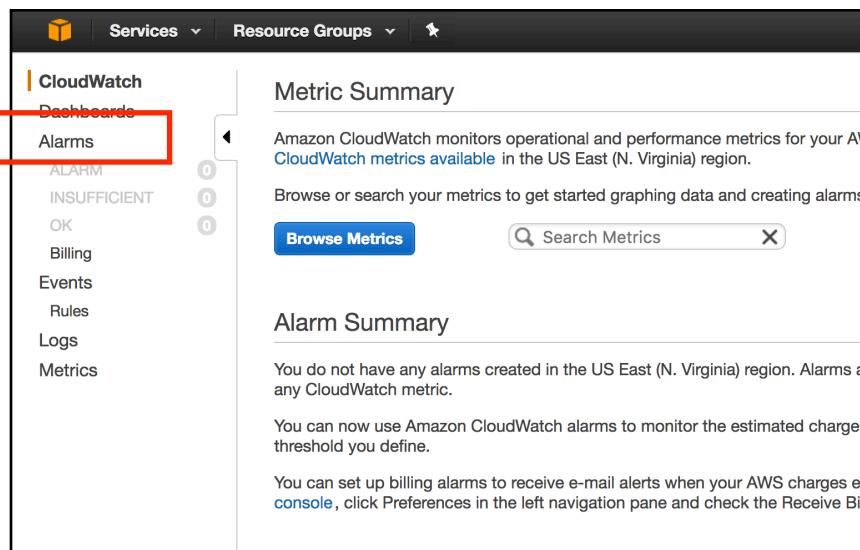
System Architecture: Step 8



Webhook URL: [https://hooks.slack.com/....](https://hooks.slack.com/)

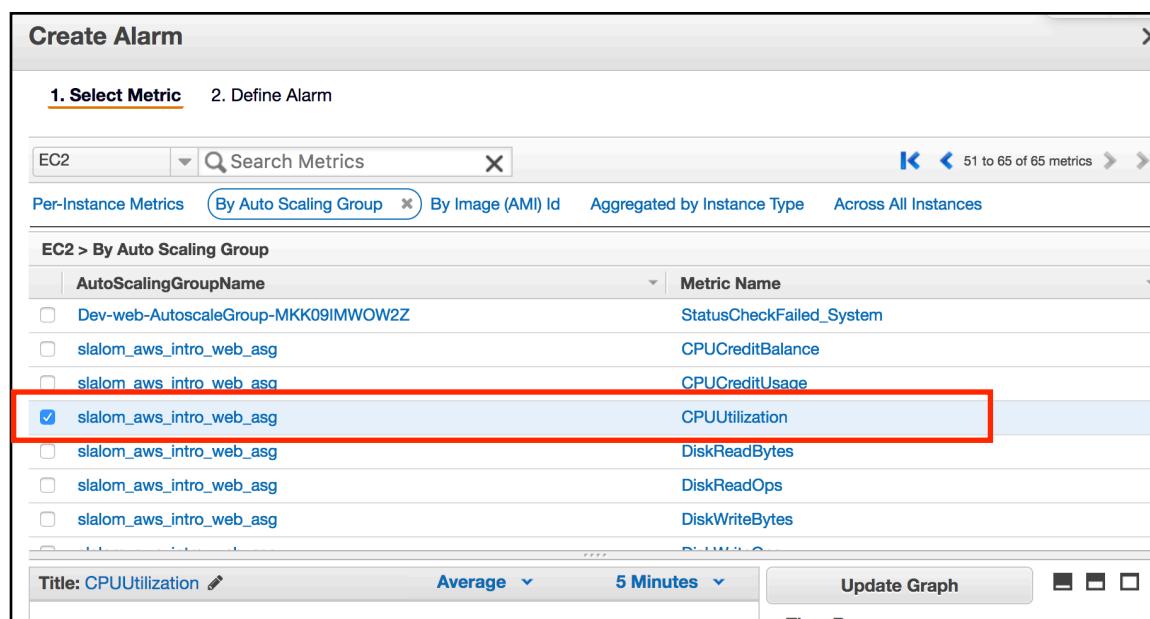
Step 9: Create an AWS CloudWatch Alarm

1. Navigate to **Services** → **CloudWatch**
2. In the left menu, select **Alarms**



The screenshot shows the AWS CloudWatch Metrics Summary page. On the left, there's a navigation pane with links like 'CloudWatch', 'Dashboards', 'Alarms' (which is highlighted with a red box), 'ALARM', 'INSUFFICIENT', 'OK', 'Billing', 'Events', 'Rules', 'Logs', and 'Metrics'. The main content area has two sections: 'Metric Summary' and 'Alarm Summary'. The 'Metric Summary' section contains text about monitoring operational and performance metrics for the US East (N. Virginia) region, a 'Browse Metrics' button, and a search bar. The 'Alarm Summary' section states that no alarms are created yet and provides instructions for setting up alarms to monitor charges.

3. Select the blue **Create Alarm** button
4. In the following screen, select the **By Auto Scaling Group** option
5. In the box that opens, select the option with **slalom_aws_intro_web_asg** in the AutoScalingGroupName column and **CPUUtilization** in the Metric Name column



The screenshot shows the 'Create Alarm' wizard, step 1: Select Metric. It has tabs for '1. Select Metric' (which is active) and '2. Define Alarm'. Below the tabs is a search bar and a dropdown for 'EC2'. Underneath are filter buttons: 'Per-Instance Metrics' (selected), 'By Auto Scaling Group' (selected), 'By Image (AMI) Id', 'Aggregated by Instance Type', and 'Across All Instances'. The main area lists metrics under 'EC2 > By Auto Scaling Group'. A table shows 'AutoScalingGroupName' and 'Metric Name'. One row for 'slalom_aws_intro_web_asg' with 'CPUUtilization' is selected and highlighted with a red box. Other rows include 'StatusCheckFailed_System', 'CPUCreditBalance', 'CPUCreditUsage', 'DiskReadBytes', 'DiskReadOps', 'DiskWriteBytes', and 'DiskWriteOps'. At the bottom are buttons for 'Title: CPUUtilization', 'Average', '5 Minutes', 'Update Graph', and a 'Time Range' dropdown.

- Click the blue **Next** button
- On the following screen for Alarm Threshold, ensure the following information is configured to match the image below:

Name: high_cpu_alarm

Whenever: CPU Utilization is ≥ 2 for 1 consecutive period

Treat missing data as: good (not breaching threshold)

Send notification to: high_cpu_notif

Period: 1 minute

NOTE: This alarm is configured with an unrealistically low threshold of 2%, but it will allow us to demonstrate this feature

Modify Alarm

[1. Select Metric](#) [2. Define Alarm](#)

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name: slalom_aws_intro_web	Description:
Whenever: CPUUtilization is: \geq 2 for: 1 consecutive period(s)	

Additional settings

Provide additional configuration for your alarm.

Treat missing data as: good (not breaching threshold)
--

Actions

Define what actions are taken when your alarm changes state.

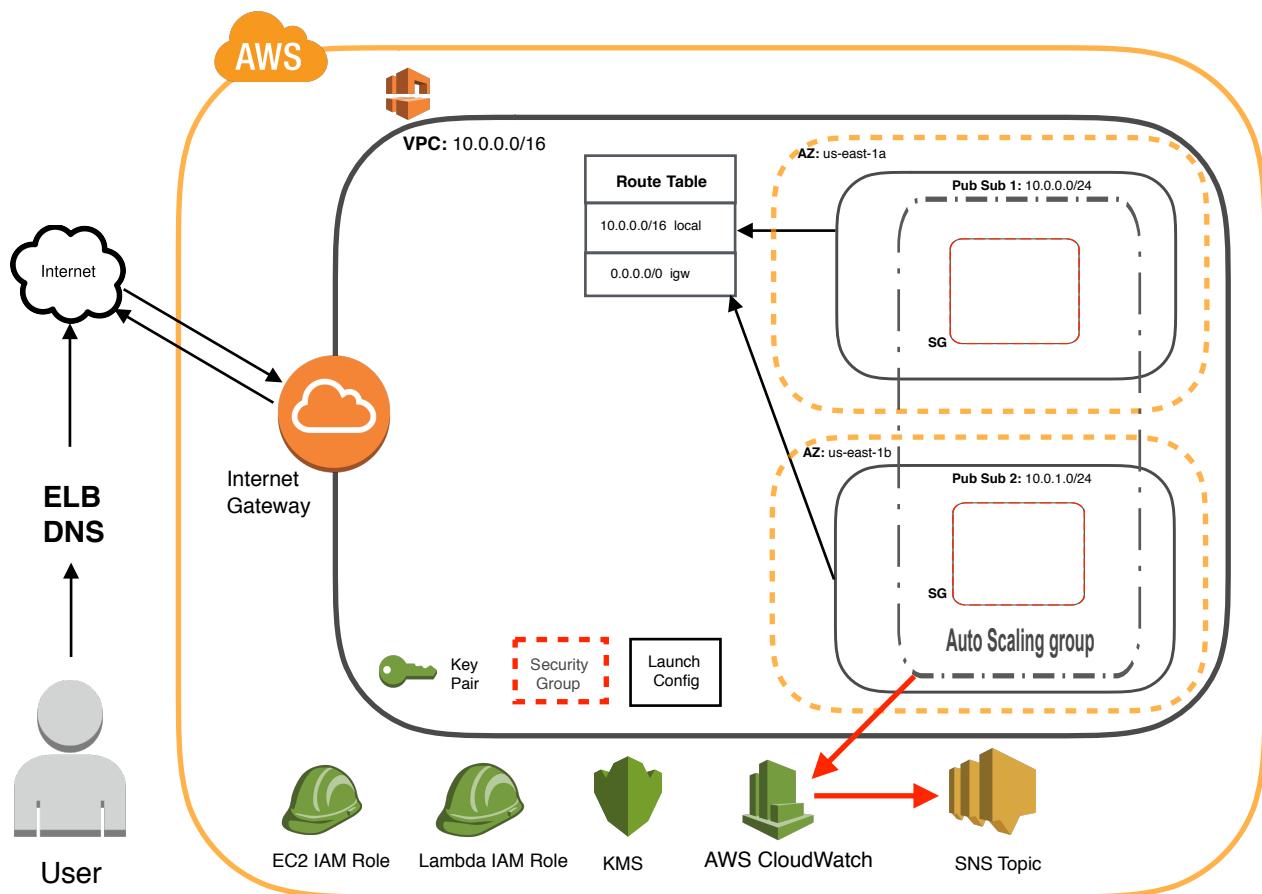
Notification Delete Whenever this alarm: State is ALARM Send notification to: high_cpu_notif New list Enter list
--

This notification list is managed in the SNS console.

[Cancel](#) [Previous](#) [Next](#) **Create Alarm**

8. Click the blue **Create Alarm** button
9. Navigate back to **Services —> EC2**
10. Click on **Auto Scaling Groups** in the left menu
11. Select your **slalom_aws_intro_web_asg** group in the list and click the grey **Edit** button at the bottom
12. Change the **Desired** and **Min** values back to 0
13. Click the blue **Save** button

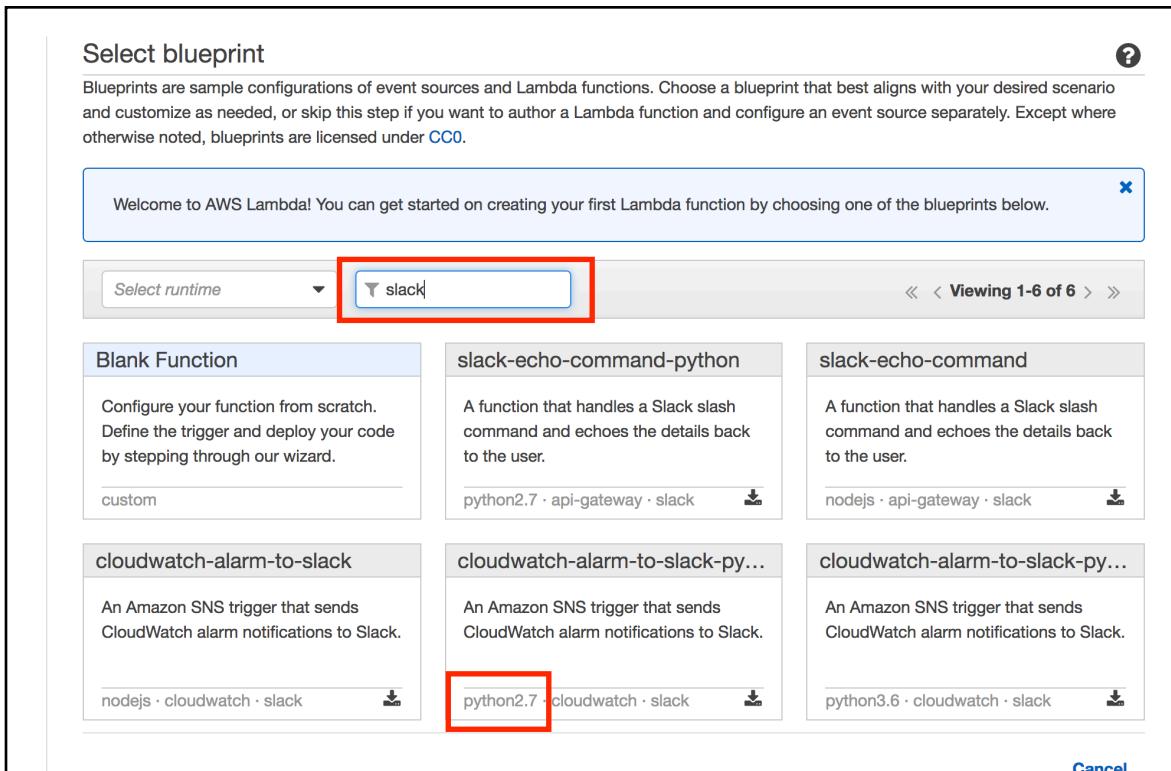
System Architecture: Step 9



Webhook URL: [https://hooks.slack.com/....](https://hooks.slack.com/)

Step 10: Create an AWS Lambda function to post to Slack

1. Navigate to **Services** → **Lambda**
2. Click the blue **Get Started Now** button
3. In the filter box, type in **slack**
3. Select the **cloudwatch-alarm-to-slack-py... for python 2.7** option

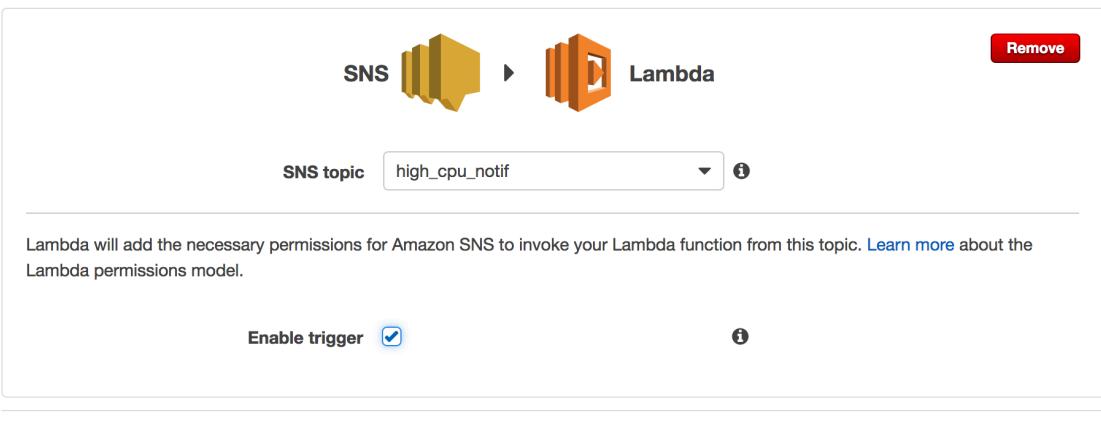


4. In the following page, ensure that the following is set:

SNS topic: high_cpu_notif
Enable Trigger: selected

Configure triggers

You can choose to add a trigger that will invoke your function.



SNS  →  Lambda

SNS topic

Lambda will add the necessary permissions for Amazon SNS to invoke your Lambda function from this topic. [Learn more](#) about the Lambda permissions model.

Enable trigger

[Cancel](#) [Previous](#) [Next](#)

5. Click the blue **Next** button
6. At the top of the Configure function page, set the **Name** to **high_cpu_slack_function**

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name*	high_cpu_slack_function
Description	An Amazon SNS trigger that sends CloudW
Runtime*	Python 2.7

Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

7. Scroll down the page to below the function code. You should see an option for **Enable encryption helpers**. Click the box to **select this option**
8. With the Enable encryption headers checkbox checked, enter the following information below.

NOTE: Remember that your slack WebHook URL should be within the text file you created earlier (if you forgot to put your WebHook URL in a text file, ask Bruce and he will show you how to find it again):

Encryption key: slalom_aws_intro

slackChannel: < Your private slack channel name > (don't include the hashtag)

kmsEncryptedHookURL: < The webhook URL you created in Step 2. **Do not include the https:// at the start >**

The screenshot shows the AWS Lambda function configuration page. At the top, there is a checkbox labeled "Enable encryption helpers" which is checked. Below it, an "Encryption key" dropdown is set to "slalom_aws_intro". Under "Environment variables", there are two entries: "slackChannel" with value "brucedemochannel" and "kmsEncryptedHookUrl" with value "hooks.slack.com/services/T*". Each entry has an "Encrypt" button and a "Code" button. A red arrow points from the text "Once you have the above information entered, only click the Encrypt button for the kmsEncryptedHookUrl field. This should give you the following:" to the "Encrypt" button for the "kmsEncryptedHookUrl" entry.

slackChannel	brucedemochannel	Encrypt	Code	X
kmsEncryptedHookUrl	hooks.slack.com/services/T*	Encrypt	Code	X
Key	Value	Encrypt	Code	X

9. Once you have the above information entered, **only** click the **Encrypt** button for the **kmsEncryptedHookUrl** field. This should give you the following:

The screenshot shows the AWS Lambda function configuration page after the "Encrypt" button for the "kmsEncryptedHookUrl" field has been clicked. The "kmsEncryptedHookUrl" entry now shows "....." in the "Value" field, indicating it has been encrypted. The "Encrypt" and "Code" buttons are still present, but the "Decrypt" button is now visible next to the "Code" button. A red arrow points from the text "Once you have the above information entered, only click the Encrypt button for the kmsEncryptedHookUrl field. This should give you the following:" to the "Decrypt" button for the "kmsEncryptedHookUrl" entry.

slackChannel	brucedemochannel	Encrypt	Code	X
kmsEncryptedHookUrl	Decrypt	Code	X
Key	Value	Encrypt	Code	X

10. Scroll down further to the **Lambda function handler and role** section

11. Within this section, ensure the following are set:

Role: Choose an existing role

Existing Role: slalom_lambda_role

Lambda function handler and role

Handler*	lambda_function.lambda_handler	i
Role*	Choose an existing role	▼ i
Existing role*	slalom_lambda_role	▼ i

12. Click the blue **Next** button at the bottom right of the page

13. Review the summary page, checking that the options match the following (other than channel name, which will be set to your own channel):

Review

Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click **Create function** to complete the setup process.

Triggers

SNS	SNS topic: arn:aws:sns:us-east-1:686289638828:high_cpu_notif	Enabled
-----	--	---------

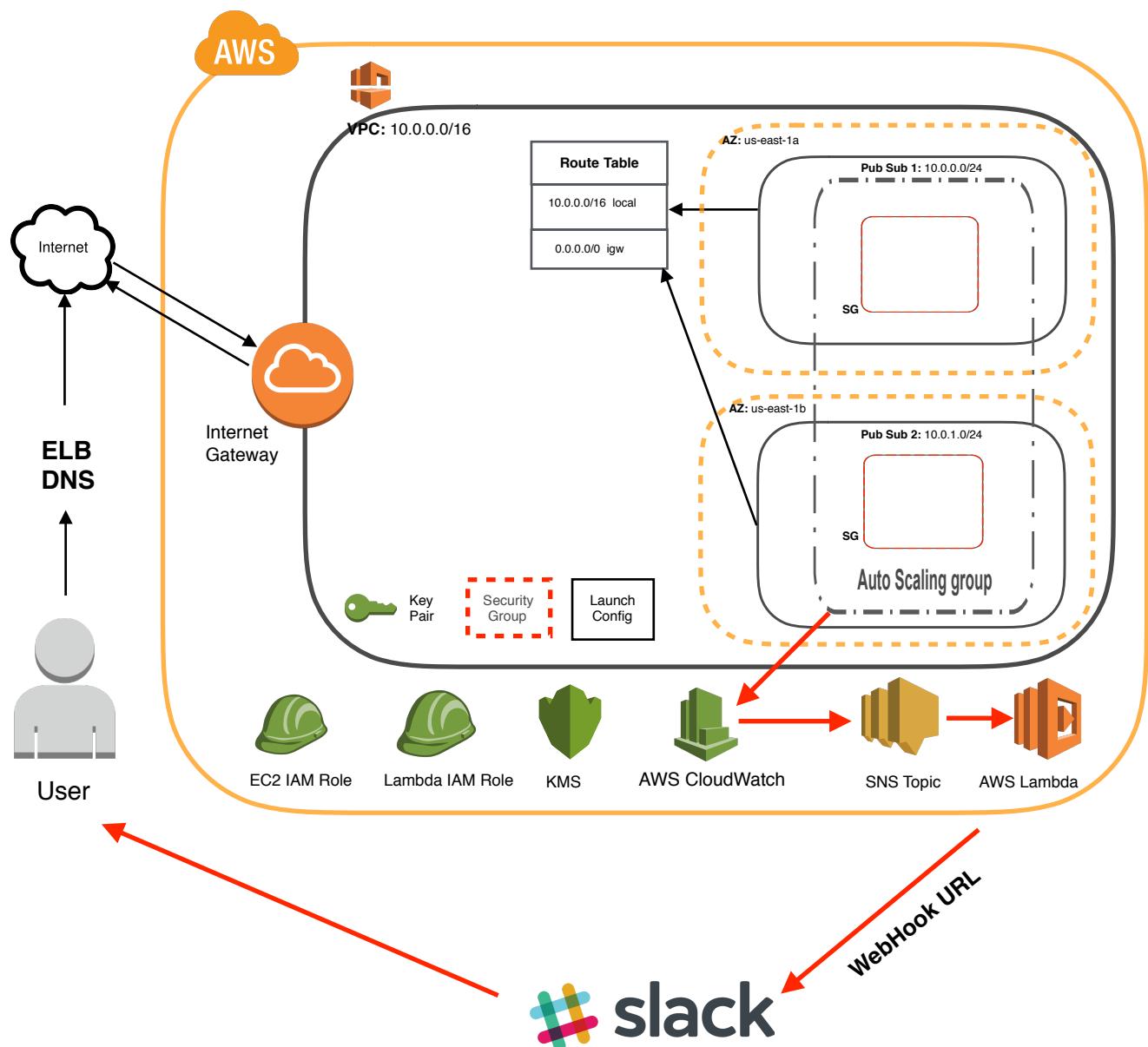
Lambda function

Name	high_cpu_slack_function	Edit
Description	An Amazon SNS trigger that sends CloudWatch alarm notifications to Slack.	
Runtime	Python 2.7	
Environment variables	slackChannel kmsEncryptedHookUrl	brucedemochannel *****
Handler	lambda_function.lambda_handler	
Existing role*	slalom_lambda_role	
Tags		

14. Click the blue **Create function** button

15. You should see a success page, letting you know that the function was created

System Architecture: Step 10

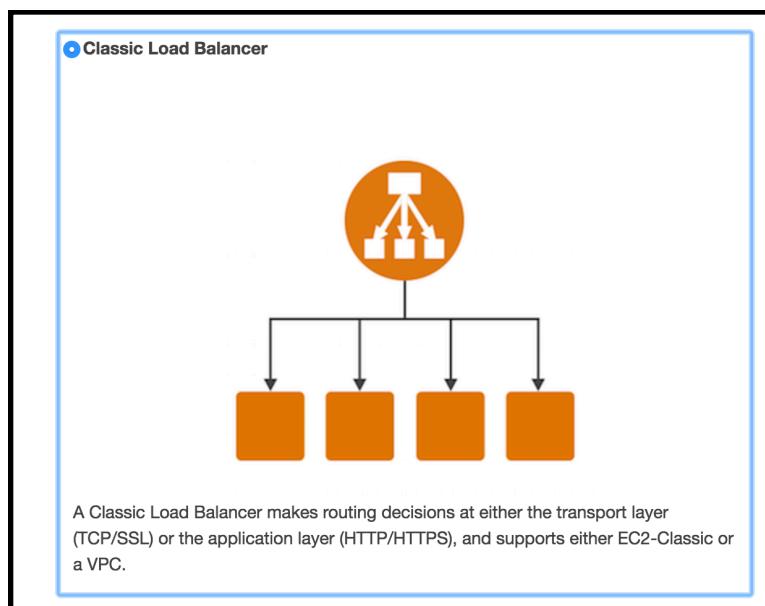


Step 11: Create a Load Balancer

1. Navigate to **Services** → **EC2**
2. In the left menu, select **Load Balancers**

The screenshot shows the AWS EC2 dashboard. On the left, there's a navigation sidebar with several categories: 'Bundle Tasks', 'ELASTIC BLOCK STORE' (with 'Volumes' and 'Snapshots'), 'NETWORK & SECURITY' (with 'Security Groups', 'Elastic IPs', 'Placement Groups', 'Key Pairs', and 'Network Interfaces'), 'LOAD BALANCING' (with 'Load Balancers' highlighted by a red box), 'TARGET GROUPS' (under 'LOAD BALANCING'), 'AUTO SCALING' (with 'Launch Configurations' and 'Auto Scaling Groups'), and 'SYSTEMS MANAGER SERVICES' (with 'Run Command' and 'State Manager'). The main content area is titled 'Resources' and displays statistics for the US East (N. Virginia) region: 0 Running Instances, 0 Dedicated Hosts, 2 Volumes, 2 Key Pairs, 0 Placement Groups, 0 Elastic IPs, 2 Snapshots, 0 Load Balancers, and 7 Security Groups. Below this, a callout box suggests trying Amazon Lightsail for free. At the bottom, there are sections for 'Create Instance' (with a 'Launch Instance' button), 'Service Health', 'Scheduled Events', and 'Service Status: US East (N. Virginia)'.

3. Select the blue **Create Load Balancer** button
4. In the following screen, select the **Classic Load Balancer** option and click the blue **Continue** button



5. In the Define Load Balancer page, enter the following information:

Load Balancer Name: slalom-aws-intro-web-elb
Create LB Inside: slalom-aws-intro

6. In the Select Subnets section at the bottom, click the + symbols beside both **slalom-aws-intro-pub1** and **slalom-aws-intro-pub2**

This should leave you with the following setup:

Step 1: Define Load Balancer

Load Balancer name: slalom-aws-intro-web-elb
Create LB Inside: vpc-59e4c43f (10.1.0.0/16) | slalom-aws-intro
Create an internal load balancer: (what's this?)
Enable advanced VPC configuration:

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

Add

Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Zones to provide higher availability for your load balancer.

VPC vpc-59e4c43f (10.1.0.0/16) | slalom-aws-intro

Available subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name

Selected subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
-	us-east-1a	subnet-f5c1faae	10.1.0.0/24	slalom-aws-intro-pub1
-	us-east-1b	subnet-518bca34	10.1.1.0/24	slalom-aws-intro-pub2

7. Click the **Next: Assign Security Groups** button at the bottom right
8. Use the **Select an existing security group** option and select **slalom_aws_intro_webserver** from the list of security groups
9. Click the **Next: Configure Security Settings** button
10. A warning message will appear, informing us that we can improve the load balancers security by using a secure listener. For the purposes of this workshop we will be using HTTP instead of HTTPS, so we can continue by clicking **Next: Configure Health Check**
11. We can leave the configured health check as it is, so click **Next: Add EC2 Instances**
12. We don't have any EC2 instances running currently, so continue to **Next: Add Tags**

13. Add a tag with the following values:

Key = Name

Value = slalom_aws_intro_web_elb

Step 6: Add Tags

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tags.

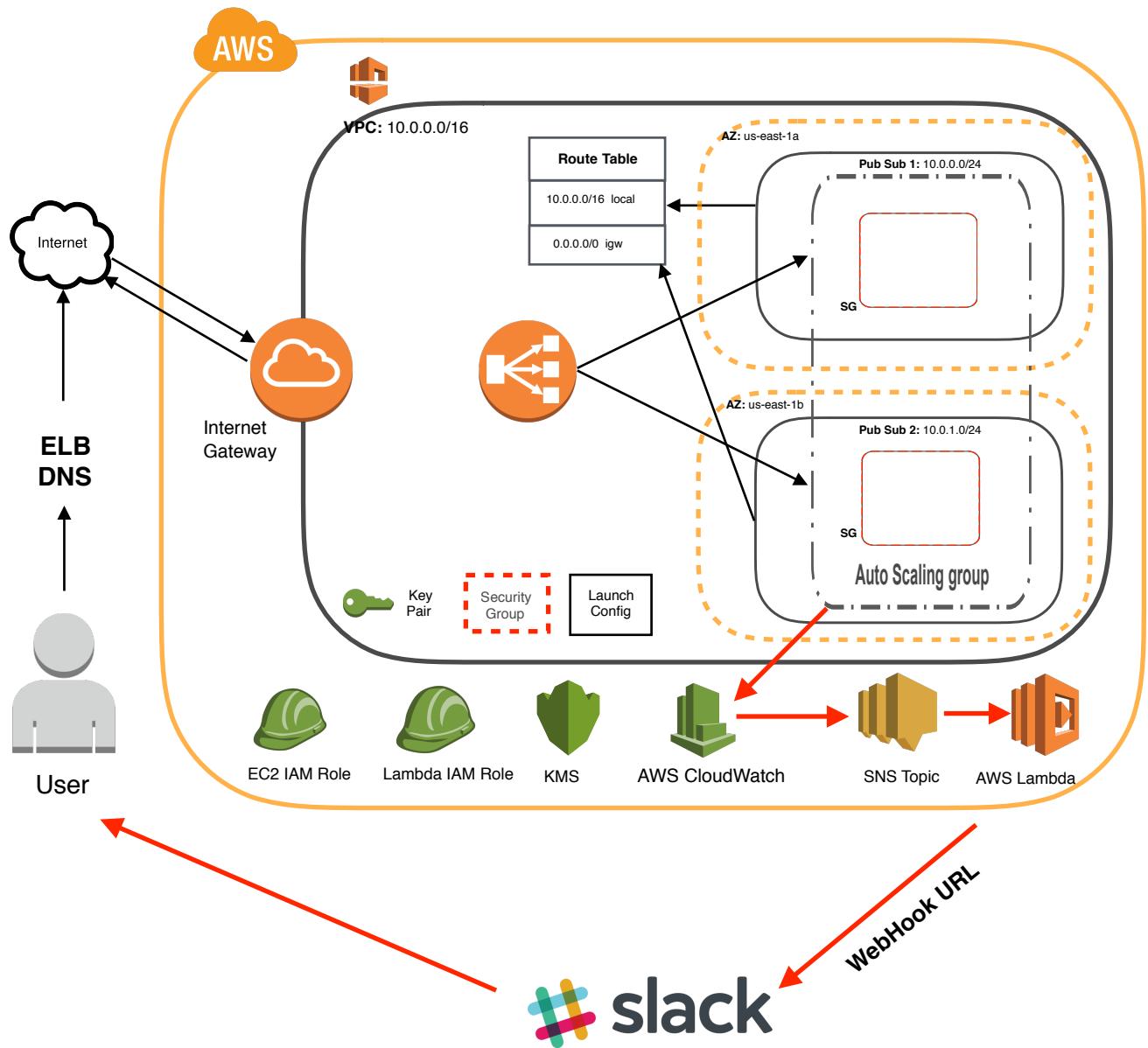
Key	Value
Name	slalom_aws_intro_web_elb

Create Tag

14. Click the blue **Review and Create** button at the bottom right

15. Click the blue **Create** button at the bottom right

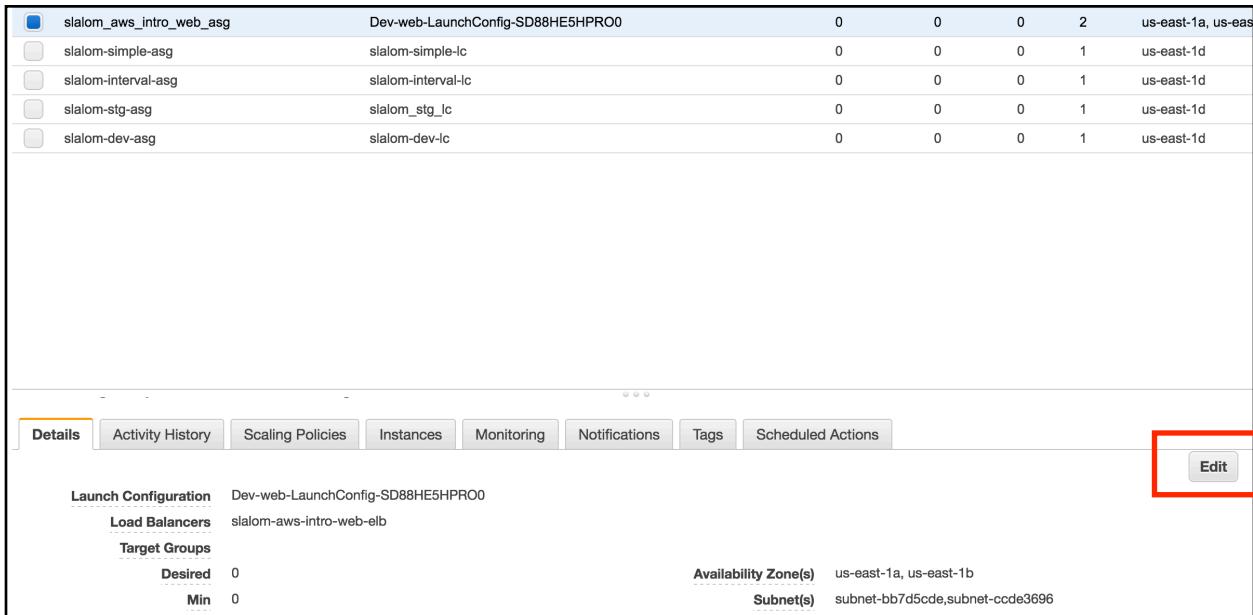
System Architecture: Step 11



Step 12: Update your Auto Scaling Group Instance Count

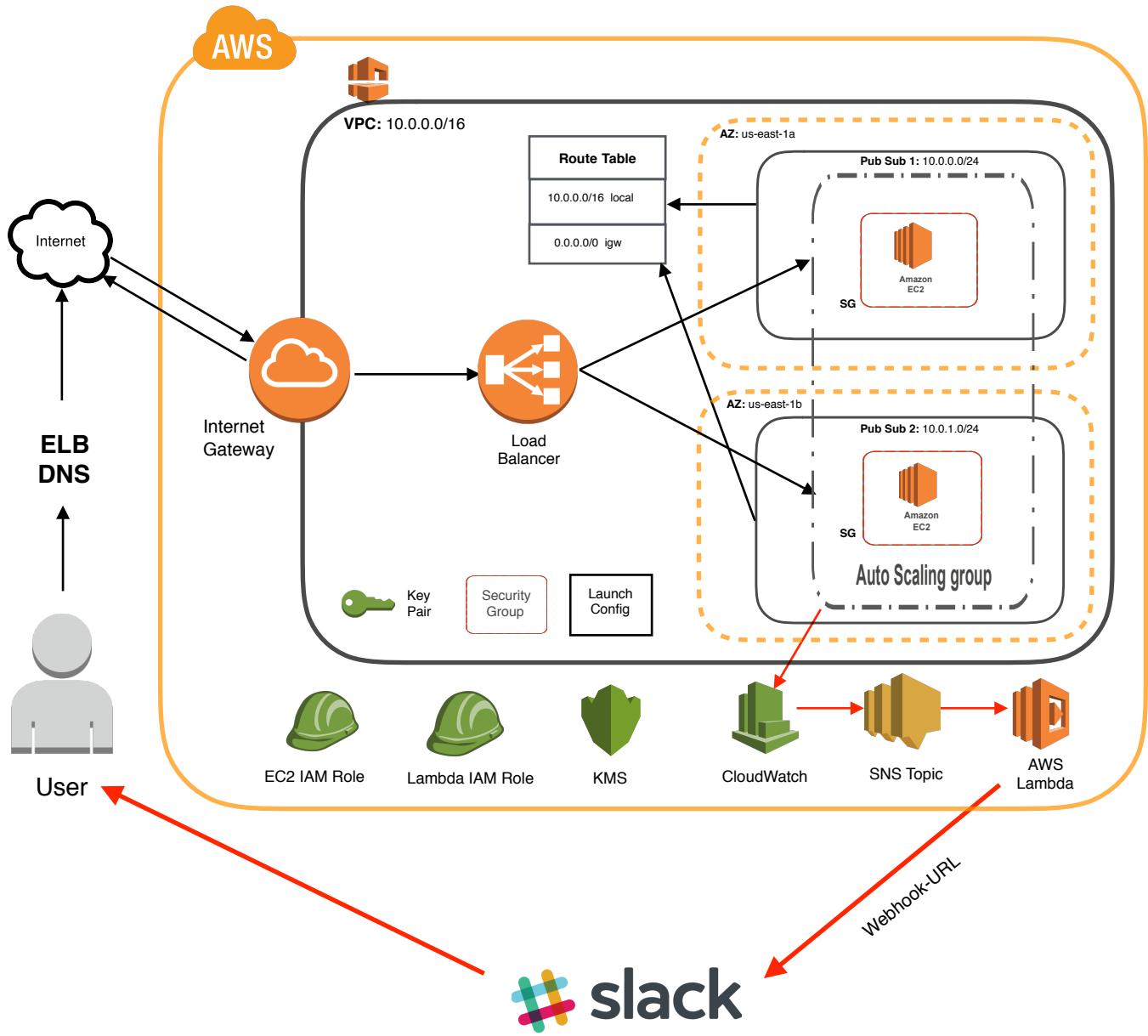
We now have all the pieces in our architecture in place to allow us to receive alerts. In this step, we will again increase the Min and Desired capacity of our Auto Scaling Group. Doing so will launch new EC2 instances, and their CPU's will spike during launch. This spike should be sufficient to trigger our CloudWatch alarm.

1. Navigate to **Services** → **EC2**
2. In the left menu, select **Auto Scaling Groups**
3. Select your **slalom_aws_intro_web_asg** group
4. In the box at the bottom, select the grey **Edit** button



5. Click in the load balancers text box and select the **slalom-aws-intro-web-elb** that was just created
6. Change the value of **Desired** and **Min** to 2. This will create 2 EC2 instances within your ASG
7. Click the blue **Save** button

System Architecture: Step 12

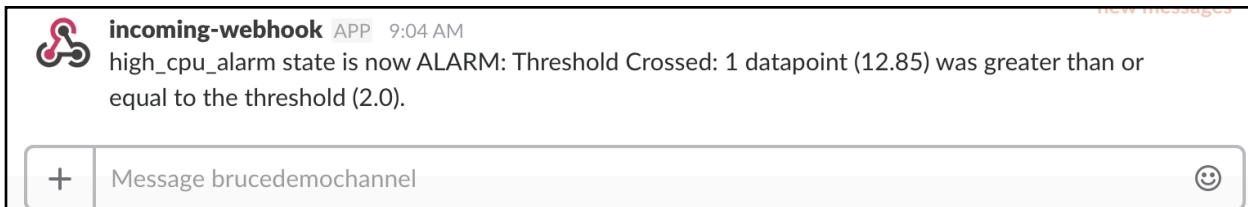


Step 13: Test it out!

As you have just increased the **Min** and **Desired** capacity from your autoscaling group, new instances EC2 instances will now be created within your Auto Scaling Group.

When new EC2 instances are created, their CPU Utilization spikes for a period time as they are booted. This spike in CPU should be enough to trigger our CloudWatch alarm which is configured to listen for CPU Utilization over 2% over a 1 minute period.

Keep an eye on your private Slack channel - You should receive an alert like the following:



System Architecture: Step 13

