



Hands-on Lab Session 3060

Build your first Omni-Bot: A chat bot that helps handle Service Management incidents

Robert Barron, IBM Cloud

You must include the first two pages of this template.

© Copyright IBM Corporation 2017

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

This document is current as of the initial date of publication and may be changed by IBM at any time.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

Table of Contents

Objectives	4
Initial setup	5
Create Slack Team	5
Sending an initial sample message from Impact	21
Sending a real event from Impact	27
Format the event sent to Slack.	37
Using a bot to update the event from Slack.	41
Advanced formatting of the event	49
References and extra material	49

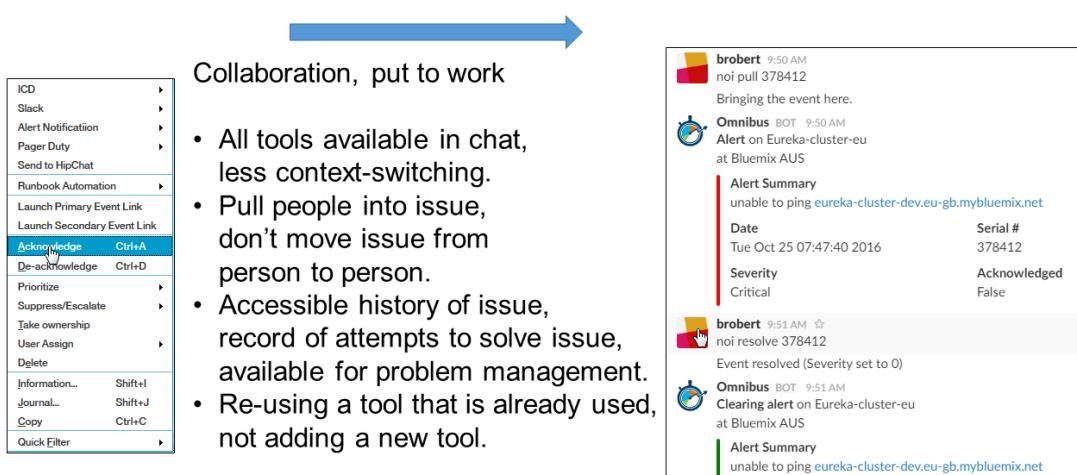
Objectives

ChatOps is a recent development in the field of Operations, wherein people use a single tool both to communicate amongst themselves and send & receive automated commands.

The aim of this lab is to demonstrate some initial capabilities of NOI and Slack integration to show a simple MVP solution which will showcase capabilities.

Code can be downloaded from <http://ibm.biz/chatops3060>

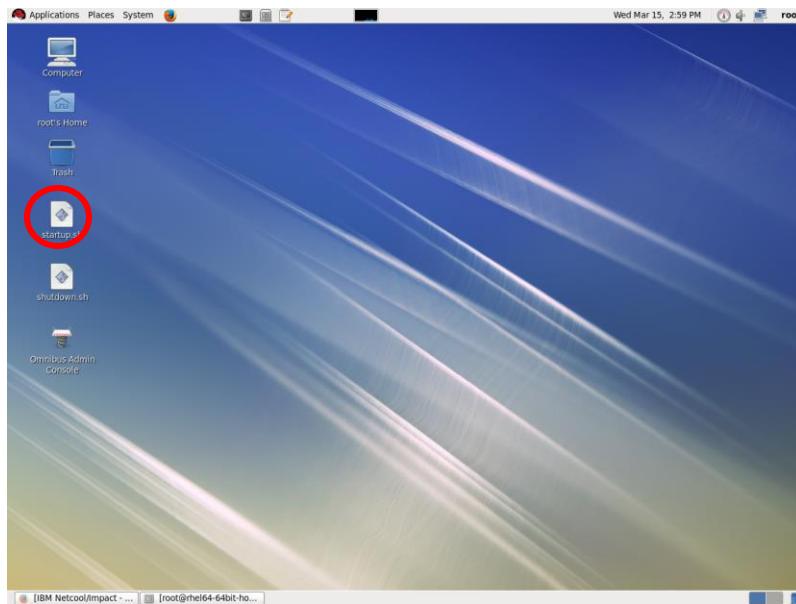
ChatOps



Initial setup

Open a terminal window and check your ip address using ifconfig -a.
Open the file /etc/hosts and make sure that the IP is correct. If it does not match, modify the IP address.

Start the application stack by running the script startup.sh which is on the desktop.



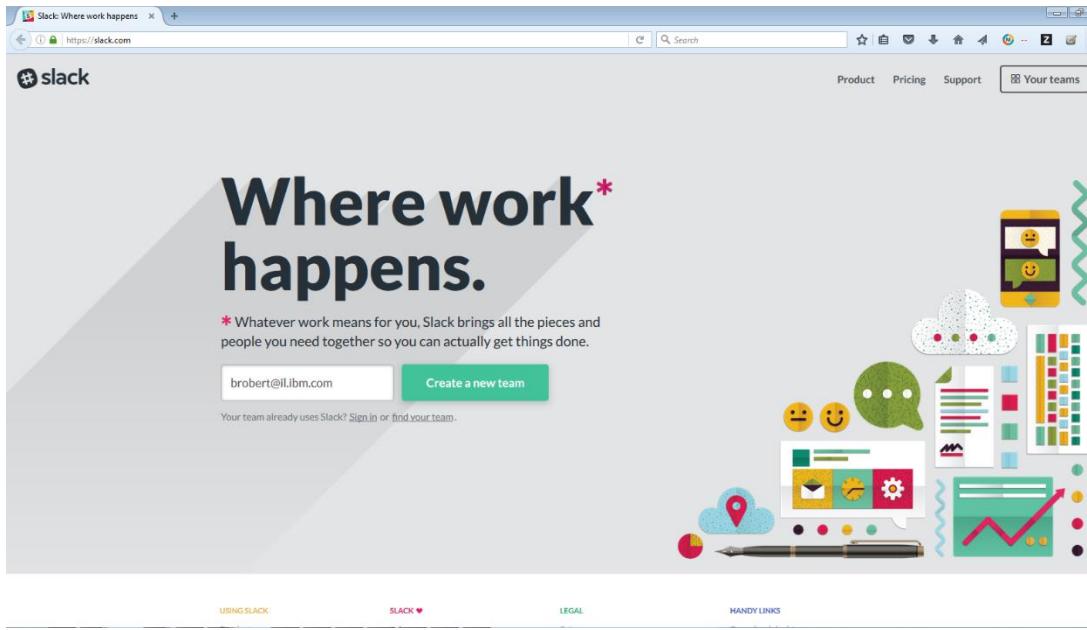
Create Slack Team

In this section, we will create the Slack team that will be used throughout the lab.

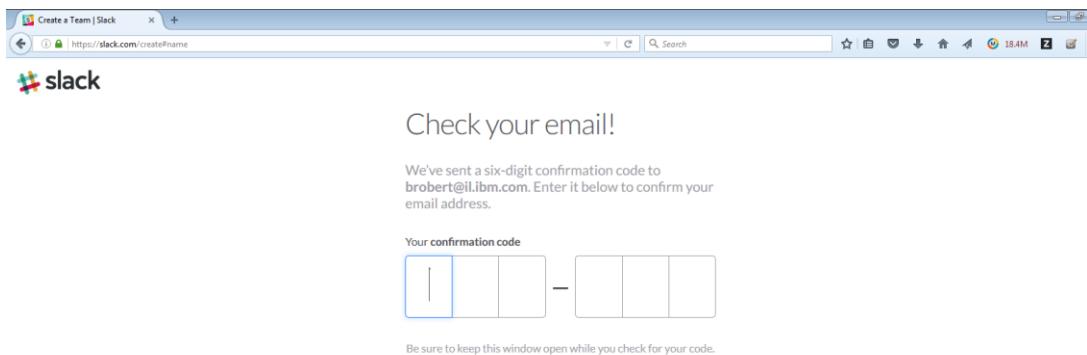
We will perform the following steps:

and create a token for API access. If you have administrative access to an existing Slack team, you may skip to step 11 by logging into your Slack.

1. Go to slack main page [www.slack.com](https://slack.com) and create a new team



2. Check your email and verify your team



3. Enter your name

The screenshot shows a web browser window with the URL <https://slack.com/create#name>. The page title is "Create a Team | Slack". The main heading is "What's your name?". Below it, a sub-instruction says "Your name will be displayed along with your messages in Slack.". There are two input fields: "first name" and "last name". The "Username" field is highlighted with an orange border and contains the placeholder "Please fill in a username.". Below the field, a note states: "Usernames must be all lowercase, with no spaces. They can only contain letters, numbers, periods, hyphens, and underscores." A checkbox labeled "It's ok to send me email about the Slack service." is checked. At the bottom is a grey button labeled "Continue to Password →".

4. Set your password

The screenshot shows a web browser window with the URL <https://slack.com/create#password>. The page title is "Create a Team | Slack". The main heading is "Set your password". Below it, a sub-instruction says "Choose a password for signing in to Slack.". There is a single input field labeled "Password" containing a series of dots. To the right of the field, a green progress bar is nearly full, with the word "Great!" written next to it. Below the field, a note says: "Passwords must be at least 6 characters long, and can't be things like password, 123456 or abcdef.". At the bottom is a green button labeled "Continue to Team Info →".

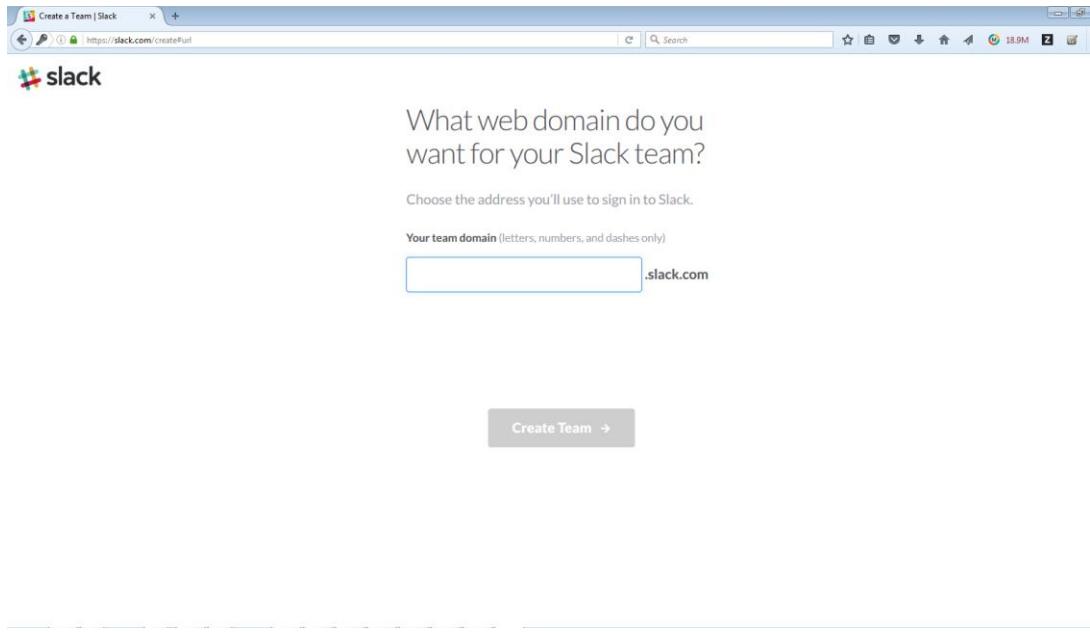
5. Fill in basic details about the team

The screenshot shows a web browser window titled "Create a Team | Slack". The URL is https://slack.com/create#teaminfo. The page has a header "Tell us about your team". There are three dropdown menus: "What will your team use Slack for?" (set to "Work"), "Great! What kind of company is it?" (set to "Other"), and "Last one: How big is your company?" (set to "1-9 people"). A green button at the bottom right says "Continue to Company Name →".

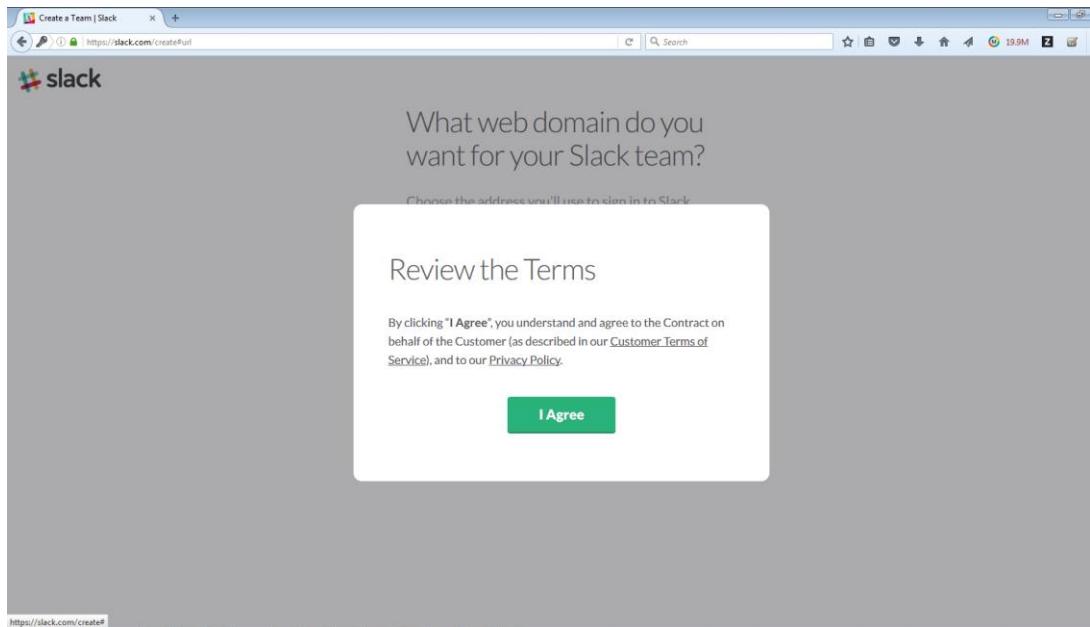
6. Fill in company name

The screenshot shows a web browser window titled "Create a Team | Slack". The URL is https://slack.com/create#teamname. The page has a header "What's your company called?". There is a text input field labeled "Company name" with the placeholder "Ex. Acme or Acme Marketing". Below the input field is a note: "We'll use this to name your Slack team, which you can always change later." A grey button at the bottom right says "Continue to Team Domain →".

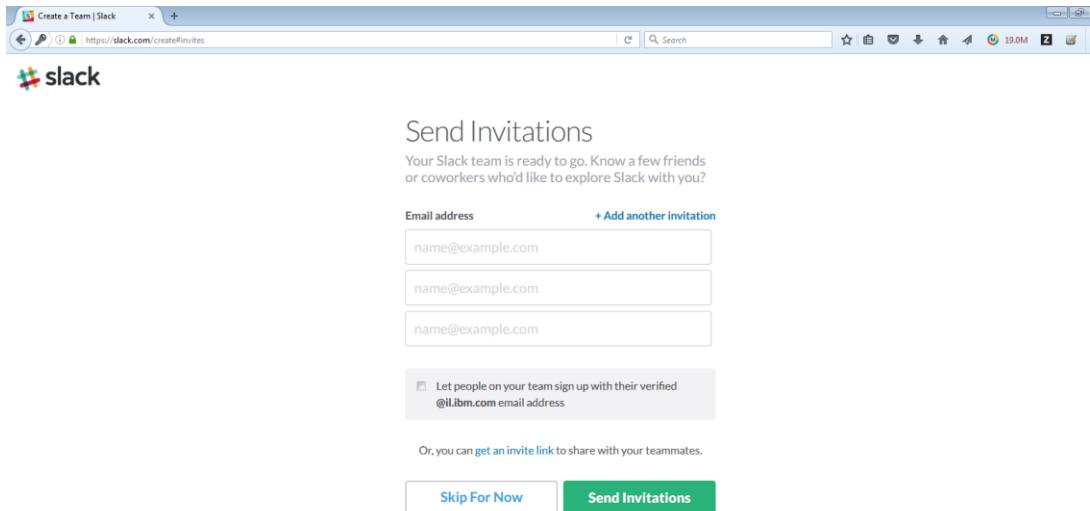
7. Choose your Slack domain



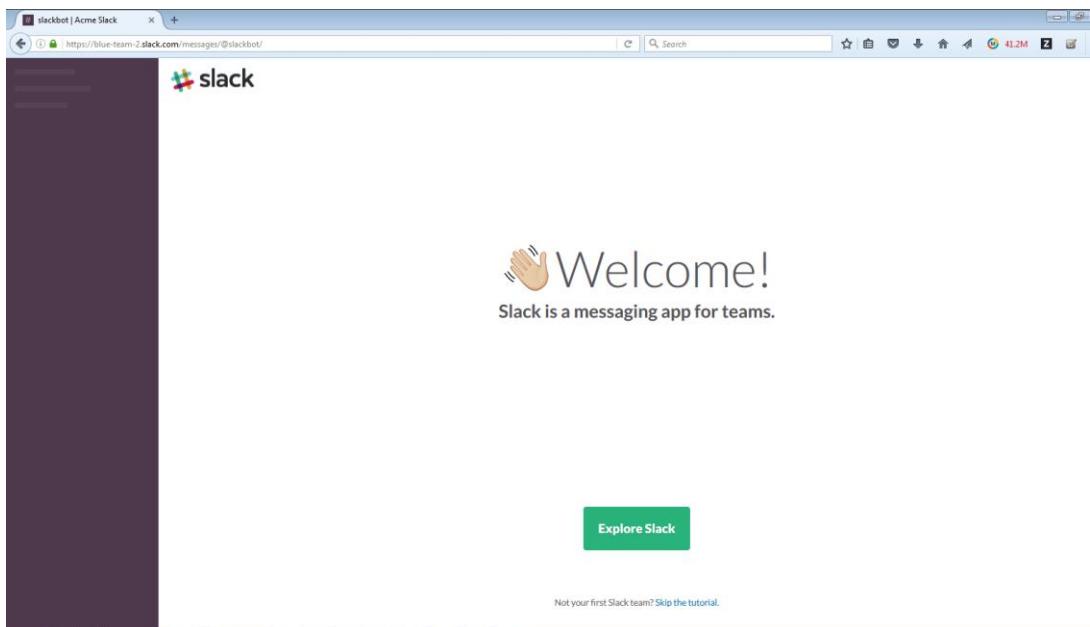
8. Agree to the terms (you have no choice, really!)



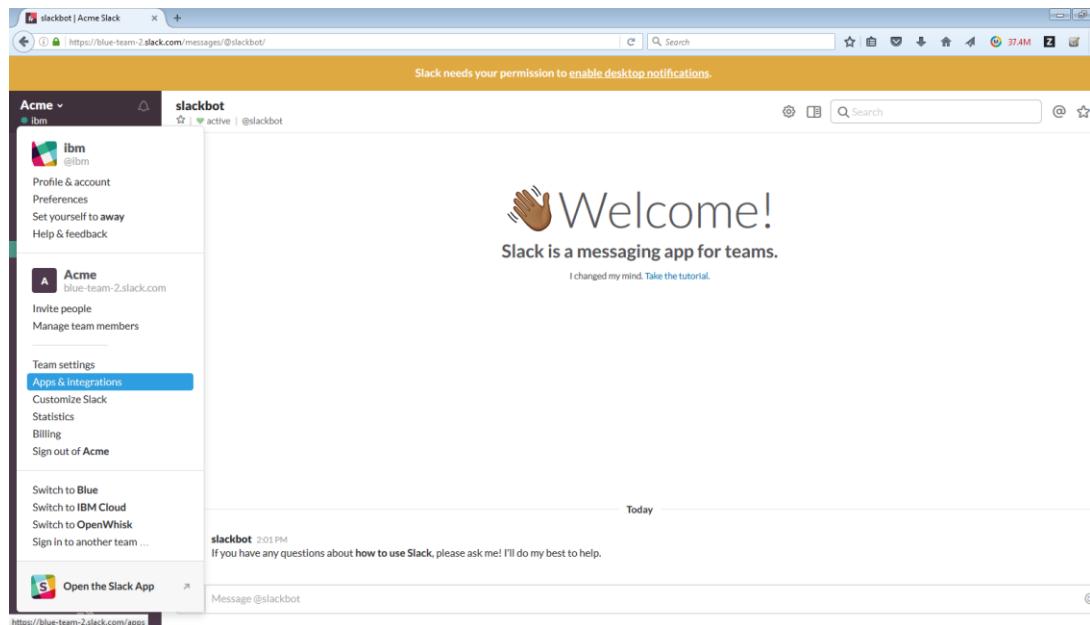
9. You may skip this step



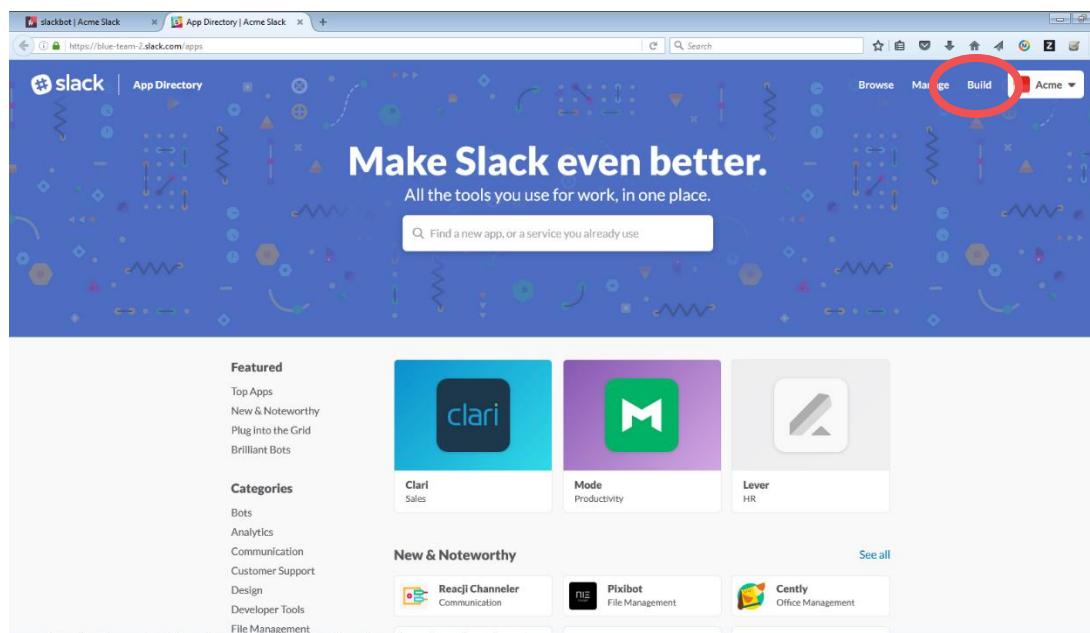
10. You have logged into your new Slack



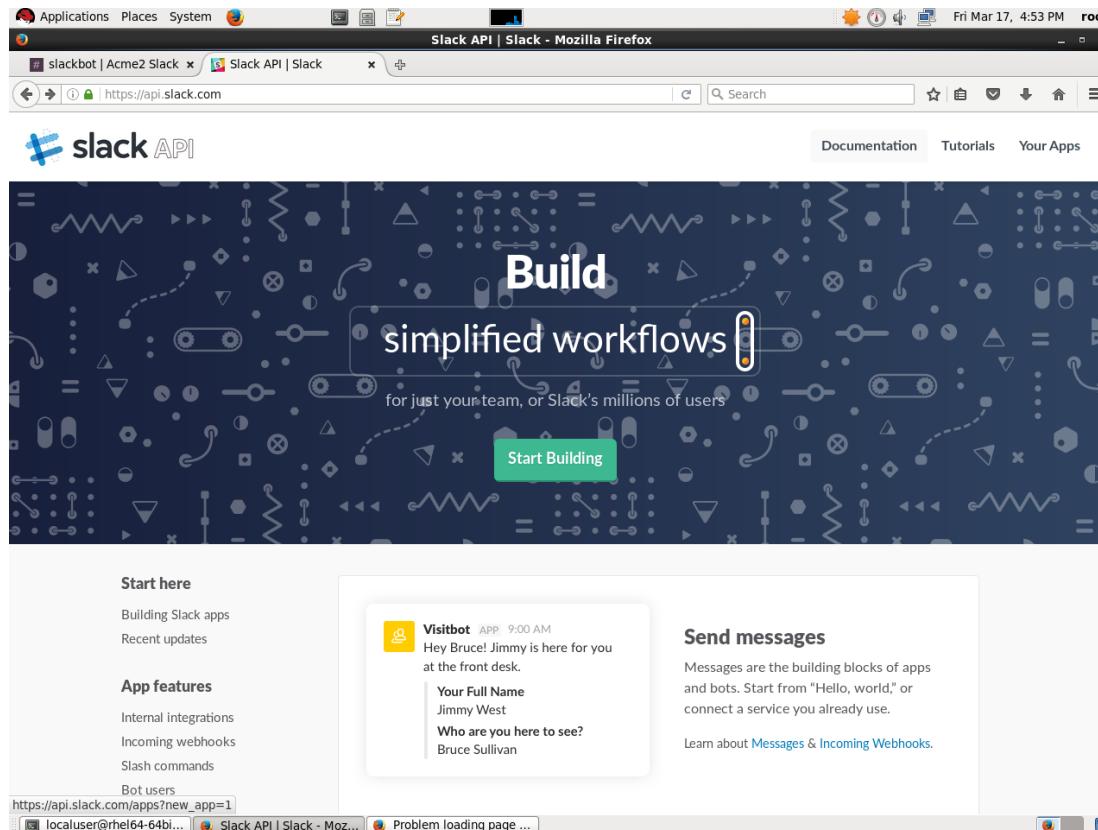
11. Open the menu to "Apps and Integrations"



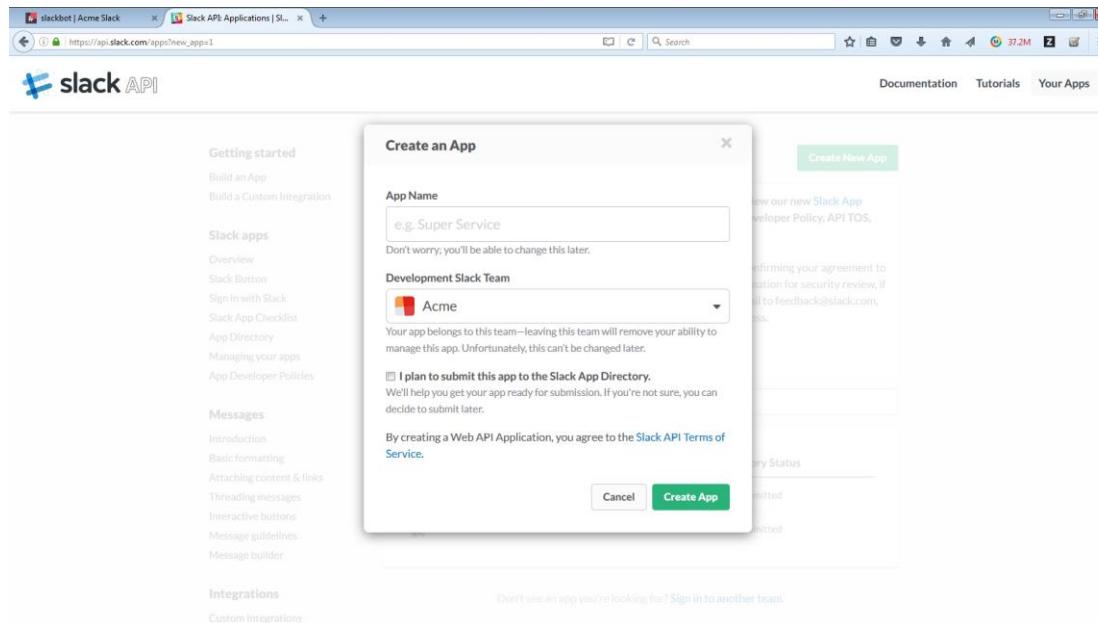
12. In the opened screen, choose the menu option Build



13. Select "Start Building"



14. Choose a name for your app (myBot, for example)



15. Congratulations – you have an app!

Scroll down and record the Client ID and Client Secret

The screenshot shows a Mozilla Firefox browser window with the title "Slack API: Applications | Acme2 Slack - Mozilla Firefox". The address bar shows the URL <https://api.slack.com/apps/A4LSD4QF9?created=1>. The main content area is titled "App Credentials". It contains the following information:

- Client ID:** 156892249351.156897160519
- Client Secret:** (redacted)
- Show** and **Regenerate** buttons are present next to the Client Secret field.

Below the credentials, a note states: "You'll need to send this secret along with your client ID when making your [oauth.access](#) request."

At the bottom of the "App Credentials" box are "Discard Changes" and "Save Changes" buttons. The browser's status bar at the bottom shows the URL <https://localuser@rhel64-64bi...>, the title "Slack API: Applications...", and the tab name "[Slack: Where work ha...]".

16. Choose OAuth & Permissions from the menu

The screenshot shows the Slack API Basic Information page for a bot named "myBot". On the left, there's a sidebar with "Settings" (Basic Information is selected), "Features" (Incoming Webhooks, Interactive Messages, Slash Commands, OAuth & Permissions, Event Subscriptions, Bot Users), and "Slack" (Help, Contact, Policies, Our Blog). The "OAuth & Permissions" link is highlighted and circled in red. The main content area is titled "Building Apps for Slack" and contains sections for "Add features and functionality" (Incoming Webhooks, Interactive Messages, Slash Commands, Event Subscriptions, Bots, Permissions), "Choose and configure the tools you'll need to create your app (or review all our documentation)", and "Create an app that's just for your team (or build one that can be used by any team) by following the steps below." At the bottom right are "Discard Changes" and "Save Changes" buttons.

17. Enter <https://slack.com/oauth/authorize> in the OAuth settings

The screenshot shows the Slack API OAuth & Permissions page for a bot named "Super Acme". On the left, there's a sidebar with "Basic Information", "Collaborators", "OAuth & Permissions" (selected), "Bot Users", "Interactive Messages", "Slash Commands", "Event Subscriptions", and "Submit to App Directory". The "OAuth & Permissions" link is highlighted. The main content area is titled "OAuth Settings" and contains a note: "You must specify at least one redirect URL for OAuth to work. If you pass a URL in an OAuth request, it must (at least partially) match one of the URLs you enter here." Below this is a "Redirect URLs" input field containing "https://slack.com/oauth/authorize". At the bottom right are "Save Changes" and "Revoke All OAuth Tokens" buttons. A "Success!" message is displayed at the top.

18. Success!

The screenshot shows the Slack API Applications interface for the 'Super Acme' app. The left sidebar includes links for Basic Information, Collaborators, OAuth & Permissions (which is selected), Bot Users, Interactive Messages, Slash Commands, Event Subscriptions, and Submit to App Directory. The main content area is titled 'OAuth & Permissions' and contains a section for 'OAuth Settings'. A warning message states: 'A redirect URL is missing. Please add at least one to enable OAuth.' Below this, there's a 'Redirect URL(s)' input field containing 'https://slack.com/oauth/authorize'. A note below the input field says: 'You must specify at least one URL for authentication to work. If you pass a URL in an OAuth request, it must (partially) match one of the URLs you enter here. [Learn more](#)'.

19. Create a bot user

The screenshot shows two browser tabs. The top tab is titled 'slackbot | Acme Slack' and the bottom tab is titled 'Slack API Applications | Ac...'. Both tabs show the URL <https://api.slack.com/apps/A4364hN74/oauth>. The main content area is titled 'Bot User' and shows a green 'Success!' bar at the top. Below it, there's a box containing the text: 'You can bundle a bot user with your app to interact with users in a more conversational manner. Learn more about how bot users work.' A blue 'Add a Bot User' button is visible. On the left sidebar, under 'Super Acme', the 'Bot Users' option is selected. Other options include Basic Information, Collaborators, OAuth & Permissions, Interactive Messages, Slash Commands, Event Subscriptions, and Submit to App Directory. At the bottom left, there are links to Slack, Help, Contact, Policies, and Our Blog.

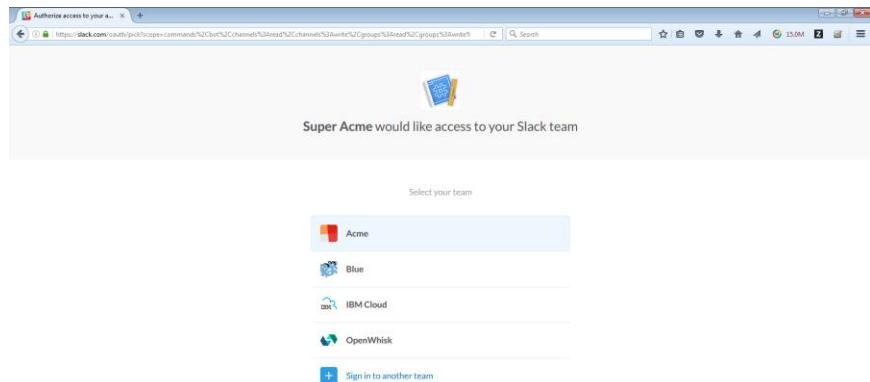
20. Success!

This screenshot shows the same interface as the previous one, but with a filled 'Default username' field. The field contains the value '@super_acme'. To the right of the field is a note: 'If this username isn't available on any team that tries to install it, we will slightly change it to make it work. Usernames must be all lowercase. They cannot be longer than 21 characters and can only contain letters, numbers, periods, hyphens, and underscores.' Below this, there's a toggle switch labeled 'Always Show My Bot as Online' which is currently set to 'Off'. The rest of the page and sidebar are identical to the previous screenshot.

21. Download and edit the file https://github.com/RobertJBarron/2017-InterConnect-3060/blob/master/SlackButtonForOAUTH_template.html. Edit the file and enter the clientId from step 15 and open the file in a browser

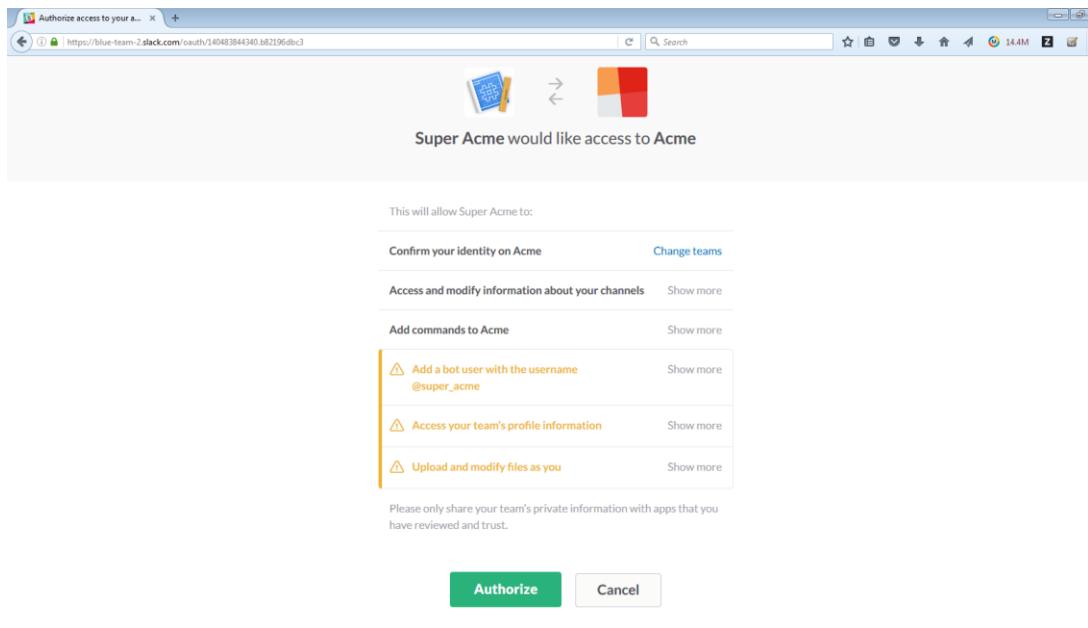


22. After pressing the button, choose your Slack team to give access to (this step depends on your existing Slack settings, you may not see it)



https://blue-team-2.slack.com/oauth/client_id-1398620713920800242&scope=commands,bot,channels:read,channels:write,groups:read,groups:write,files:read,user:read

23. Authorize the permissions



24. Record the "errorcode" in the browser URL.

This is the section of text between "code=" and "&state"



25. Go to the Slack api page <https://api.slack.com/methods/oauth.access/test> and enter the Client ID and Secret from step 16 and code from step 24 (you have 10 minutes before the code expires)

Getting started
Build an App
Build a Custom Integration

Slack apps
Overview
Slack Button
Sign in with Slack
Slack App Checklist
App Directory
Managing your apps
App Developer Policies

Messages
Introduction
Basic formatting
Attaching content & links
Threading messages
Interactive buttons
Message guidelines
Message builder

Integrations
Custom Integrations

oauth.access

Documentation Tester

Argument Value

client_id	Required	18201767.139208600242
client_secret	Required	x9e775037340e90fc9677
code	Required	1483844340.18ed904b72
redirect_url	Optional	
Extra args		

Test Method

URL
`https://slack.com/api/oauth.access?client_id=139898201767.139208600242&client_secret=x9e775037340e90fc9677&code=139898201767.140483844340.18ed904b72&pretty=1 (open raw response)`

26. Record the access tokens you receive (your own and the bot's)

Getting started
Build an App
Build a Custom Integration

Slack apps
Overview
Slack Button
Sign in with Slack
Slack App Checklist
App Directory
Managing your apps
App Developer Policies

Messages
Introduction
Basic formatting
Attaching content & links
Threading messages
Interactive buttons
Message guidelines
Message builder

Integrations
Custom Integrations
Incoming Webhooks
Slash Commands
Bot Users
Outgoing Webhooks

APIs
Web API

oauth.access

Documentation Tester

Argument Value

client_id	Required	18201767.139208600242
client_secret	Required	x9e775037340e90fc9677
code	Required	1483844340.18ed904b72
redirect_url	Optional	
Extra args		

Test Method

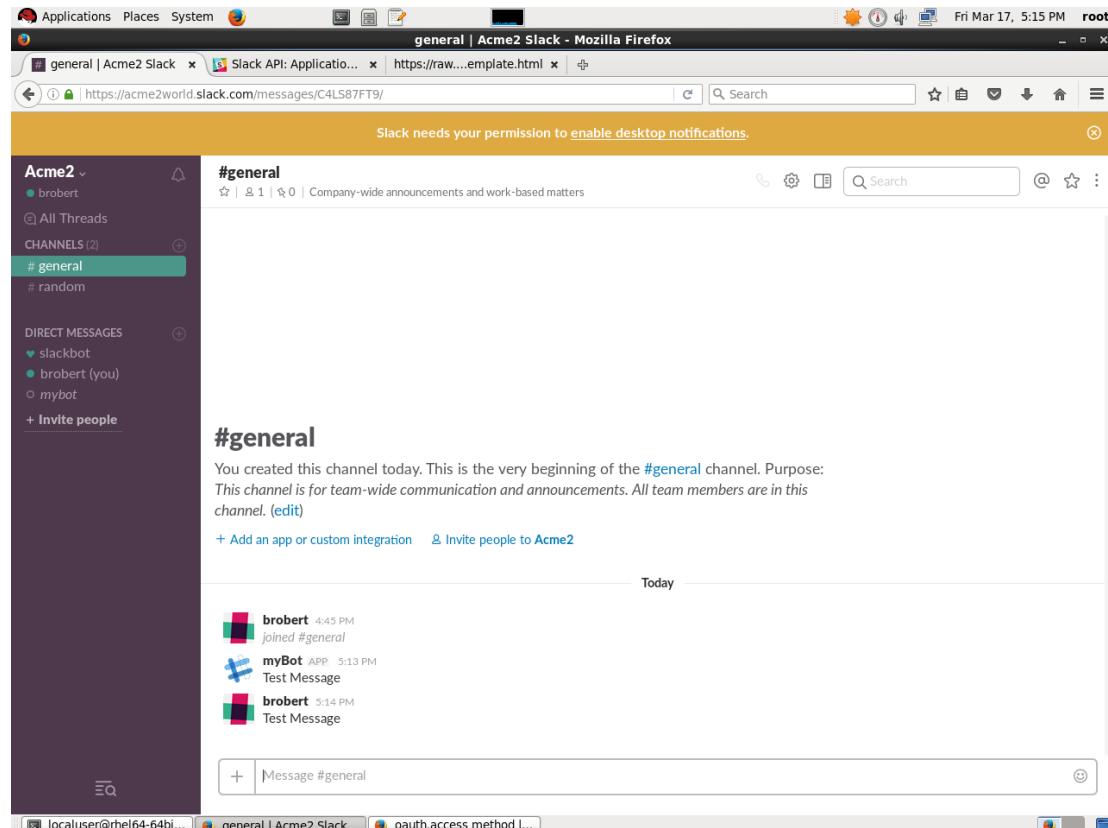
URL
`https://slack.com/api/oauth.access?client_id=139898201767.139208600242&client_secret=x9e775037340e90fc9677&code=139898201767.140483844340.18ed904b72&pretty=1 (open raw response)`

```
{
  "ok": true,
  "access_token": "xoxp-139898201767-1392086091166-140494712801-a089cb13f3d59cf83d4891",
  "scope": "bot:identity,bot:commands,channels:read,groups:read,users:read,channels:write",
  "team_id": "D438E6X4L",
  "team_name": "Acme",
  "team_id": "T438E6XNN",
  "bot": {
    "bot_user_id": "U452R7ND4",
    "bot_access_token": "xoxb-141086936446-m7vvn9AxHmljf6tNb4o4ehkj"
  }
}
```

Test your tokens using the command

```
curl -X POST -d "channel=%23general&text=Test%20Message"  
-d "token=__YourTokenHere__"  
https://slack.com/api/chat.postMessage
```

Verify that there a difference between running he command with each Token.



Sending an initial sample message from Impact

We will use Impact to send the initial event to Slack.

1. Export the Slack server certificate using the following command:

```
echo -n | openssl s_client -connect slack.com:443 | sed  
-ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >  
/tmp/slack.cert
```

2. Once you have exported the certificate, you must load it into Impact by running the following command:

```
/opt/IBM/tivoli/impact/sdk/bin/keytool -importcert -  
alias Slack-IC17-3060 -file /tmp/slack.cert -keystore  
/opt/IBM/tivoli/impact/wlp/usr/servers/NCI/resources/se  
curity/trust.jks -storepass impactadmin
```

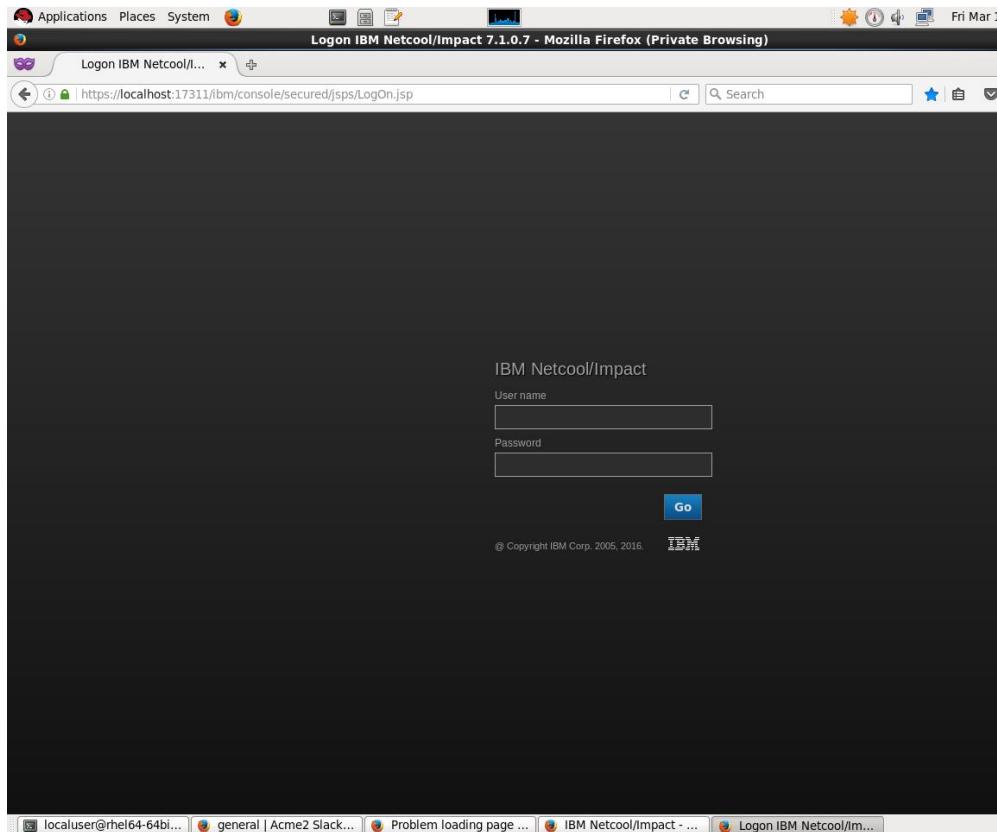
Respond "yes" when asked whether to trust the certificate.

3. Restart Impact using the commands

```
/opt/IBM/tivoli/impact/bin/stopImpactServer.sh  
/opt/IBM/tivoli/impact/bin/startImpactServer.sh
```

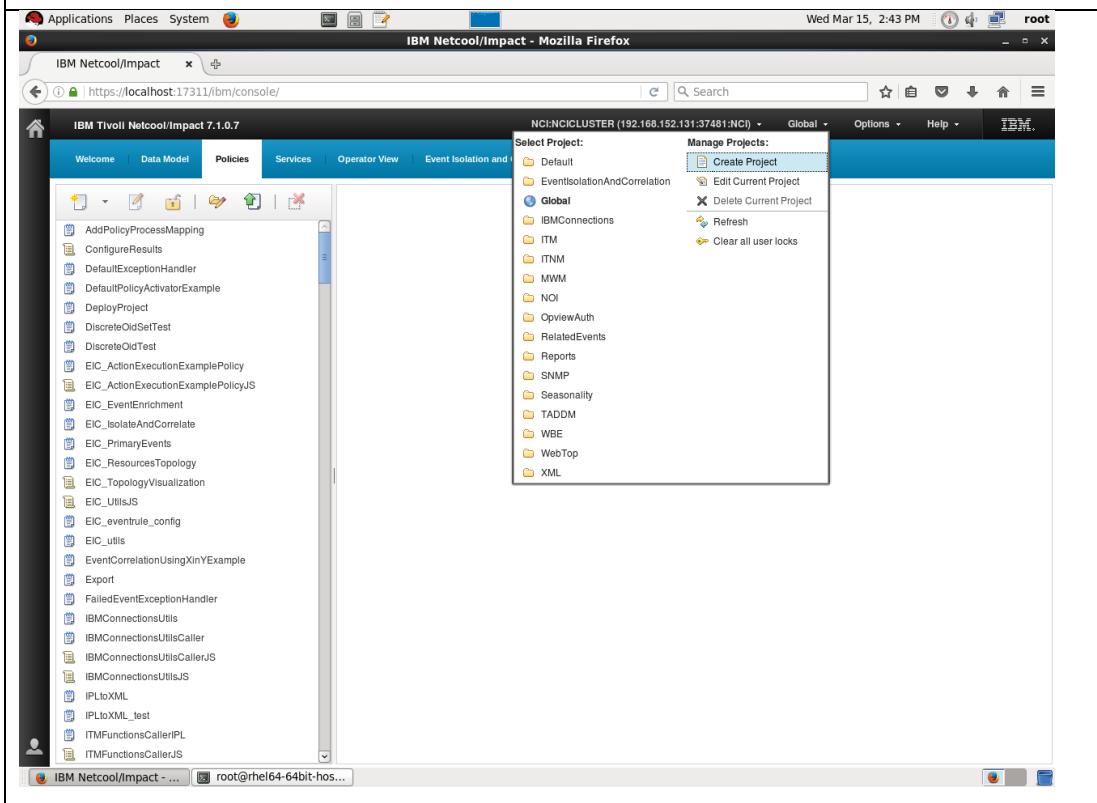
4. Open a new Firefox tab and login to Impact.

<http://localhost:17310/ibm/console> (or the bookmark Logon IBM Netcool/Impact 7.1.0.7)

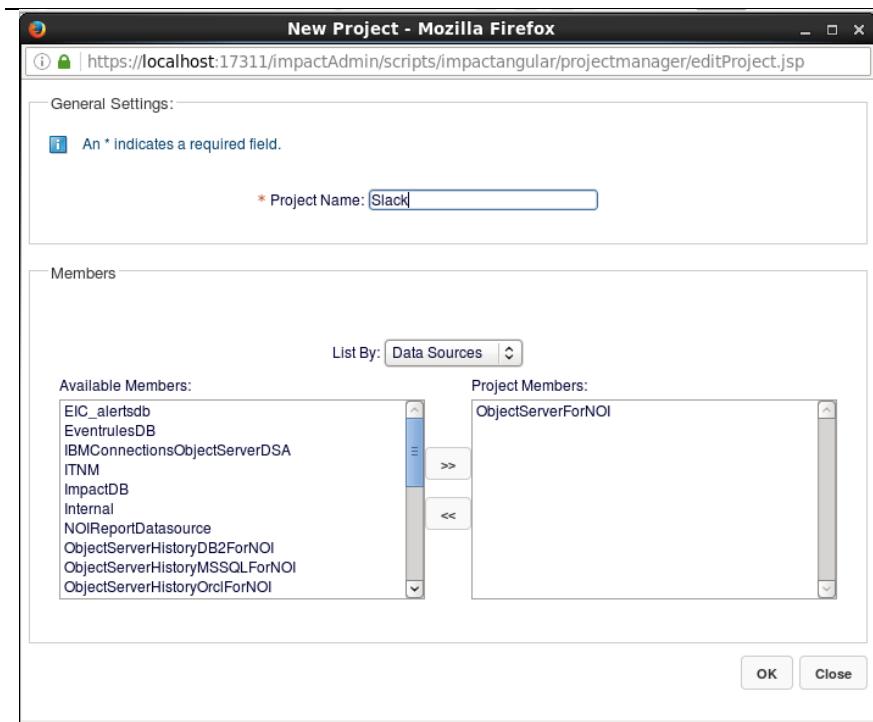


User/password is impactadmin/impactadmin

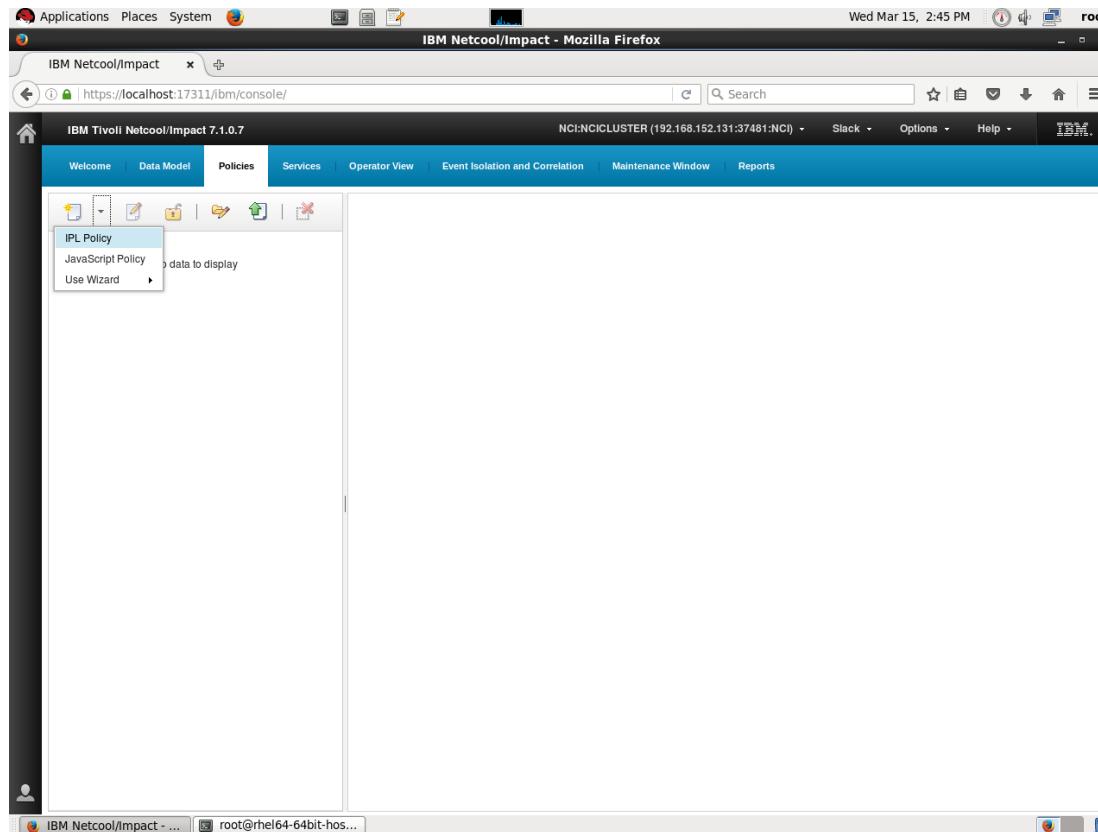
5. Click "Global" and then "Create Project"



6. Create a new project called "Slack" and add the datasource "ObjectServerForNOI"



7. Create a new IPL policy



Enter the following contents into the Policy and update the value of the Slack token to your own token:

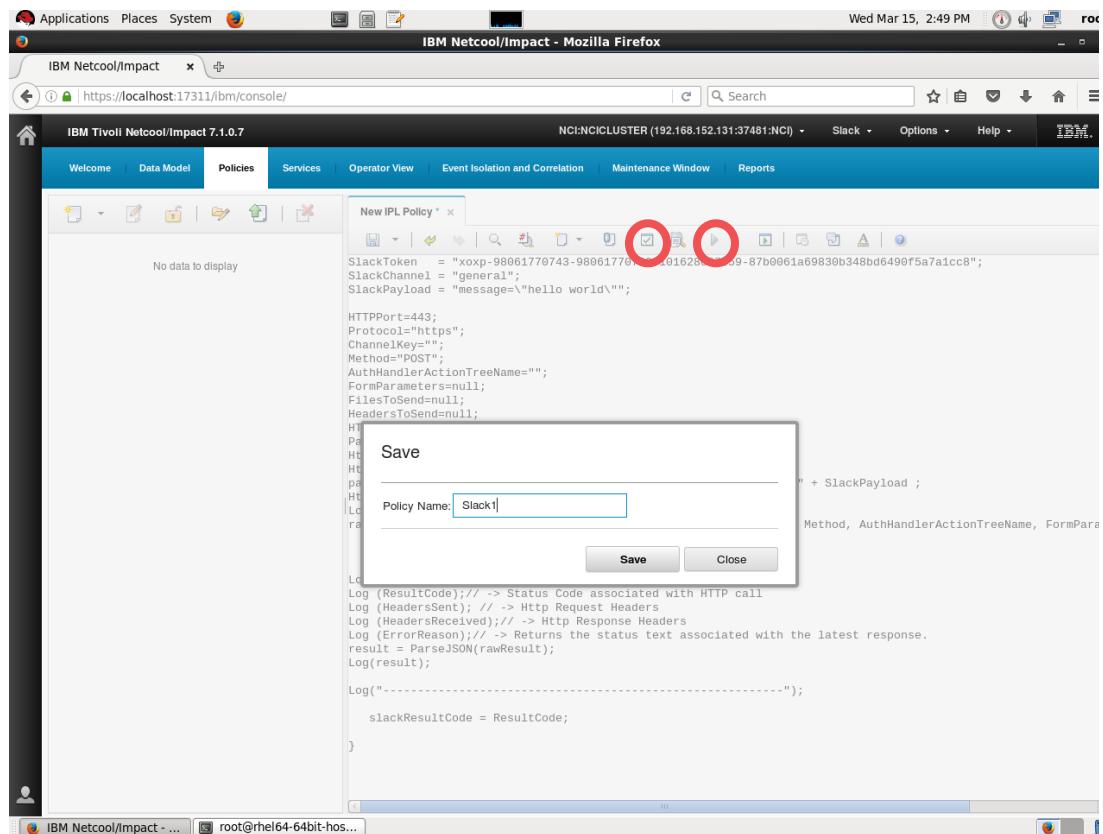
```
SlackToken = "__YOUR_TOKEN_HERE__";
SlackChannel = "general";
SlackPayload = "text=hello world";

HTTPPort=443;
Protocol="https";
ChannelKey="";
Method="POST";
AuthHandlerActionTreeName="";
FormParameters=null;
FilesToSend=null;
HeadersToSend=null;
HTTPHost="slack.com";
Path="/api/chat.postMessage";
HttpProperties=NewObject();
HttpProperties.ContentType="application/x-www-form-urlencoded";
payload = "token=" + SlackToken + "&channel=" + SlackChannel + "&" + SlackPayload ;
HttpProperties.Content=( payload );
Log( HttpProperties);
rawResult=GetHTTP(HTTPHost, HTTPPort, Protocol, Path, ChannelKey, Method, AuthHandlerActionTreeName, FormParameters, FilesToSend, HeadersToSend, HttpProperties);

Log (ThePage); // "ThePage" is useful for diagnosing http errors
Log (resultCode); // -> Status Code associated with HTTP call
Log (HeadersSent); // -> Http Request Headers
Log (HeadersReceived); // -> Http Response Headers
Log (ErrorReason); // -> Returns the status text associated with the latest response.
result = ParseJSON(rawResult);
Log(result);

Log ("-----");
```

8. Validate the syntax of the code and run the policy.



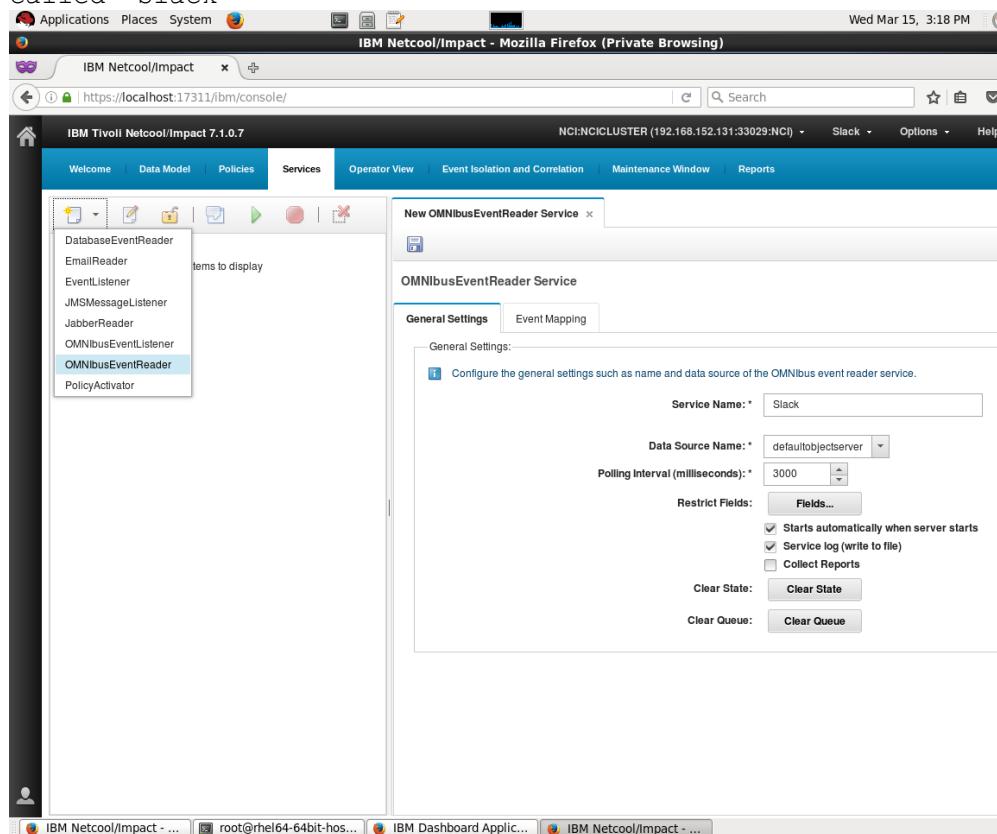
9. Check the contents of the policy log file at : /opt/IBM/tivoli/impact/logs/

Sending a real event from Impact

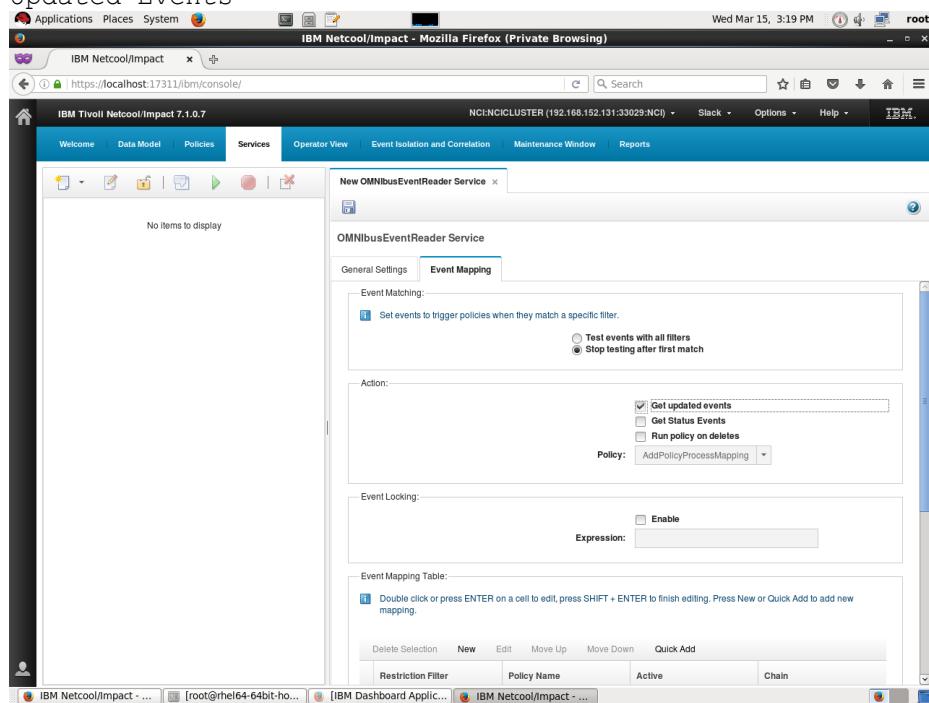
1. Create a file and insert the following SQL commands

```
alter table alerts.status add Slack integer;
go
insert into alerts.conversions values ('Slack0', 'Slack', 0, 'Not
Sent');
go
insert into alerts.conversions values ('Slack1', 'Slack', 1,
'Queued');
go
insert into alerts.conversions values ('Slack2', 'Slack', 2,
'Sent');
go
alter table alerts.status add SlackChannel varchar (32);
go
```

2. In Impact, open the "Services" tab and create a new service called "Slack"



3. Move to the tab "Event Mapping" and check the option "Get Updated Events"



-
4. Add a new entry to the "Event Mapping Table" which will run the policy you created when the event "Slack = 1" occurs (mark it active)

The screenshot shows a Mozilla Firefox browser window with the URL <https://localhost:17311/impactAdmin/jsp/manageEventMapping.jsp?action=new&widgetId=1>. The title bar says "Create New Mapping - Mozilla Firefox (Private Browsing)".

Create New Event Filter Dialog:

- Filter Expression: Slack = 1
- Policy to Run: Slack1
- Active
- Chain
- Analyze Filter: Analyze Filter

Event Mapping Table View:

Policy Name	Active	Chain
No items to display		

Buttons at the bottom of the dialog: Help, Ok, Cancel.

Buttons at the top of the table view: Edit, Move Up, Move Down, Quick Add.

Buttons at the bottom of the table view: Analyze Event Filter Mapping, Order By: [text input], Analyze Event Filter Mapping.

The status bar at the bottom of the browser window shows several tabs and the current tab: Create New Mapping - ...

5. Save the service

The screenshot shows the IBM Netcool/Impact 7.1.0.7 interface. The top navigation bar includes Applications, Places, System, and a Firefox icon. The title bar reads "IBM Netcool/Impact - Mozilla Firefox (Private Browsing)" and the address bar shows "https://localhost:17311/ibm/console/". The main menu has tabs for Home, Welcome, Data Model, Policies, Services, Operator View, Event Isolation and Correlation, Maintenance Window, and Reports. The Services tab is selected. A sidebar on the left lists various services, with "Slack" highlighted. The main content area is titled "OMNibusEventReader Service" and shows the "Event Mapping" tab selected. It contains an "Event Mapping Table" with the following data:

Restriction Filter	Policy Name	Active	Chain
Slack = 1	Slack1	Yes	No

Buttons at the bottom include "Delete Selection", "New", "Edit", "Move Up", "Move Down", and "Quick Add". There is also an "Analyze Event Filter Mapping" button and a "Order By:" input field.

6. Start the Service

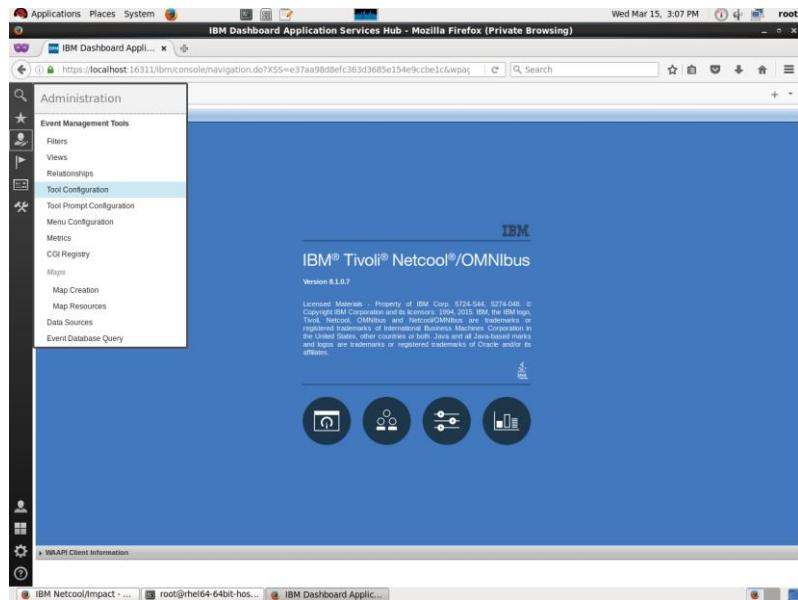
This screenshot is identical to the one above, showing the same configuration for the OMNibusEventReader Service in the IBM Netcool/Impact 7.1.0.7 interface. The service "Slack" is listed in the sidebar, and the "Event Mapping" table shows the same mapping as the previous screenshot.

7. Reopen the Slack policy that you created in the previous step and modify it in the following ways:

These changes will make sure that the event is only sent once. The policy is triggered by the service when the value of Slack=1 and the policy itself changes the value to 2 when it completes.

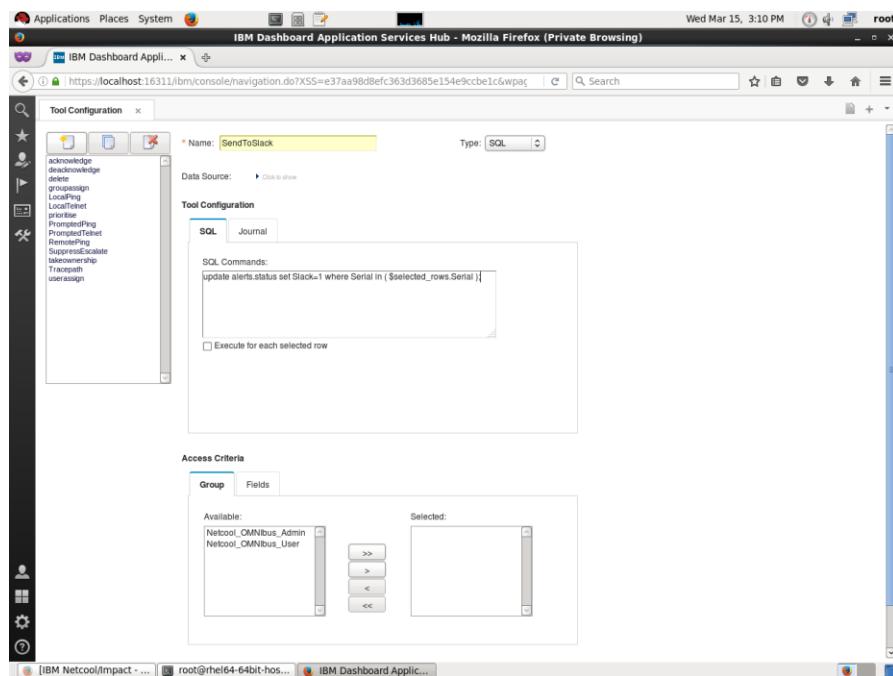
8. Login to DASH at <http://localhost:16311/ibm/console> with the user/password of ncoadmin/nocadmin (note that you cannot be logged into to DASH and Impact during the same session – you will either need to logout of Impact or open a private tab for the DASH session)

9. Open the "Tool config" menu

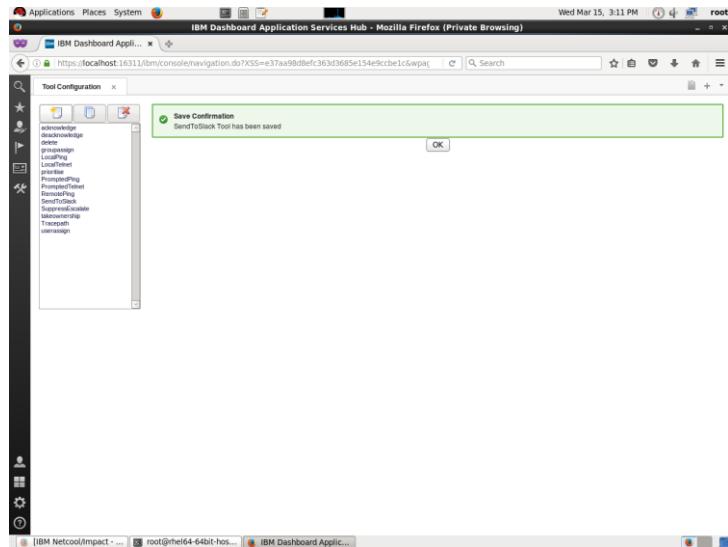


10. Create a new SQL tool called "SendToSlack" and enter the following command :

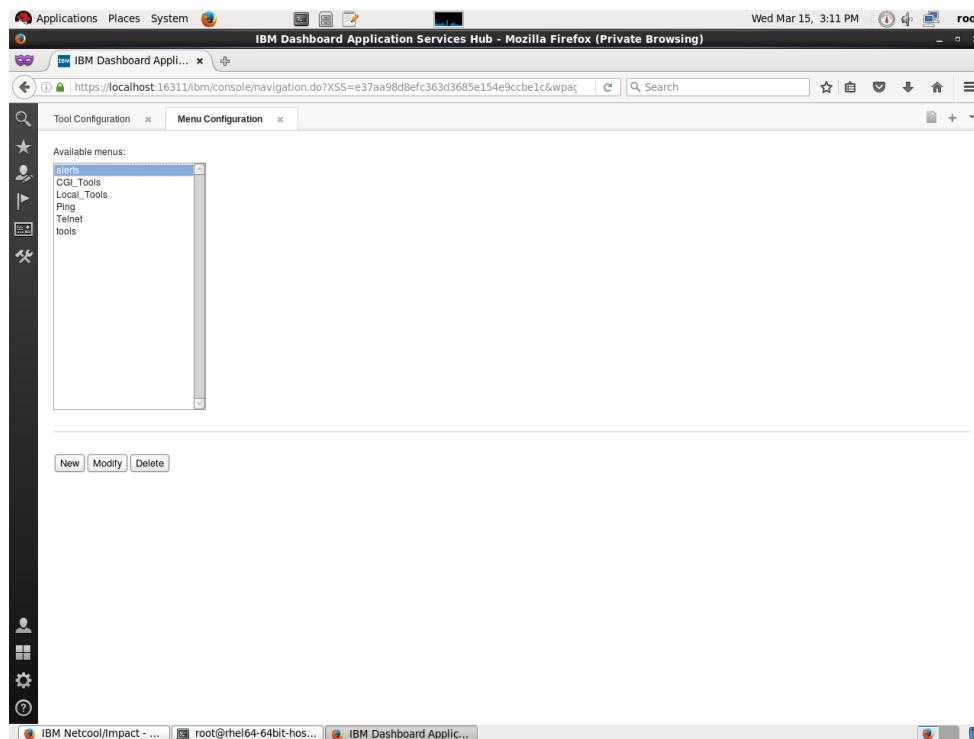
```
update alerts.status set Slack=1 where Serial in
($selected_rows.Serial);
```



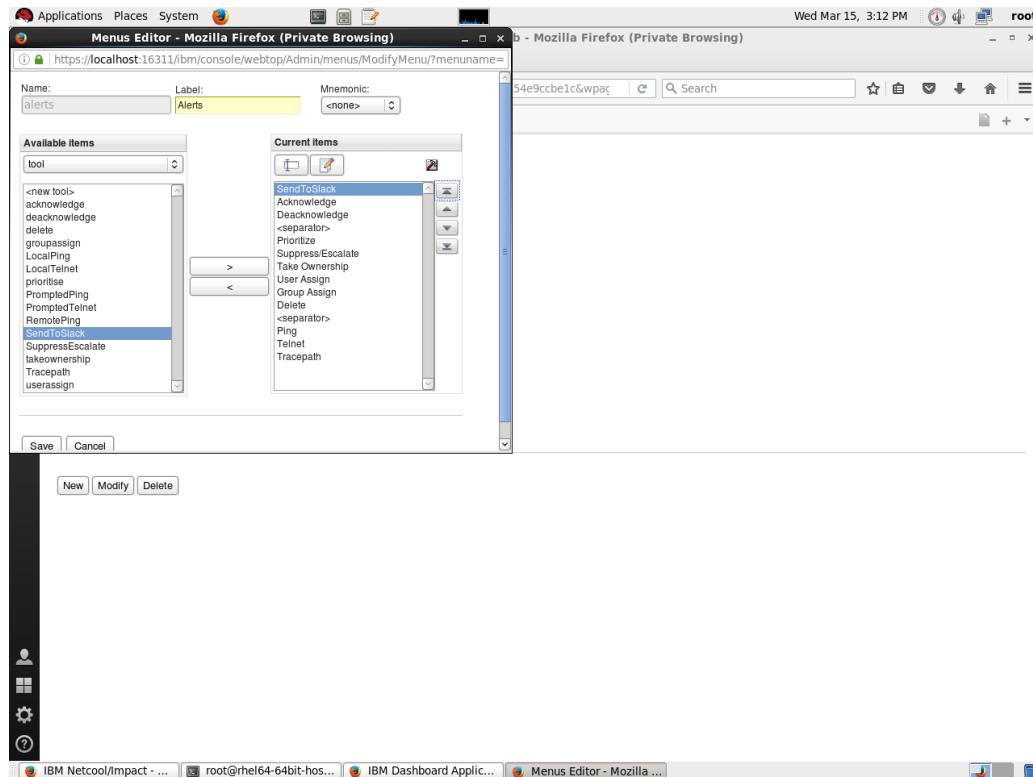
11. Save the tool



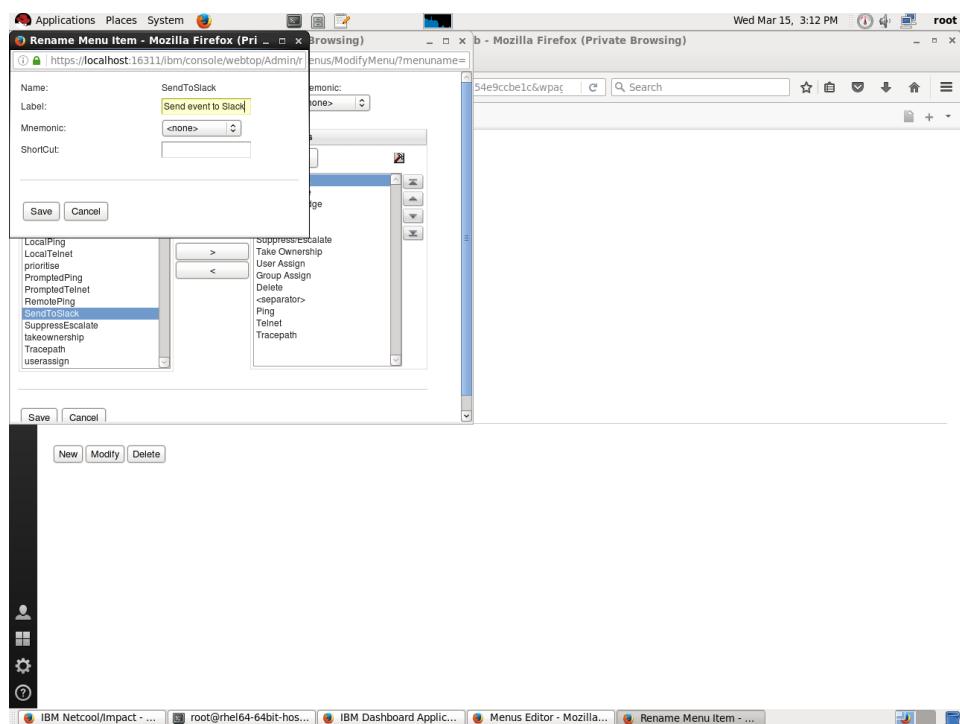
12. Open the "Menu Configuration" tab and choose the "alerts" menu to modify



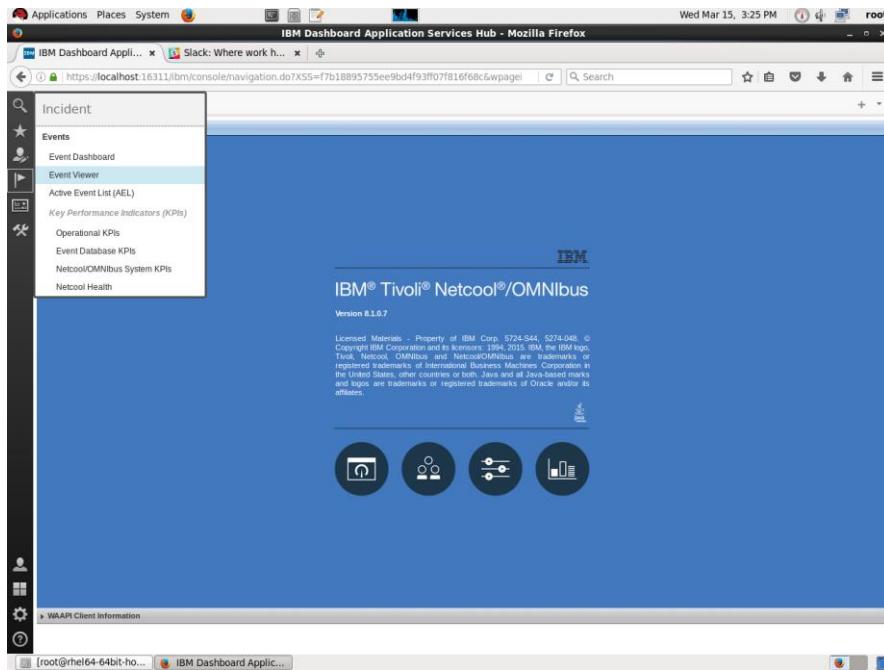
13. Modify the "alerts" menu by moving the SendToSlack tool to the right



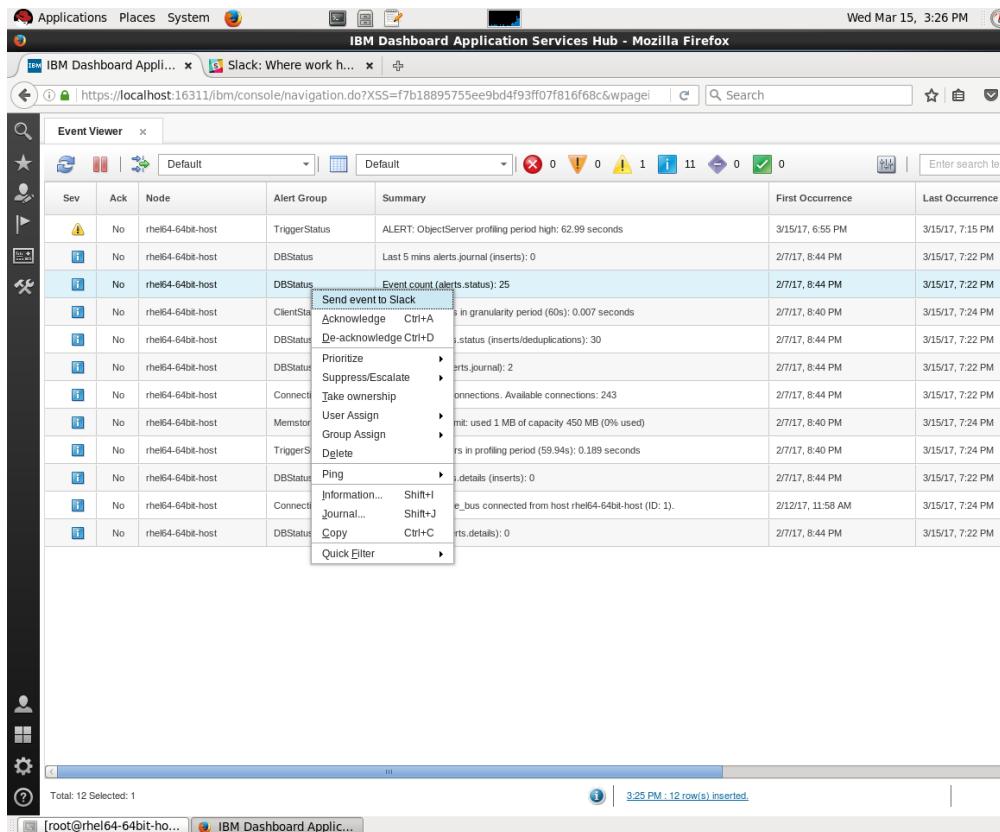
14. Rename the tool in the menu to "Send event to Slack"



15. Open the event viewer tab



16. right click an event to verify that you see the new tool.



17. Right click again and show the event details, verify that the event has been modified.

Field	Value
ScopeID	
Serial	5
ServerName	AGG_P
ServerSerial	5
Service	
SiteName	
Slack	Sent
SlackChannel	
StateChange	3/15/17, 7:32 PM
SuppressEscal	Normal
TTNumber	
Tally	431
TaskList	Not in Task List
Type	Information

18. Check Slack and see if your event appears

Format the event sent to Slack.

19. Create a new Impact policy with the following contents:

```

SlackToken      = "__REPLACE_WITH_TOKEN__";
DASH_Server    =
"https://__REPLACE_WITH_SERVER_FQDN_OR_IP__:16311/ibm/console"
;

HTTPPPort=443;
Protocol="https";
ChannelKey="";

```

```
Method="POST";

AuthHandlerActionTreeName="";

FormParameters=null;

FilesToSend=null;

HeadersToSend=null;

HTTPHost="slack.com";

Path="/api/chat.postMessage";

HttpProperties=NewObject();

HttpProperties.ContentType="application/x-www-form-urlencoded";

SlackChannel = "general";

SlackNode      = @Node;

SlackTime      = LocalTime(@LastOccurrence, "EEE MMM dd HH:mm:ss yyyy");

SlackMessage = replace(@Summary,"'','');

if (@Severity == 0 ) { SlackSeverity = "Clear";
SlackColour = "#00FF00"; PreText = "*Clearing alert*"; }
elseif

    (@Severity == 1 ) { SlackSeverity = "Indeterminate";
SlackColour = "#808080"; PreText = "*Information*"; }
elseif

    (@Severity == 2 ) { SlackSeverity = "Warning";
SlackColour = "#00FFFF"; PreText = "*Alert*"; }
elseif

    (@Severity == 3 ) { SlackSeverity = "Minor";
SlackColour = "#FFFF00"; PreText = "*Alert*"; }
```

```
(@Severity == 4 ) { SlackSeverity = "Major";
SlackColour = "#FFA500"; PreText = "*Alert*"; }
elseif

    (@Severity == 5 ) { slackSeverity = "Critical";
SlackColour = "#FF0000"; PreText = "*Alert*"; }
else

    { SlackSeverity = "Indeterminate";
SlackColour = "#808080"; PreText = "*Information*"; }

if (@Acknowledged == 0) { SlackAcked = 'False'; }
else { SlackAcked = 'True'; }

SlackSerial = @Serial; // Need to use local variables so
that Impact will not try to update them fields after we
pass them as parameters.

SlackIdentifier = @Identifier; // Need to use local
variables so that Impact will not try to update them after
we pass them as parameters.

SlackPayload =
"mrkdwn=true&username=Omnibus&icon_emoji=:itsm:&text=" +
PreText + " on " +
SlackNode + "&" ;

Attachments = "attachments=[ { \"color\":\""+ SlackColour +
"\",\"ts\":"+ @LastOccurrence + ",\"text\": \""+
"\", \"mrkdwn_in\": [\"text\"], \"fields\": [
" +
"\"{\"title\": \"Alert Summary\" + " (Serial #"
+ SlackSerial + ") + ClassText + "\",\"value\": \""+
SlackMessage + "\",\"short\": false}\" +
```

```
        "]} ] " ;  
        // Post the main body  
  
        payload = "token=" + SlackToken + "&channel=" +  
        slackChannel + "&" + slackPayload ;  
  
        HttpProperties.Content=( payload ) ;  
  
        Log( HttpProperties ) ;  
  
        rawResult=GetHTTP(HTTPHost, HTTPPort, Protocol, Path,  
        ChannelKey, Method, AuthHandlerActionTreeName,  
        FormParameters, FilesToSend,  
  
        HeadersToSend, HttpProperties ) ;  
  
        Log (ThePage); // "ThePage" is useful for diagnosing http  
        errors  
  
        Log (resultCode); // -> Status Code associated with HTTP  
        call  
  
        Log (HeadersSent); // -> Http Request Headers  
  
        Log (HeadersReceived); // -> Http Response Headers  
  
        Log (ErrorReason); // -> Returns the status text associated  
        with the latest response.  
  
        result = ParseJSON(rawResult);  
  
        Log(result);  
  
        Log("-----");  
  
        slackresultCode = resultCode;
```

```
@Slack = 2;  
ReturnEvent(EventContainer);
```

20. Correct the values for the Slack token and the DASH IP.
21. Modify the Impact service mapping to use the new policy instead of the old one and stop/start the service (see steps 4 & 5)

Using a bot to update the event from Slack.

1. Modify the Objectserver configuration file to allow remote HTTP access
modify the file /opt/IBM/tivoli/netcool/omnibus/etc/AGG_P.props

```
NHttpd.EnableHTTP: TRUE  
NHttpd.ListeningPort: 7070  
NRestOS.Enable: TRUE
```

2. Restart the objectserver with the following commands:
`/opt/IBM/tivoli/netcool/omnibus/bin/nco_pa_stop -process MasterObjectServer`
`/opt/IBM/tivoli/netcool/omnibus/bin/nco_pa_start -process MasterObjectServer`

Use the root password when asked.

3. Test that you have remote access by running the command
`telnet localhost 7070`
4. Open a cmd window and run the following commands as root:

```
curl -sL https://rpm.nodesource.com/setup\_6.x | sudo -E bash -  
yum install -y nodejs
```

These commands install node.js to the server

5. Run the following command as root:

```
npm install -g yo generator-hubot
```

This command installs a bot "generator".

6. Run the following command as root:

```
npm install -g node-omnibus
```

This command installs a node.js library that can converse with the ObjectServer over REST

7. The following commands must be run as localuser

```
su localuser
```

```
cd ~
```

```
mkdir bot
```

```
cd bot
```

```
yo hubot
```

8. Enter your own details for the bot name and description, choose "slack" as the adaptor type

```
[localuser@rhel64-64bit-host ~]$ yo hubot
[localuser@rhel64-64bit-host ~]$ cd mybot/
[localuser@rhel64-64bit-host mybot]$ yo hubot
Extracting input for self-replication process
? Owner user@ibm.com
? Bot name mybot
? Description My first NOI-Slackbot
? Bot adapter slack
  create bin/hubot
  create bin/hubot.cmd
  create Procfile
  create README.md
  create external-scripts.json
  create hubot-scripts.json
  create .gitignore
  create package.json
  create scripts/example.coffee
  create .editorconfig
Self-replication process complete...
Good luck with that.

[localuser@rhel64-64bit-host ~]$
```

9. Run the command "ls -ltr" and you will see the following files:

```
[localuser@rhel64-64bit-host ~]$ ls -ltr
total 36
drwxrwxr-x. 2 localuser localuser 4096 Mar 17 16:38 bin
-rw-r--r--. 1 localuser localuser 7820 Mar 17 16:38 README.md
-rw-r--r--. 1 localuser localuser 24 Mar 17 16:38 Procfile
-rw-r--r--. 1 localuser localuser 213 Mar 17 16:38 external-scripts.json
drwxrwxr-x. 2 localuser localuser 4096 Mar 17 16:38 scripts
-rw-r--r--. 1 localuser localuser 2 Mar 17 16:38 hubot-scripts.json
drwxrwxr-x. 173 localuser localuser 4096 Mar 17 16:39 node_modules
-rw-r--r--. 1 localuser localuser 611 Mar 17 16:39 package.json
[localuser@rhel64-64bit-host ~]$
```

10. Edit the file package.json and add a dependency to the node-omnibus component:

```
"node-omnibus": "^0.1.1"
```

Create the file noi.coffee in the ./scripts subdirectory and insert the following code into it:

```
console.log ('Omnibus = ' + process.env.HUBOT_OMNIBUS_HOST)
```

```
omnibus = require('node-omnibus')

request = require('request')

HubotSlack = require 'hubot-slack'

noiChannelName = ""

omnibusConnection = omnibus.createConnection(
  host: process.env.HUBOT_OMNIBUS_HOST
  port: '8080'
  user: process.env.HUBOT_OMNIBUS_USER
  password: process.env.HUBOT_OMNIBUS_PASSWORD)

module.exports = (robot) ->
  robot.hear /noi ack (.*)/i, (msg) ->
    sql = 'UPDATE alerts.status set Acknowledged =
where Serial=' + msg.match[1]
    omnibusConnection.sqlCommand sql, (err, rows,
    numrows, coldesc) ->
      console.log "err:" + err
      if numrows == 0
        msg.send "I can't find any event with a
serial number of " + msg.match[1]
        msg.send "Perhaps the event has been
closed already? Otherwise, try another serial number."
      else
        msg.send "Event acknowledged"
```

```
robot.hear /noi deack (.*)/i, (msg) ->

    sql = 'UPDATE alerts.status set Acknowledged = 0
where Serial=' + msg.match[1]

    omnibusConnection.sqlCommand sql, (err, rows,
numrows, coldesc) ->

        console.log "err:" + err

        if numrows == 0

            msg.send "I can't find any event with a
serial number of " + msg.match[1]

            msg.send "Perhaps the event has been
closed already? Otherwise, try another serial number."

        else

            msg.send "Event de-acknowledged"
```

11. Run the bot using the command

```
HUBOT_OMNIBUS_PASSWORD= HUBOT_OMNIBUS_HOST=localhost
PORT=7070 HUBOT_SLACK_TOKEN=<<YOURTOKENHERE>>
./bin/hubot --adapter slack
```

**12. From the Slack window, invite the bot to your slack channel with the command
/invite <botname>**

**13. In the DASH event viewer, choose an event and view it's details to find the Serial
number. Run the command "noi ack <serial#>" in the Slack command and verify
that it has been acknowledged in the DASH event viewer.**

14. Stop the bot using ctrl-C and update the file noi.coffee with the following rows:

```
robot.hear /noi journal (.*?) (.*)/i, (msg) ->

    dt = Date.now()

    dt = dt / 1000

    sql = 'Insert into alerts.journal (KeyField,
Serial, UID, Chrono, Text1) values (''' + msg.match[1] +
```

```
'_`' + dt + '\', ' + msg.match[1] + ', 10000 , ' + dt + ',
\'' + msg.match[2] + '\')'

    console.log sql

    omnibusConnection.sqlCommand sql, (err, rows,
numrows, coldesc) ->

        console.log "Err=" + err

        msg.send "Journal entry added"
```

15. Start the bot and run the command "noi journal <serial#> This is a test". Check in the event viewer to see if the event's journal has been updated.

16. Stop the bot using ctrl-C and update the file noi.coffee with the following rows:

```
robot.hear /noi sev (\d) (.*)/i, (msg) ->

    sql = 'UPDATE alerts.status set Severity = ' +
msg.match[1] + ' where Serial=' + msg.match[2]

    omnibusConnection.sqlCommand sql, (err, rows,
numrows, coldesc) ->

        console.log "Err=" + err

        if numrows == 0

            msg.send "I can't find any event with a
serial number of " + msg.match[2]

            msg.send "Perhaps the event has been
closed already? Otherwise, try another serial number."

        else

            msg.send "The event severity has been
set to " + msg.match[1]
```

17. Start the bot and run the command "noi sev <serial#> 5". Check in the event viewer to see if the event's priority has been changed to Critical/Red.

18. Stop the bot using ctrl-C and update the file noi.coffee with the following rows:

```
robot.hear /noi resolve (.*)/i, (msg) ->

    sql = 'UPDATE alerts.status set Severity =
0,CASE_ICD_Status=\'RESOLVED\' where Serial=' +
msg.match[1]

        omnibusConnection.sqlCommand sql, (err, rows,
numrows, coldesc) ->

            console.log "Err=" + err

            msg.send "Event resolved (Severity set to
0)"
```

19. Start the bot and run the command "noi resolve <serial#>". Check in the event viewer to see if the event's priority has been changed to Clear/Green.

20. Stop the bot using ctrl-C and update the file noi.coffee with the following rows:

```
robot.hear /noi show (.*)/i, (msg) ->

    query = 'SELECT Serial, Summary, Severity, Node,
Acknowledged, LastOccurrence from alerts.status where
Serial=' + msg.match[1]

        omnibusConnection.query query, (err, rows,
numrows, coldesc) ->

            console.log err

            i = 0

            msg.send

                text : rows[i].Summary

                attachments: [ {

                    title: 'Node'

                    text: rows[i].Node

                    fields: [ {

                        short : true
```

```
        title : 'Severity'  
        value : rows[i].Severity  
    }, {  
        title : 'Acknowledged'  
        value : rows[i].Acknowledged  
        short : true  
    }, {  
        title : 'LastOccurrence'  
        value :  
rows[i].LastOccurrence  
        short : true  
    }, {  
        title : 'Serial'  
        value : rows[i].Serial  
        short : true  
    } ]  
} ]  
  
username:  
process.env.HUBOT_SLACK_BOTNAME  
as_user: true
```

21. Start the bot and run the command "noi show <serial#>"

Advanced formatting of the event

References and extra material

To be done