InterConnect
2017

March 19–23
MGM Grand &
Mandalay Bay
Las Vegas, NV

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.

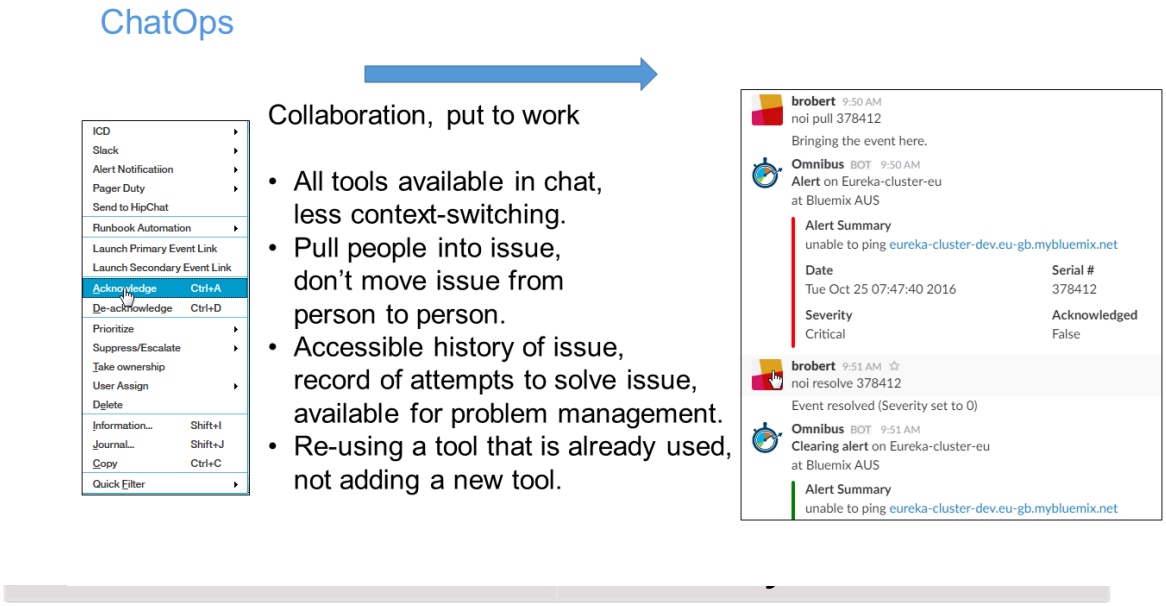
Table of Contents

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



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.

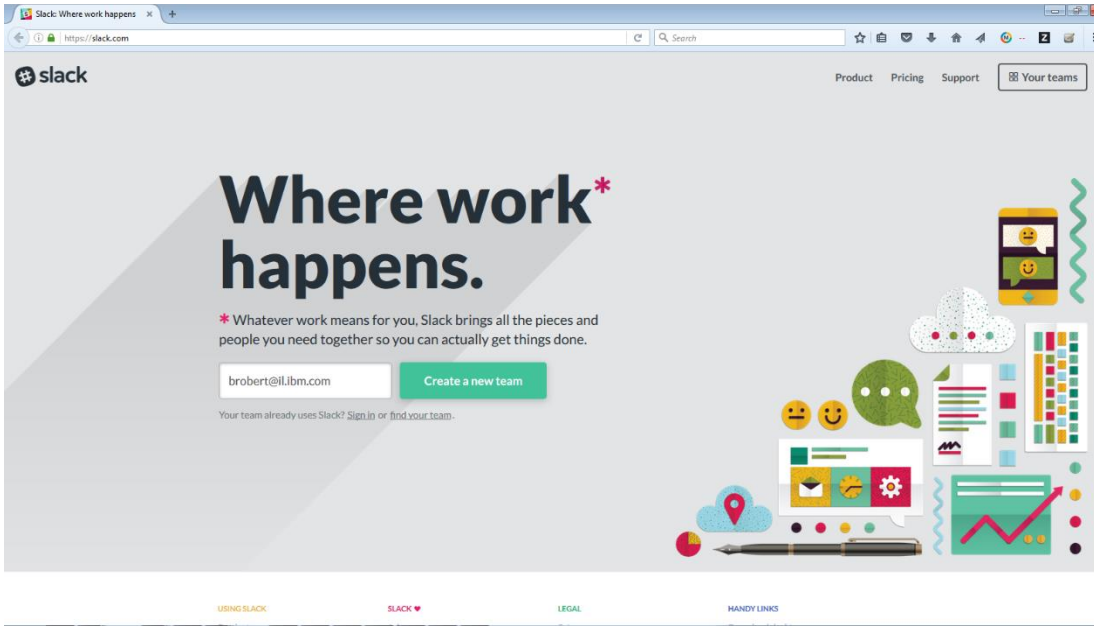
Create Slack Team

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

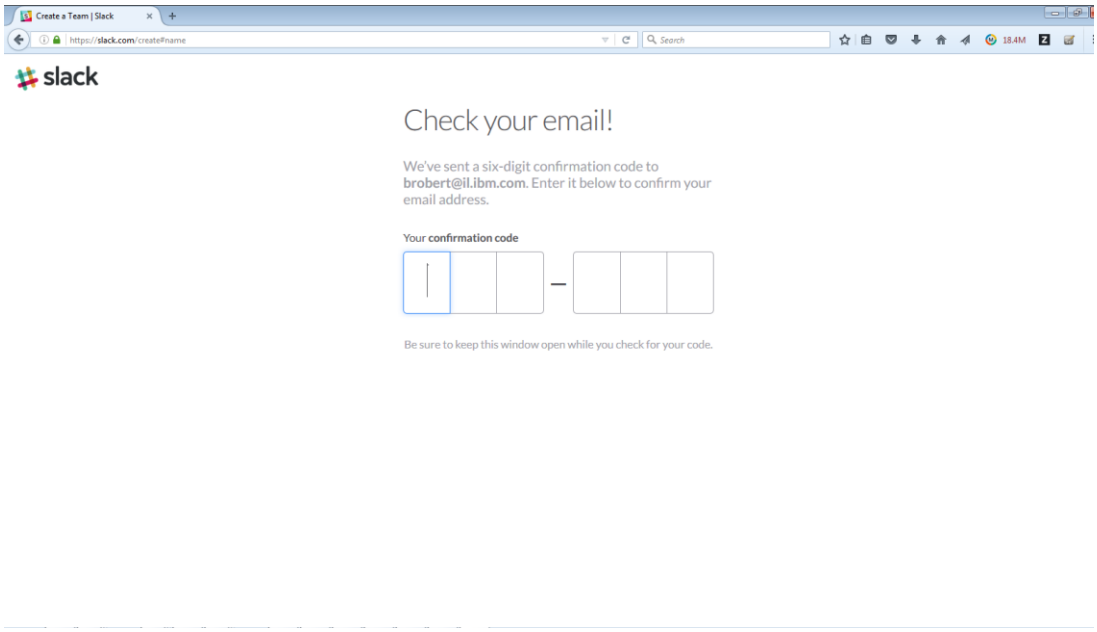
We will perform the following steps:

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.

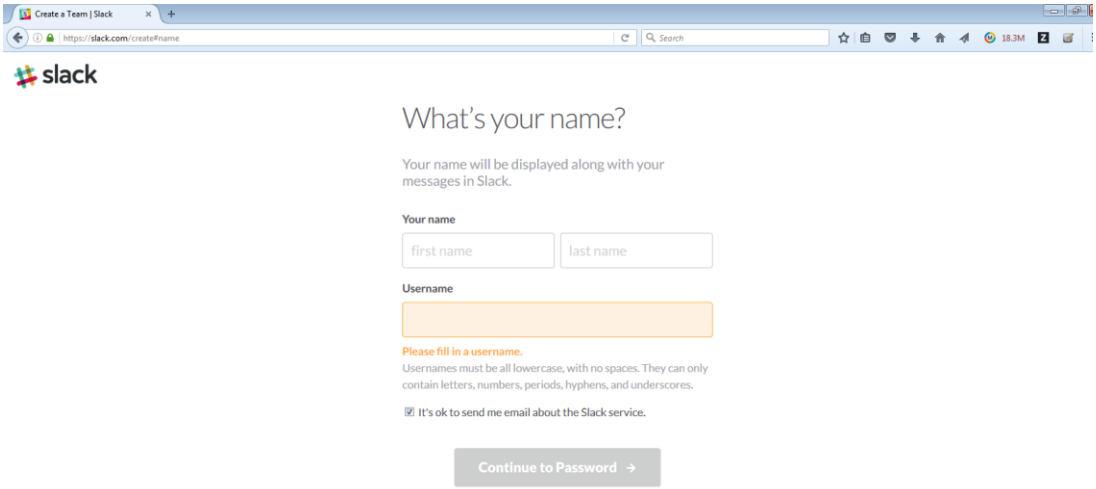
1. Go to slack main page www.slack.com and create a new team



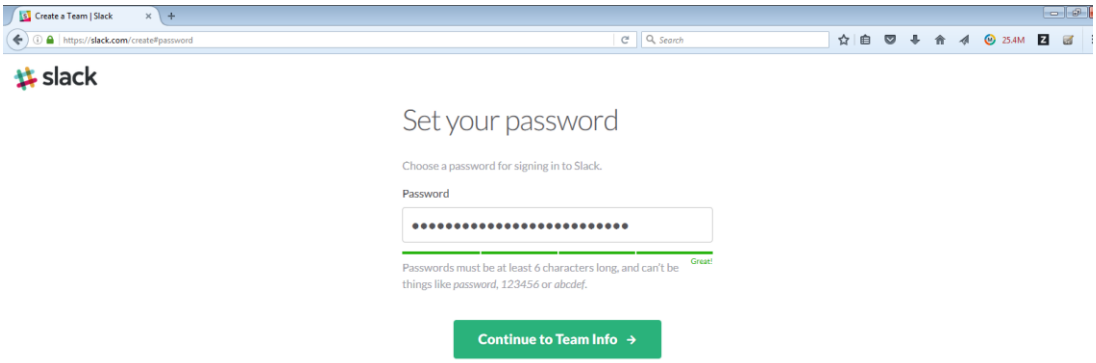
2. Check your email and verify your team



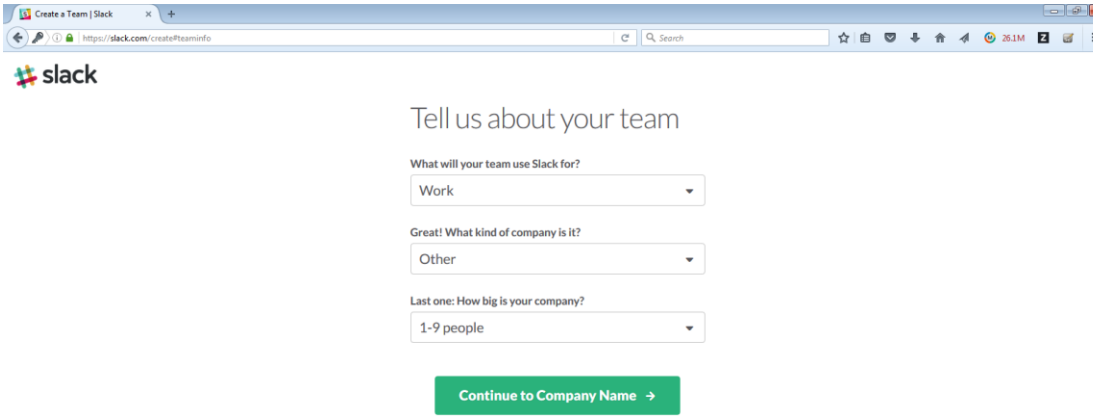
3. Enter your name



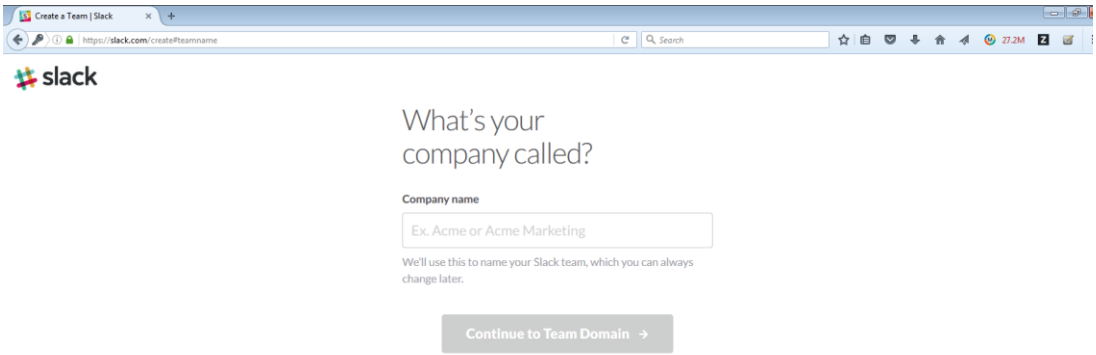
4. Set your password



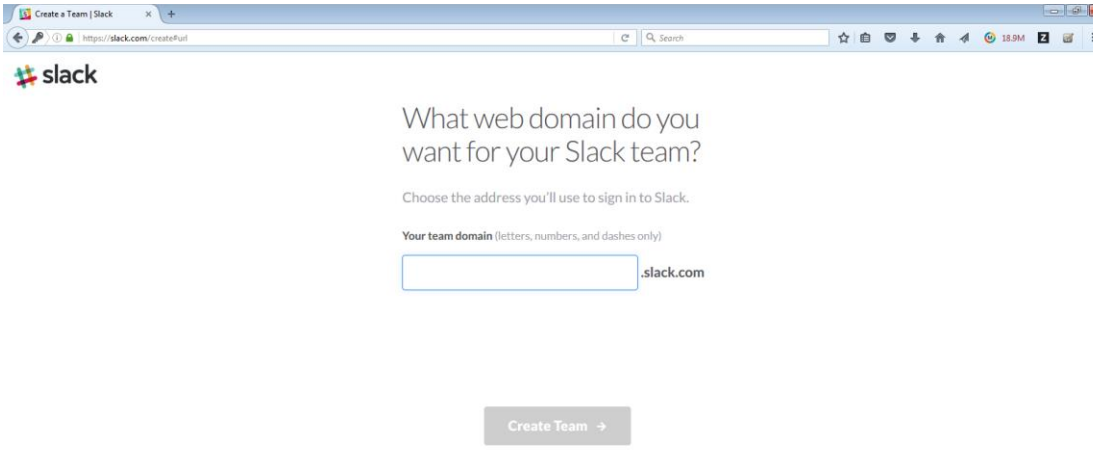
5. Fill in basic details about the team



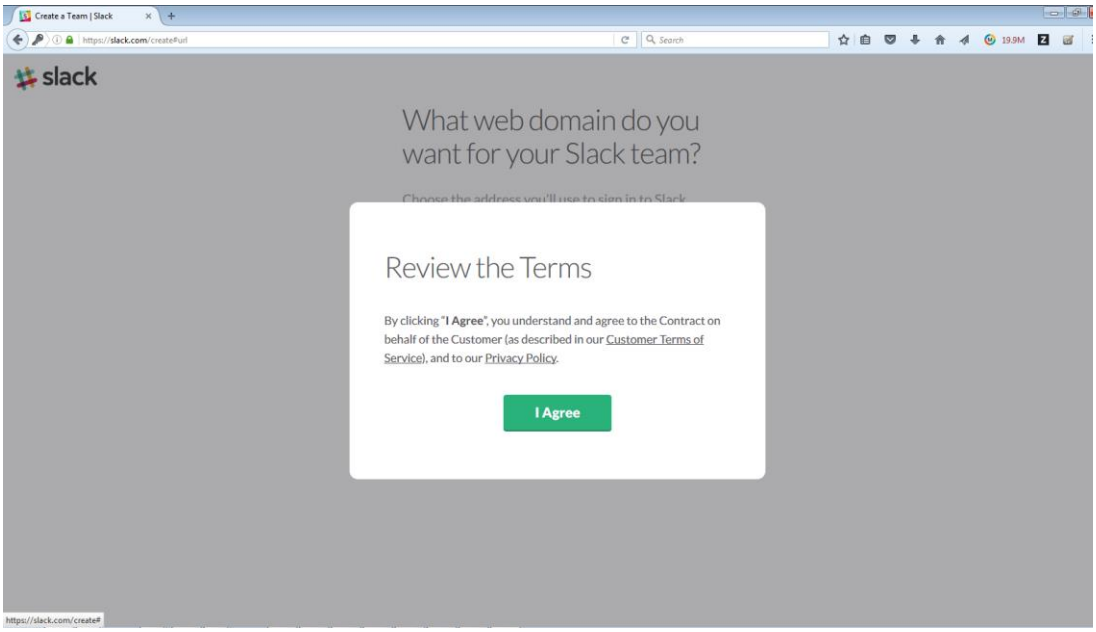
6. Fill in company name



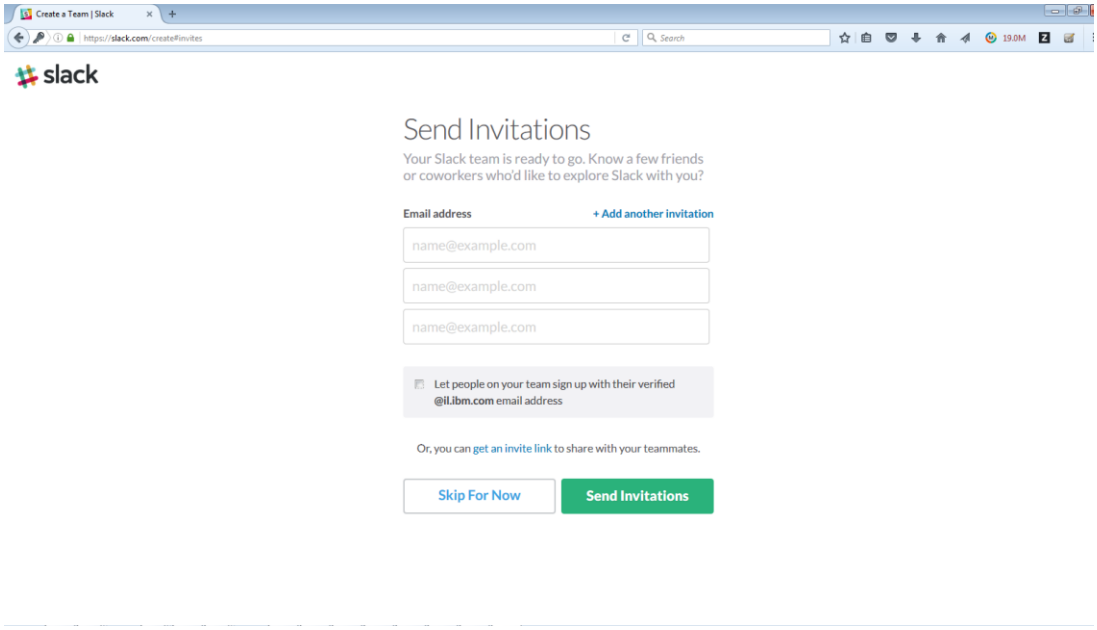
7. Choose your Slack domain



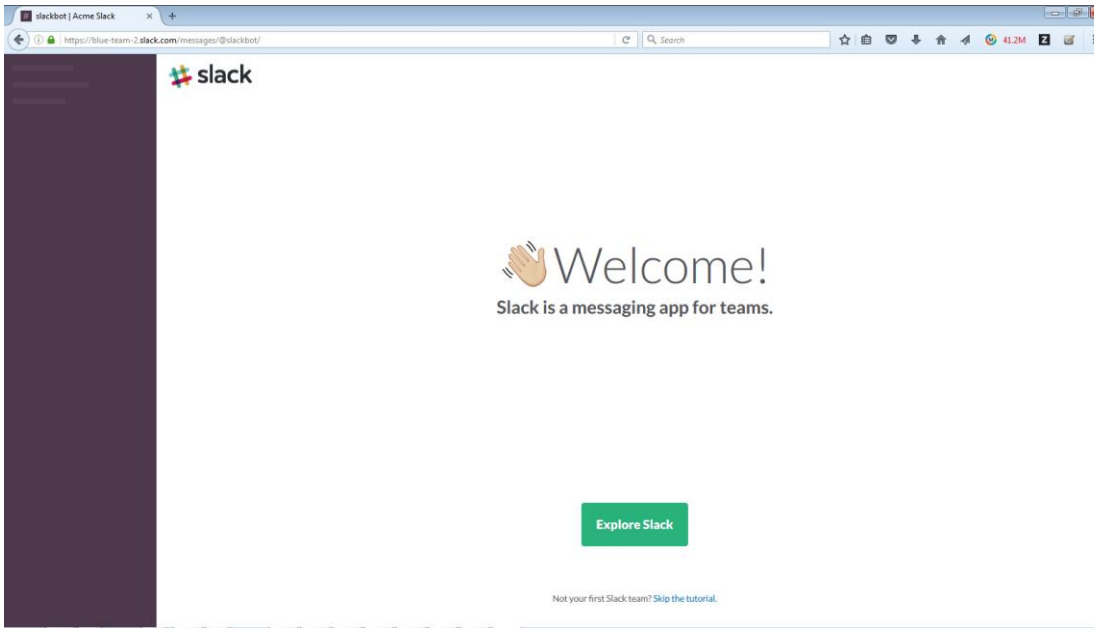
8. Agree to the terms (you have no choice, really!)



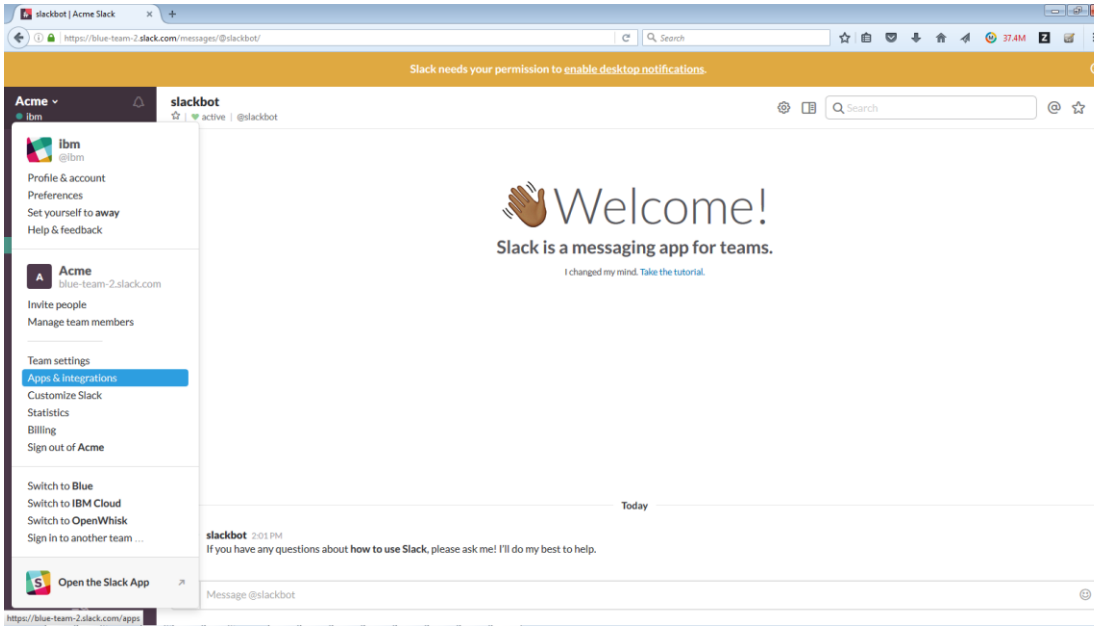
9. You may skip this step



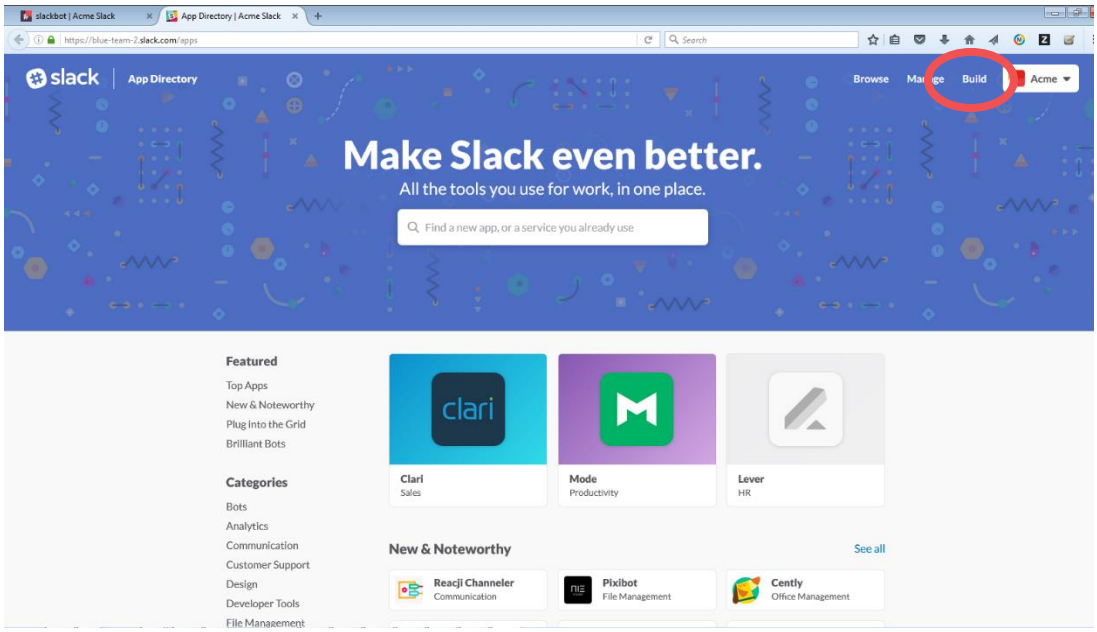
10. You have logged into your new Slack



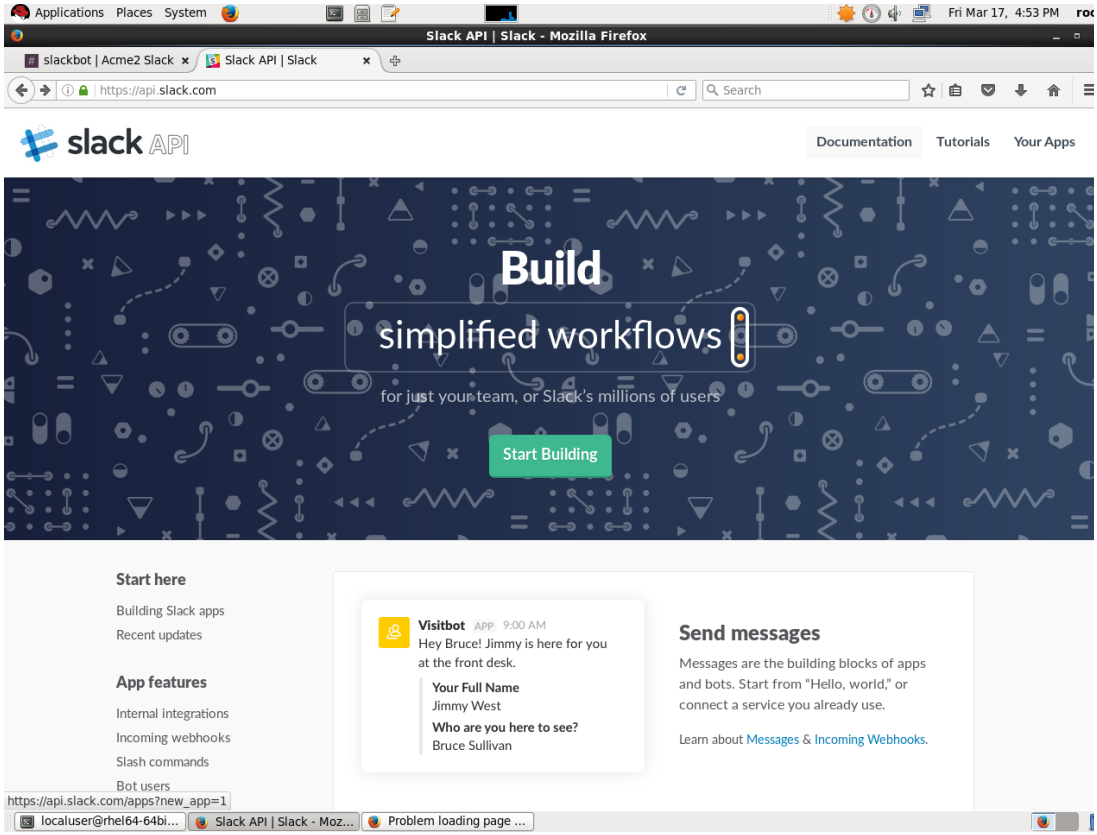
11. Open the menu to "Apps and Integrations"



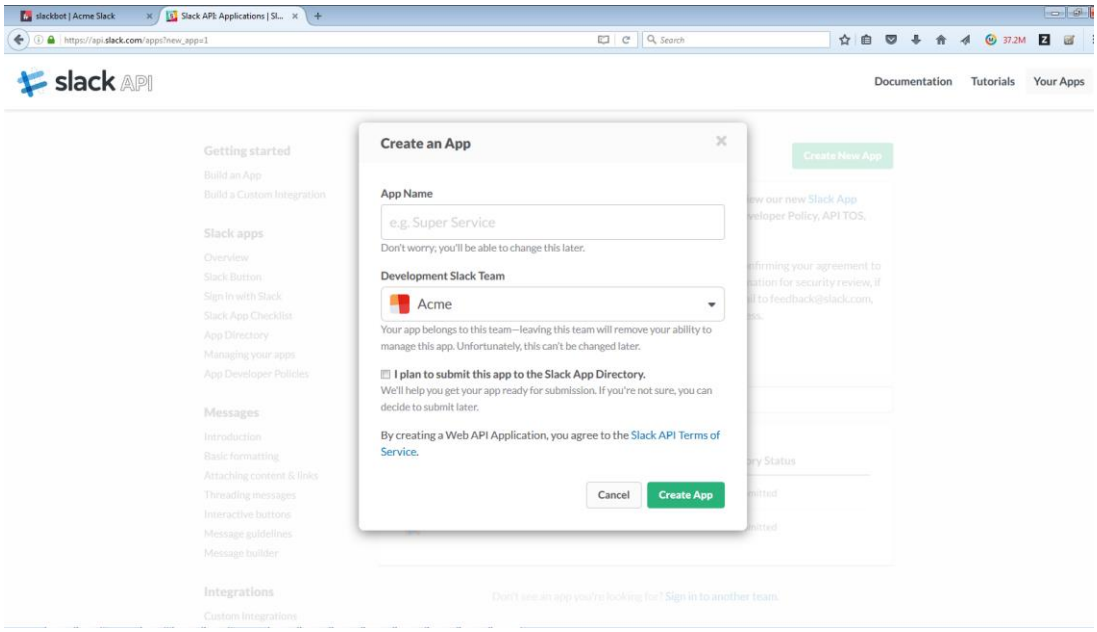
12. In the opened screen, choose the menu option Build



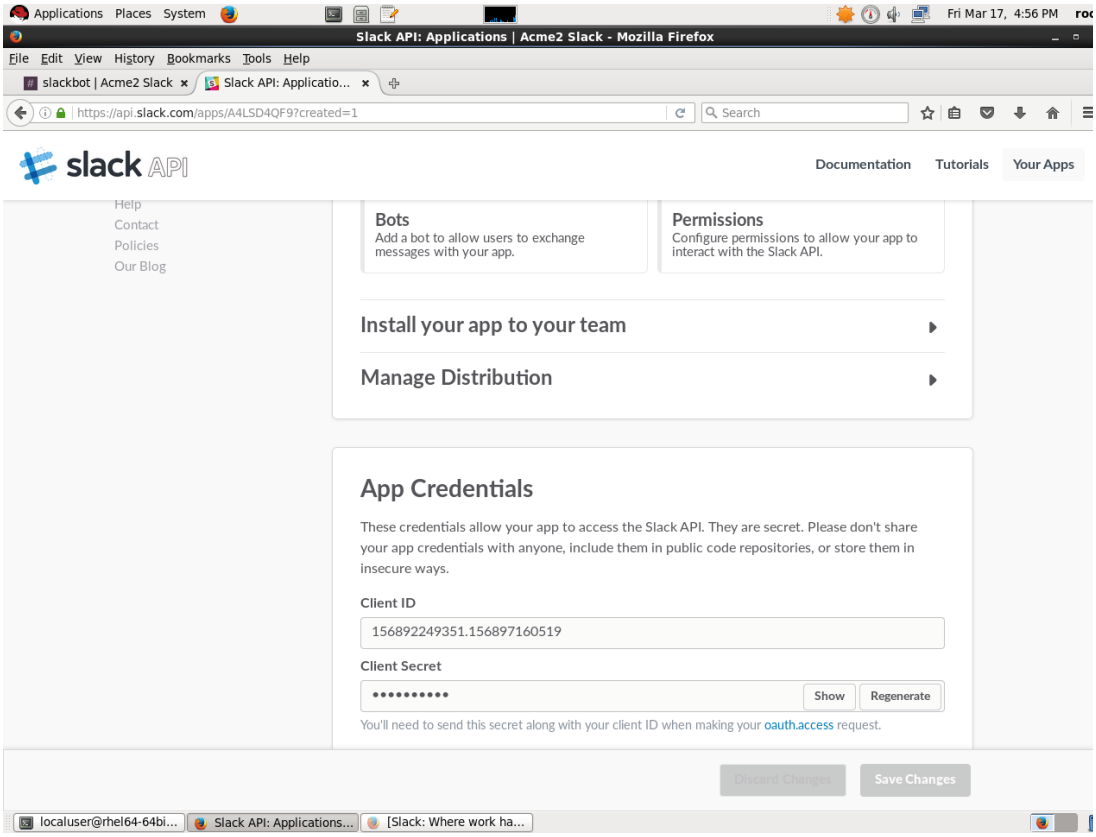
13. Select "Start Building"



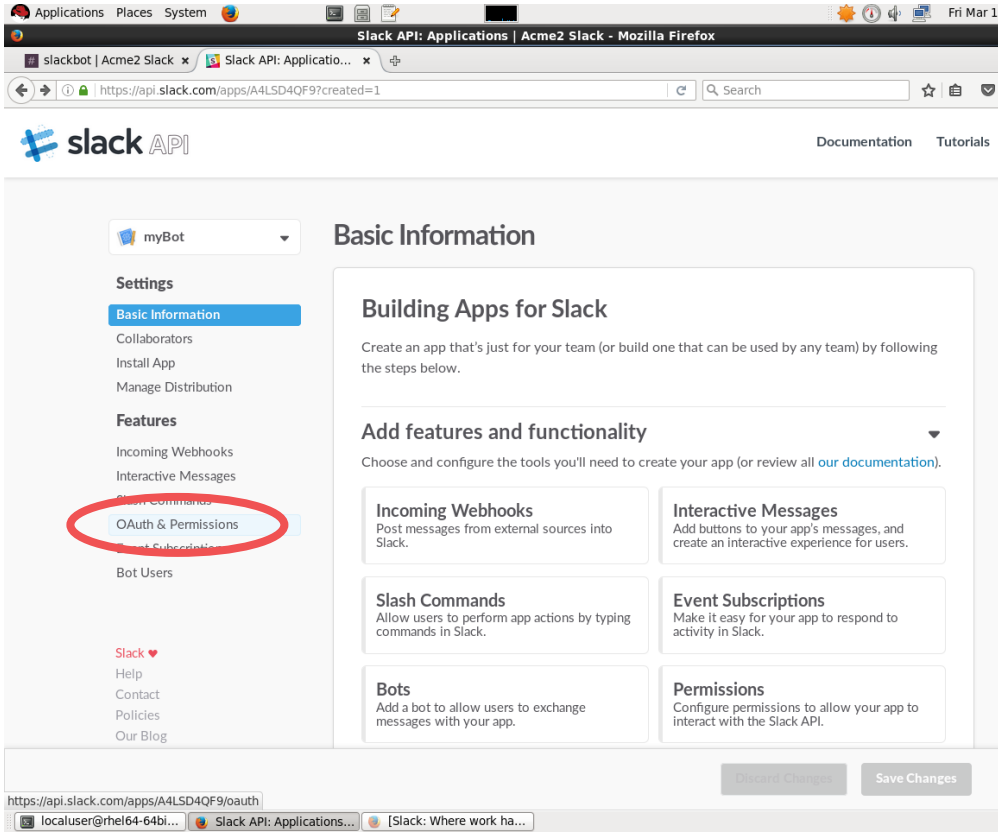
14. Choose a name for your app (myBot, for example)



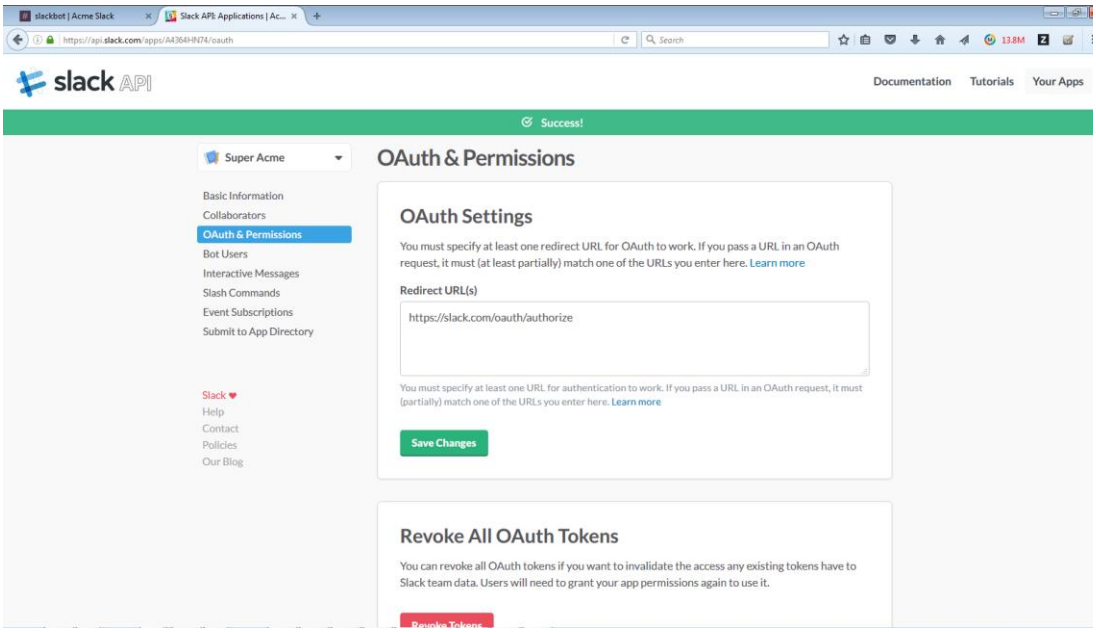
15. Congratulations – you have an app!
    Scroll down and record the Client ID and Client Secret



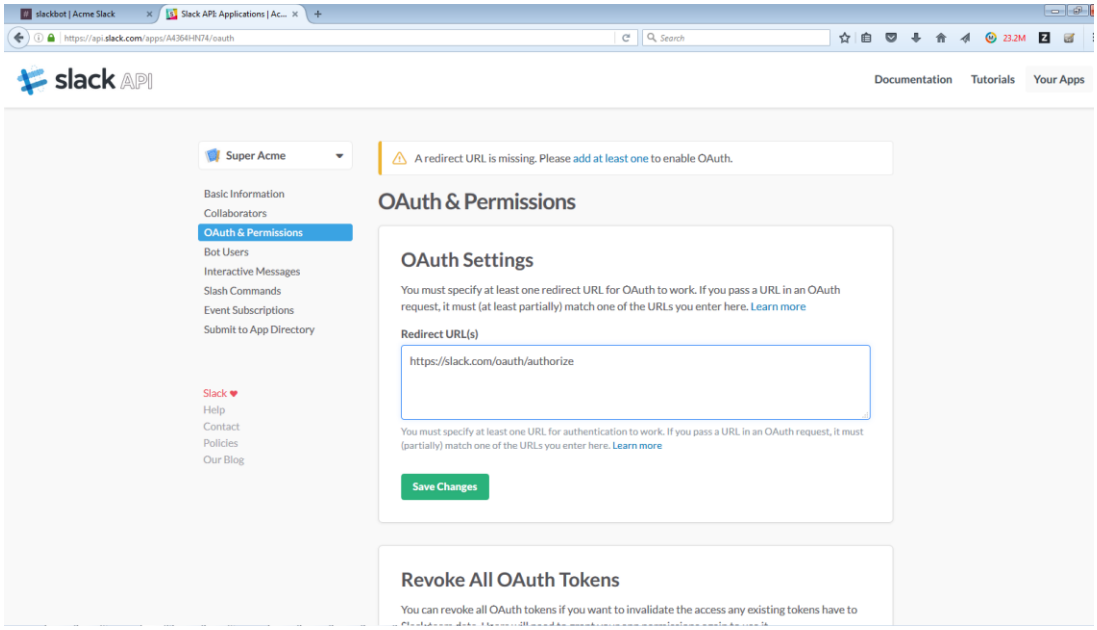
16. Choose OAuth & Permissions from the menu



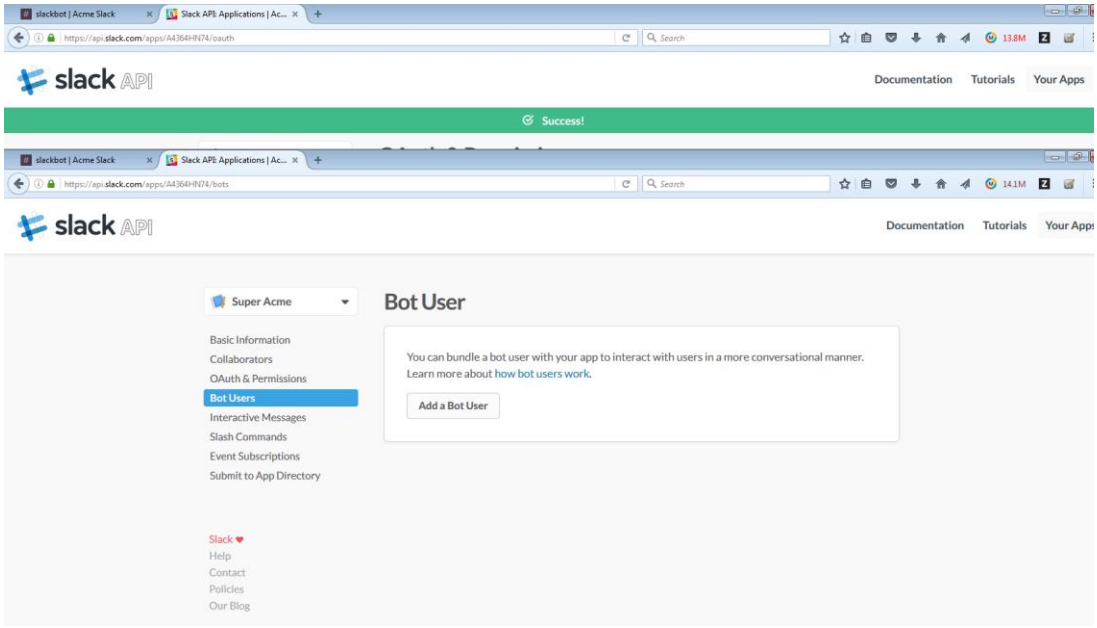
17. Enter https://slack.com/oauth/authorize in the OAuth settings



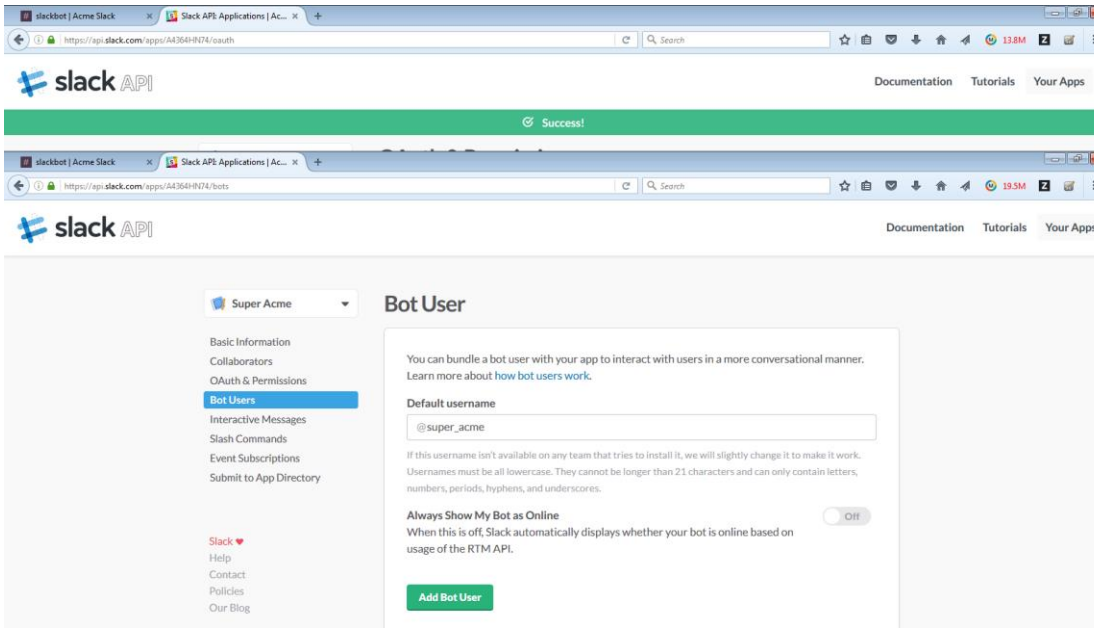
18. Success!


slackbot | Acme Slack
Slack API: Applications | Ac...
slack API
Documentation
Tutorials
Your Apps
Super Acme
A redirect URL is missing. Please add at least one to enable OAuth.
Basic Information
Collaborators
OAuth & Permissions
Bot Users
Interactive Messages
Slash Commands
Event Subscriptions
Submit to App Directory
Slack
Help
Contact
Policies
Our Blog
OAuth & Permissions
OAuth Settings
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. Learn more
Redirect URL(s)
https://slack.com/oauth/authorize
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
Save Changes
Revoke All OAuth Tokens
You can revoke all OAuth tokens if you want to invalidate the access any existing tokens have to

19. Create a bot user

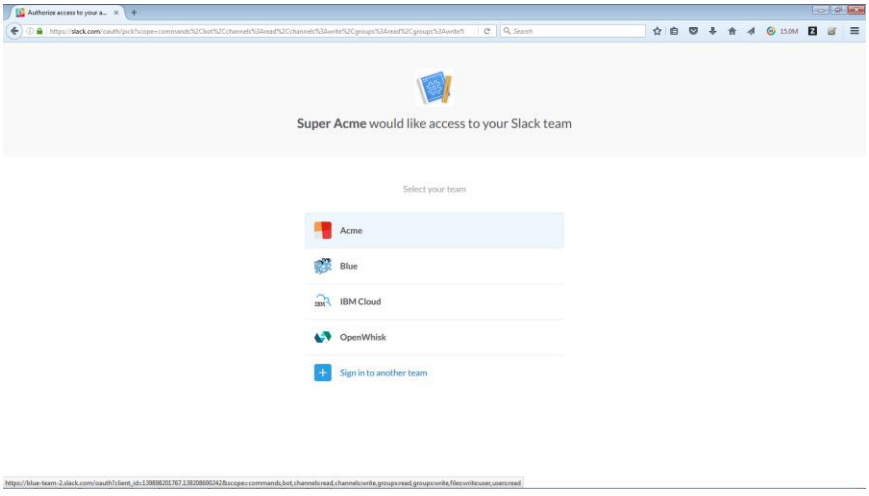


20. Success!

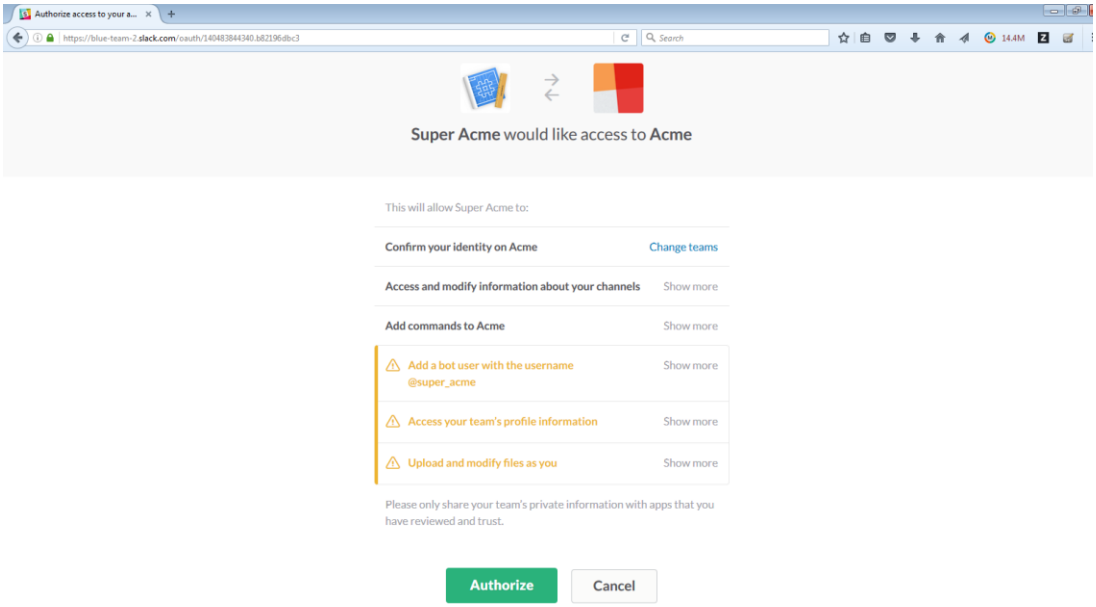


21. Download and edit the file https://github.com/RobertJBarron/2017-InterConnect-3060/blob/master/SlackButtonForOAUTH_template.html. Edit the file and enter the clientID from step 15 and open the file in a browser

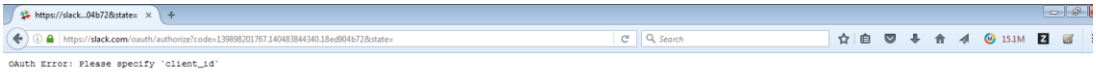
22. After pressing the button, choose your Slack team to give access to (this step depends on your existing Slack settings, you may not see it)



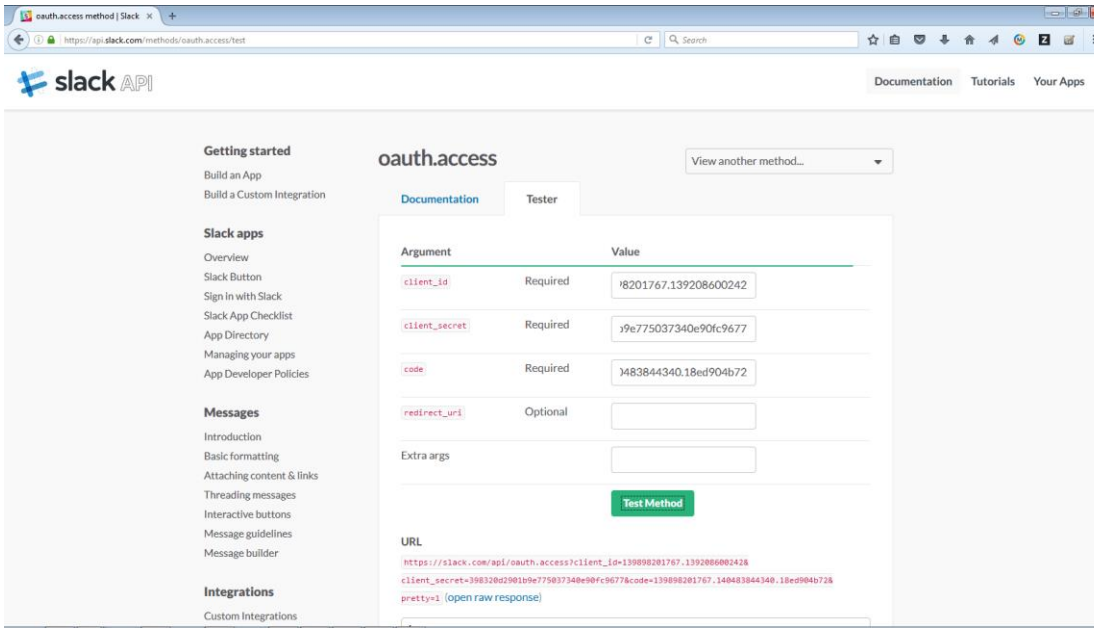
23. Authorize the permissions



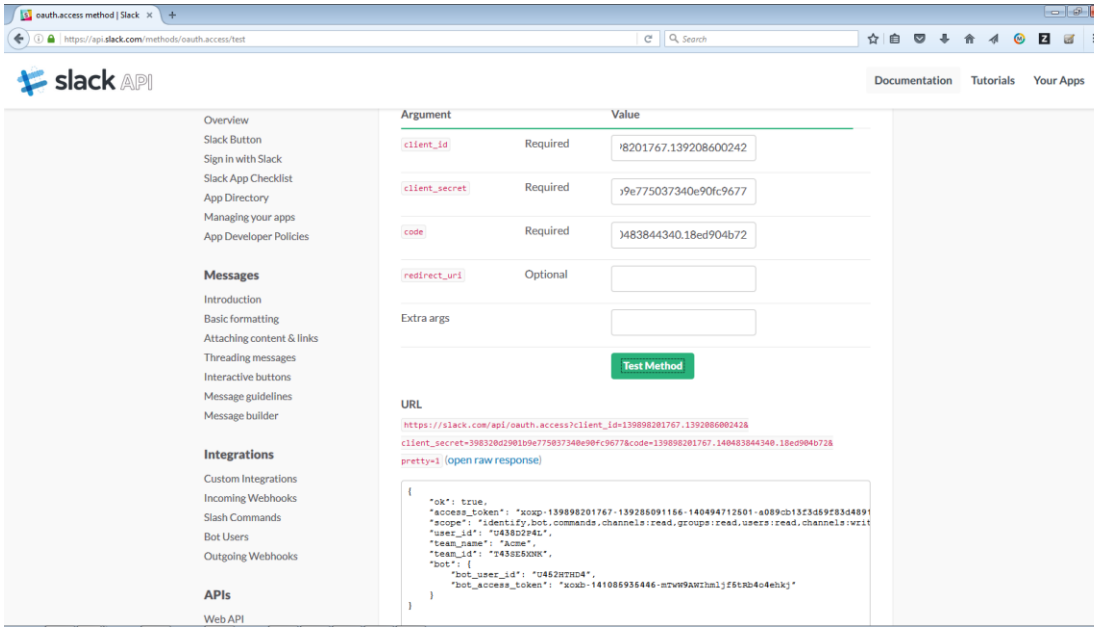
24. Record the "errorcode" in the browser URL.
    This is the section of text between "code=" and "&state"



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)



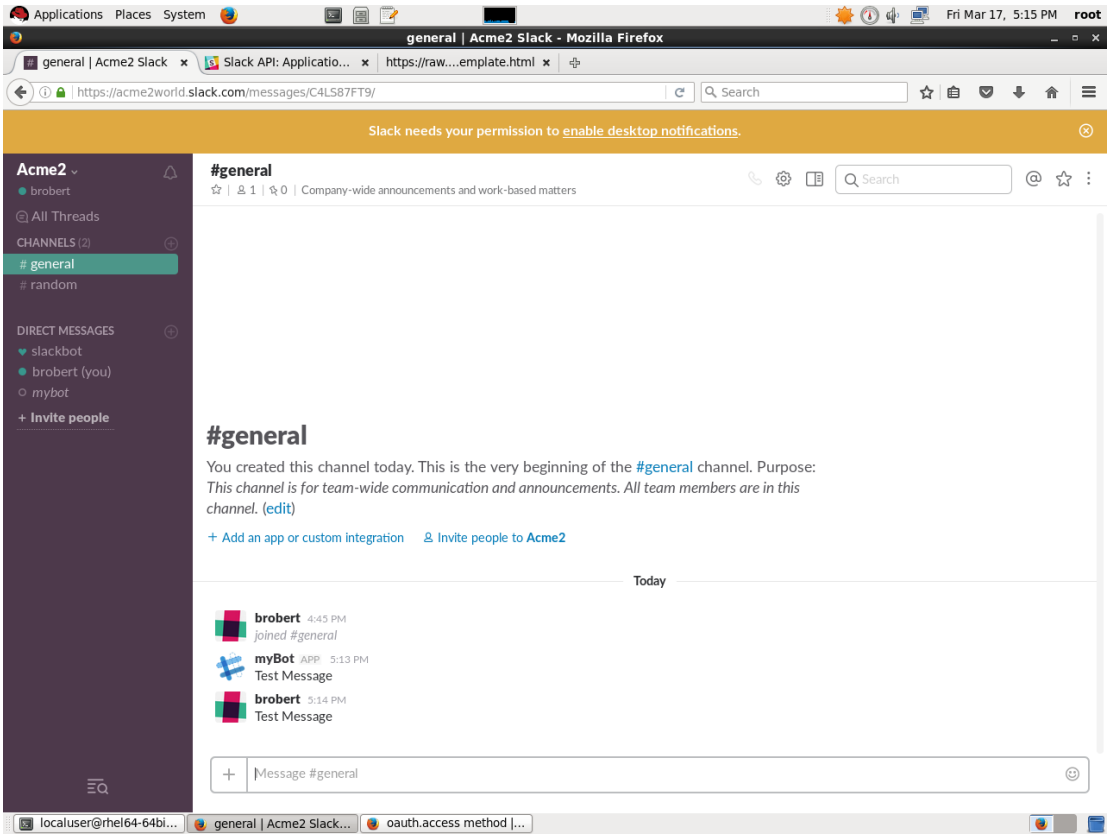
26. Record the access tokens you receive (your own and the bot's)



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.

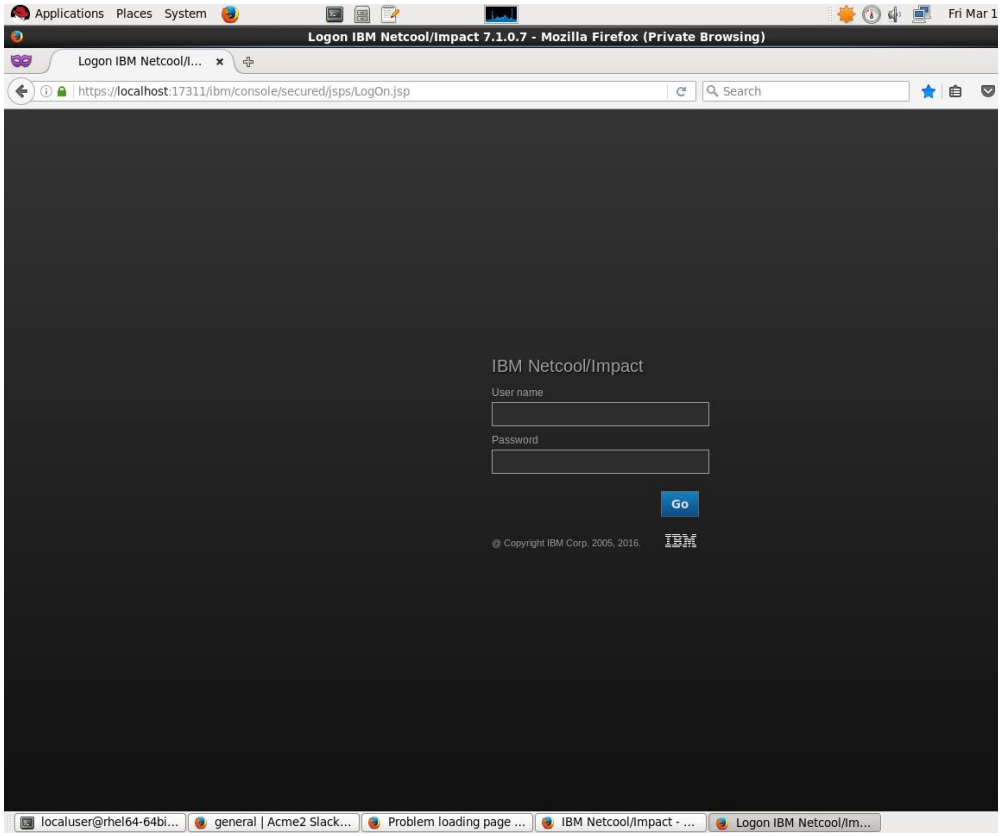
1. Export the Slack server certificate using the following command:
```
echo -n | openssl s_client -connect slack.com:443 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/slack.cert
```

2. Once you have the 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/security/trust.jks -storepass impactadmin
```

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

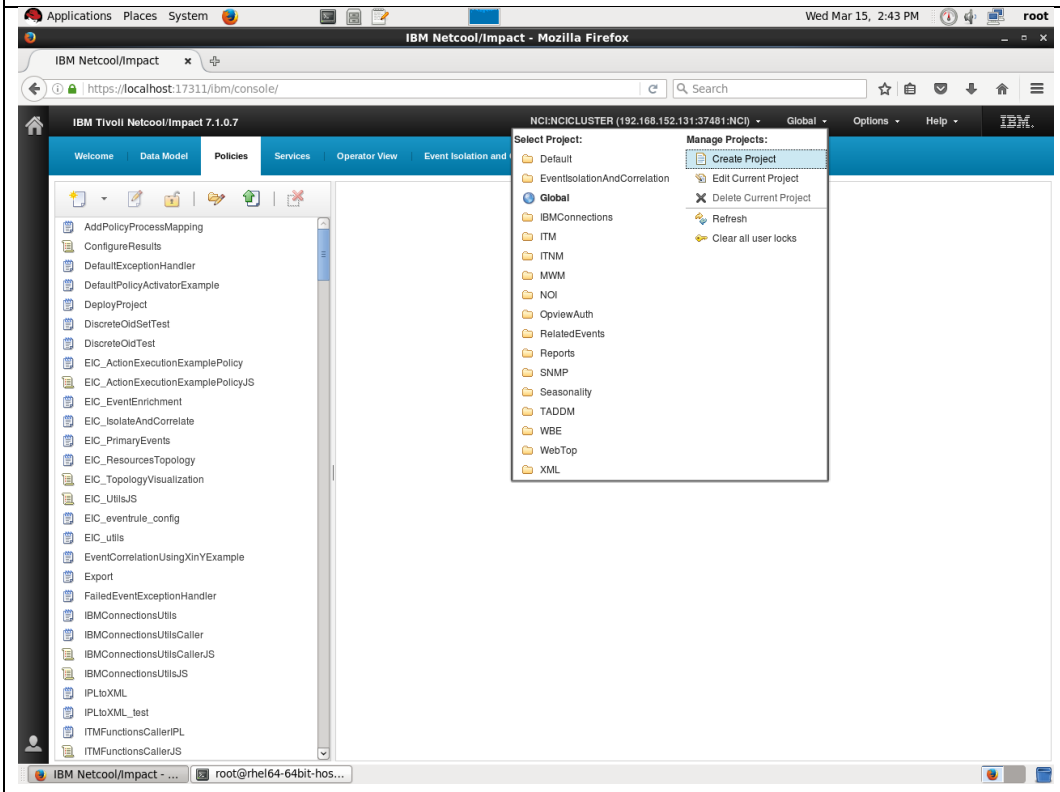
3. Restart Impact using the commands
```
/opt/IBM/tivoli/impact/bin/stopImpactServer.sh
/opt/IBM/tivoli/impact/bin/startImpactServer.sh
```

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)

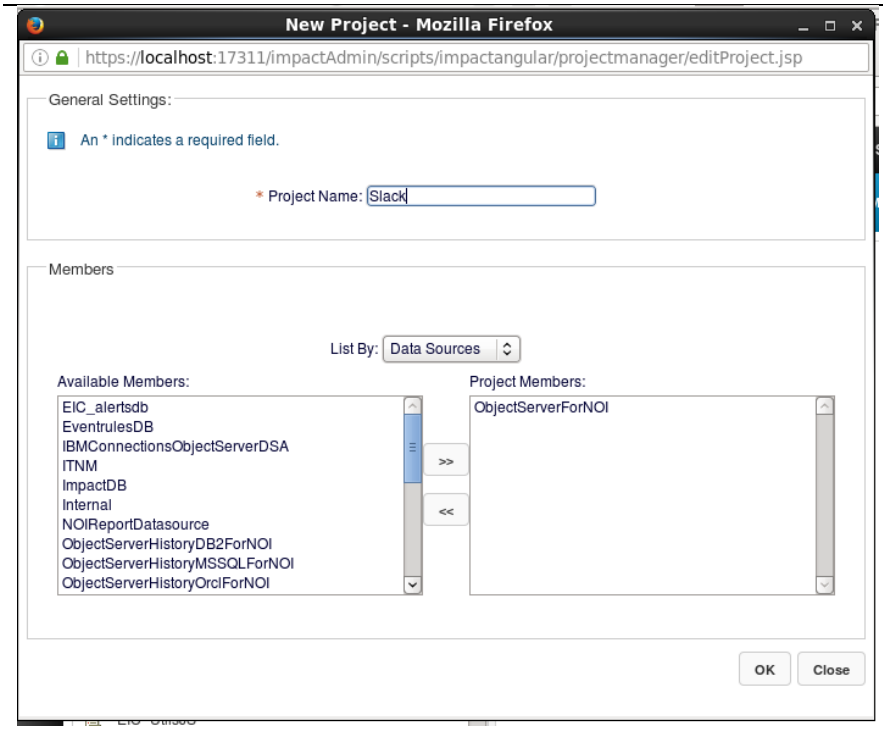


User/password is impactadmin/impactadmin

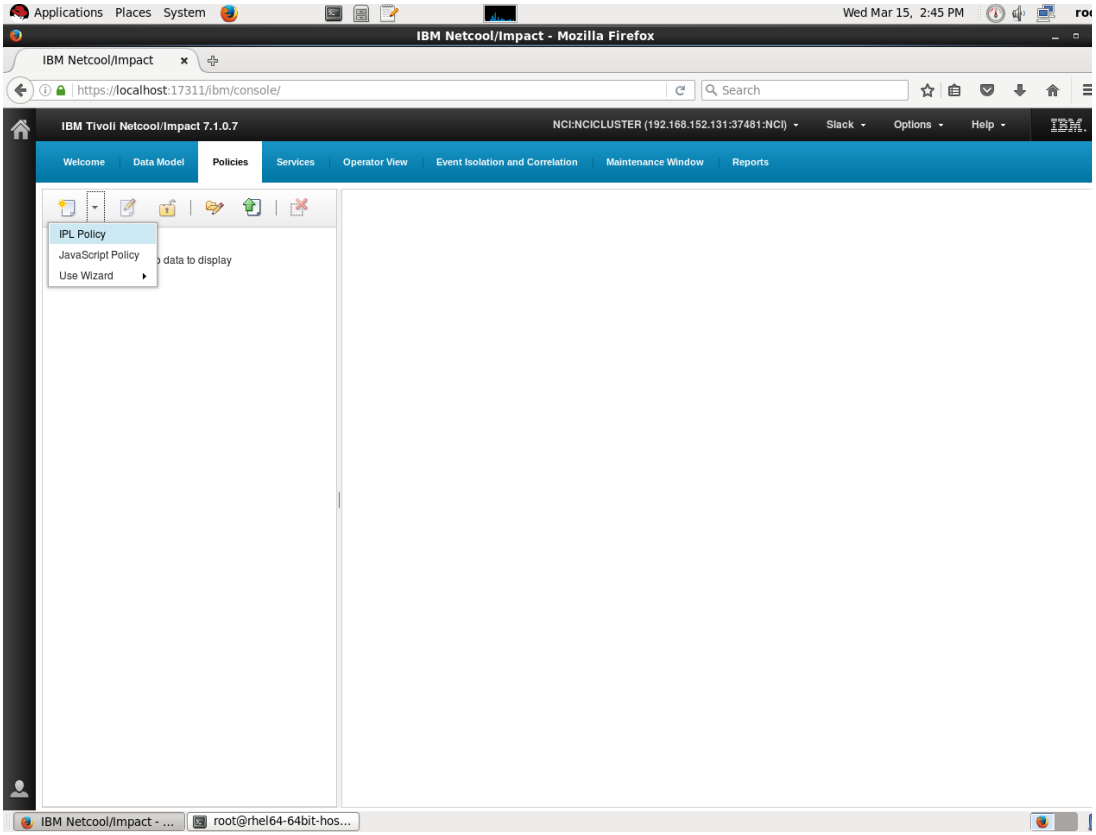
5. Click "Global" and then "Create Project"



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



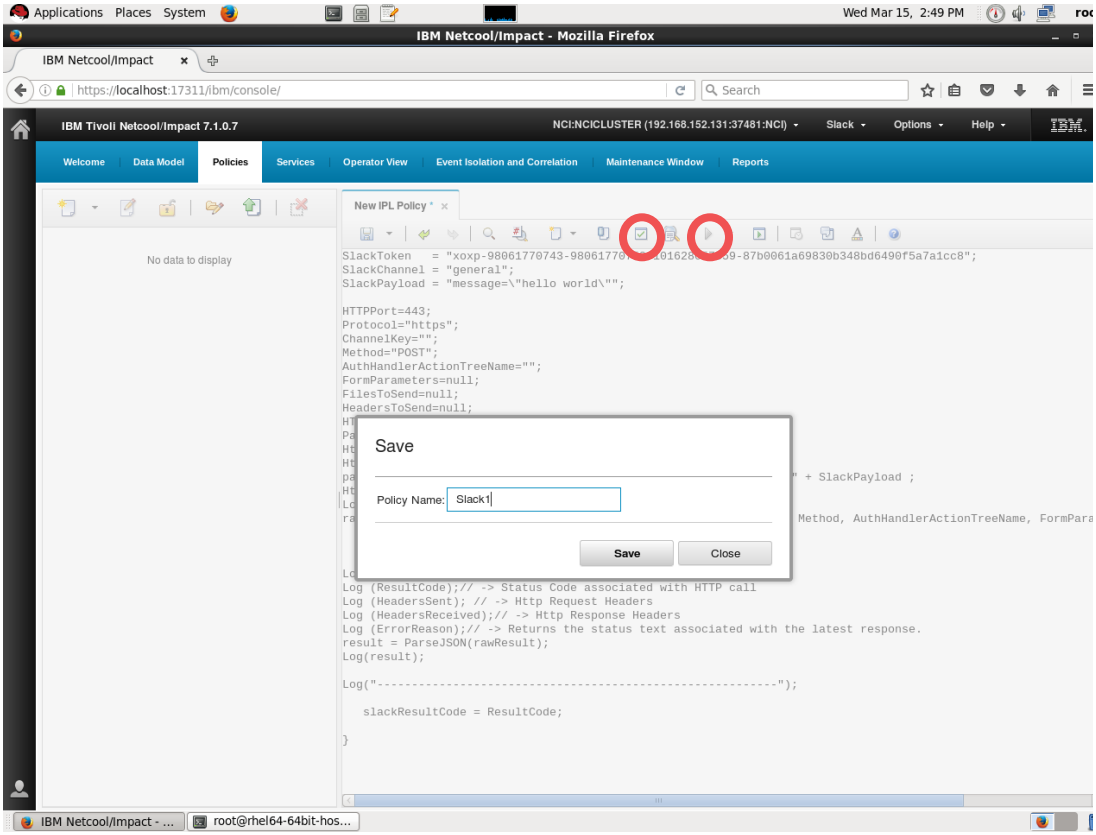
7. Create a new IPL policy



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.



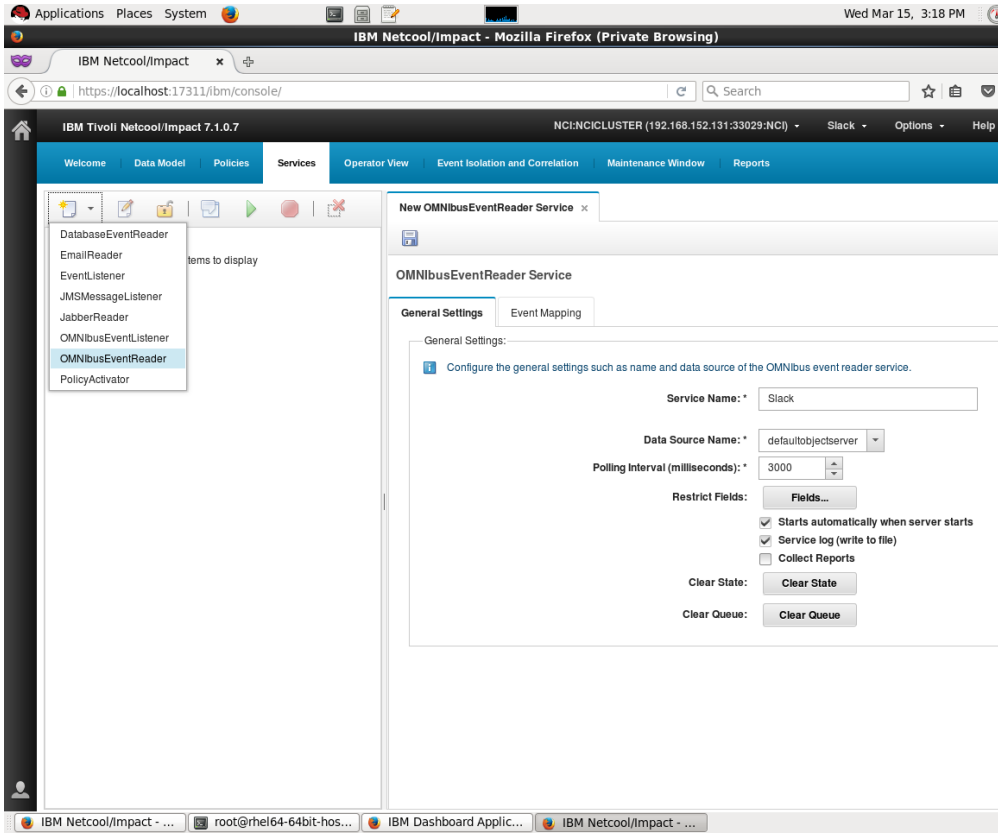
9. Check the contents of the policy log file at : /opt/IBM/tivoli/impact/logs/

Sending a real event from Impact

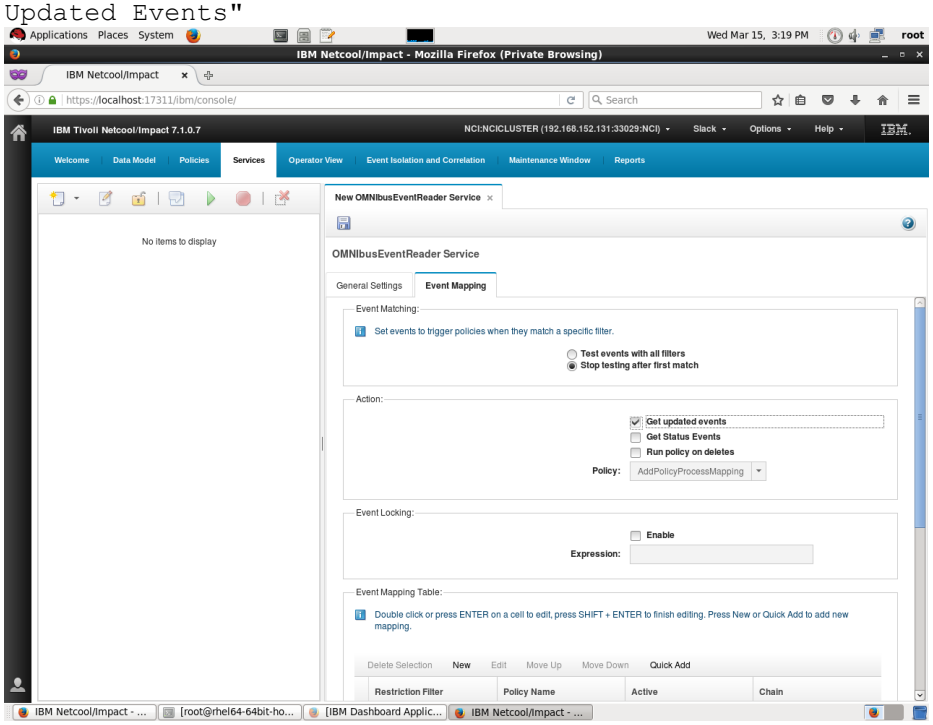
1. Create a file and instert the following SQL commands

```
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
```

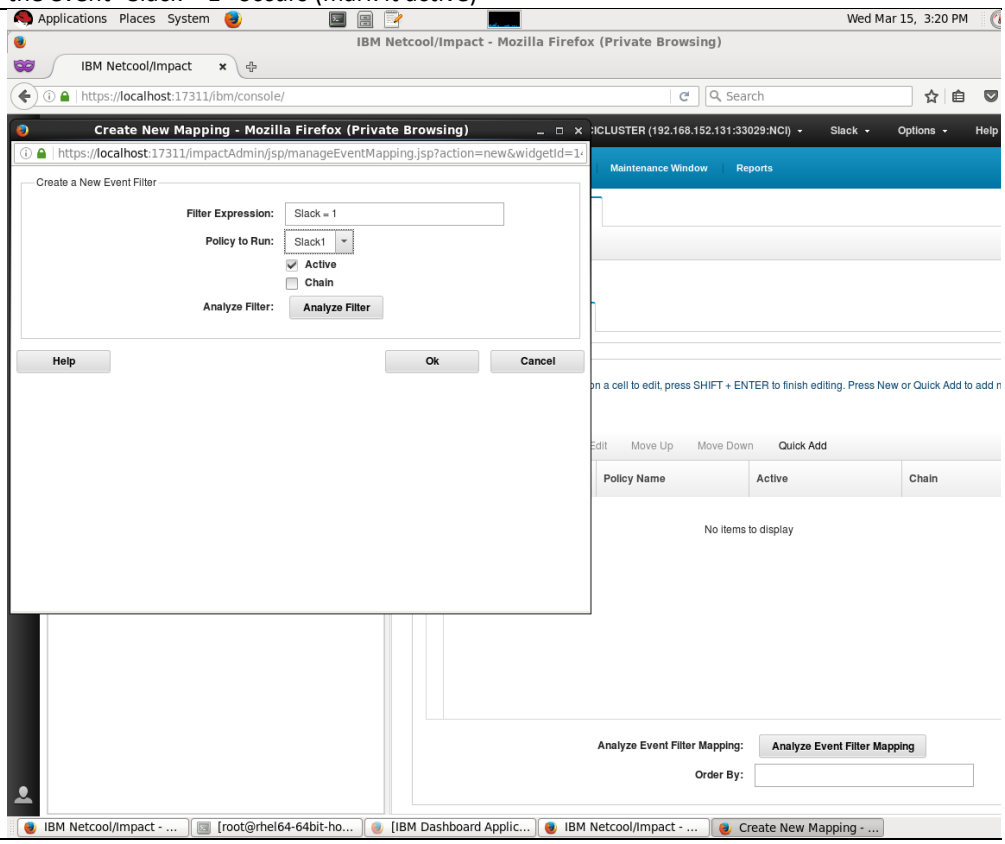
2. In Impact, open the "Services" tab and create a new service called "Slack"



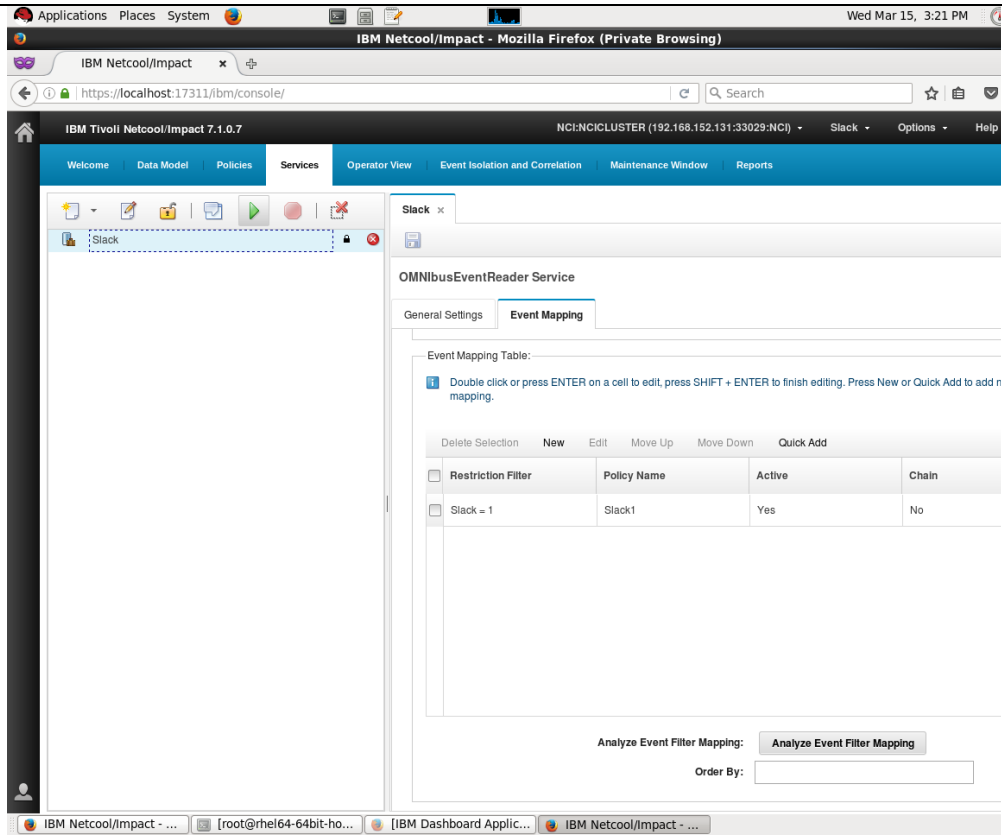
3. Move to the tab "Event Mapping" and check the option "Get Updated Events"



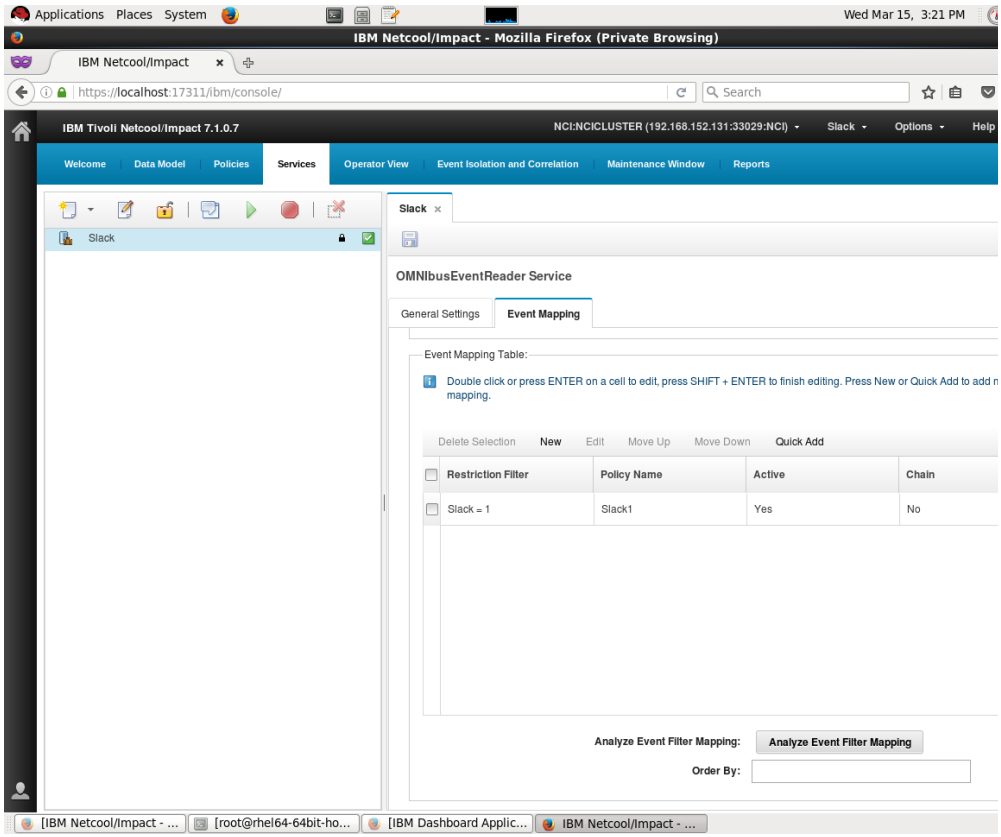
4. Add a new entry to the "Event Mapping Table" which will run the policy you created when the event "Slack = 1" occurs (mark it active)



5. Save the service



6. Start the Service

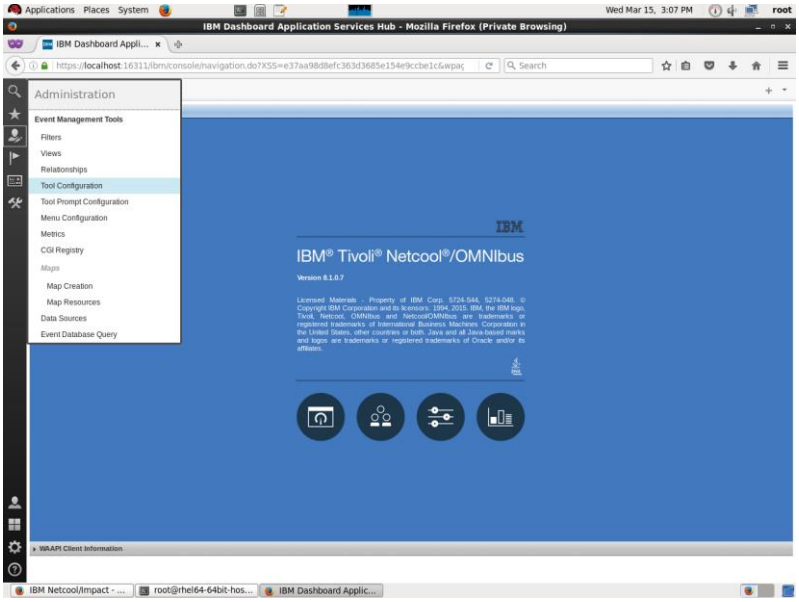


7. Reopen the Slack policy that you created in the previous step and modify it in the following ways:

   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.

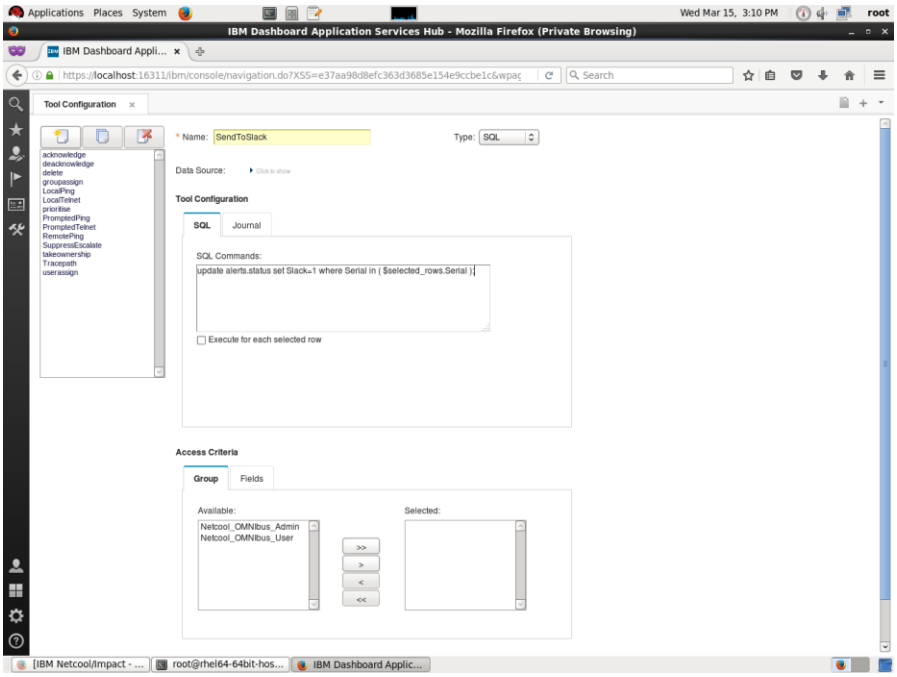
8. Login to DASH at http://localhost:16311/ibm/console with the user/password of ncoadmin/nocadmin (note that you cannot be logged into to DASH and Impact during the same session – you will either need to logout of Impact or open a private tab for the DASH session)

9. Open the "Tool config" menu

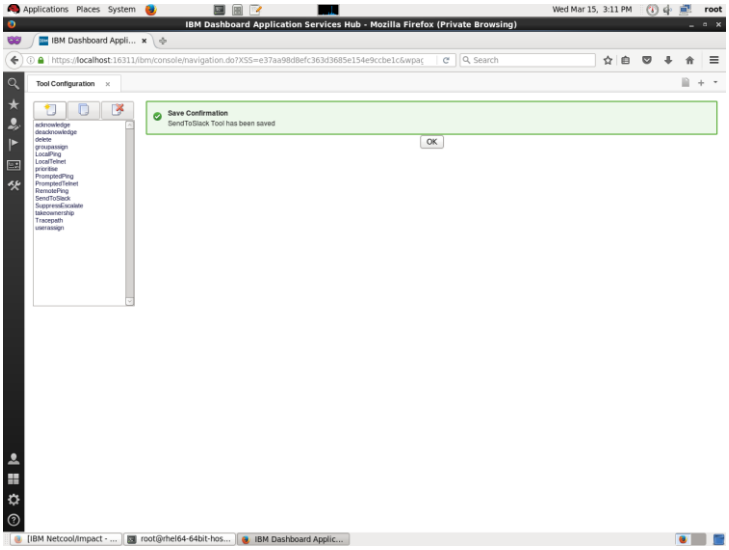


10. Create a new SQL tool called "SendToSlack" and enter the following command :

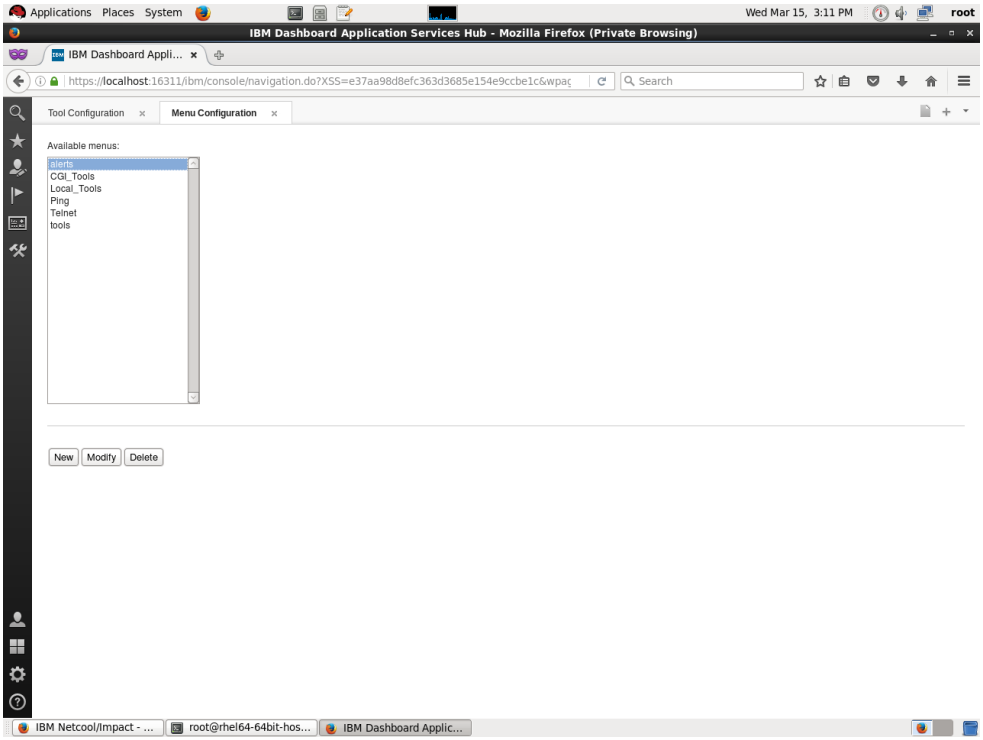
```
update alerts.status set Slack=1 where Serial in
($selected_rows.Serial);
```



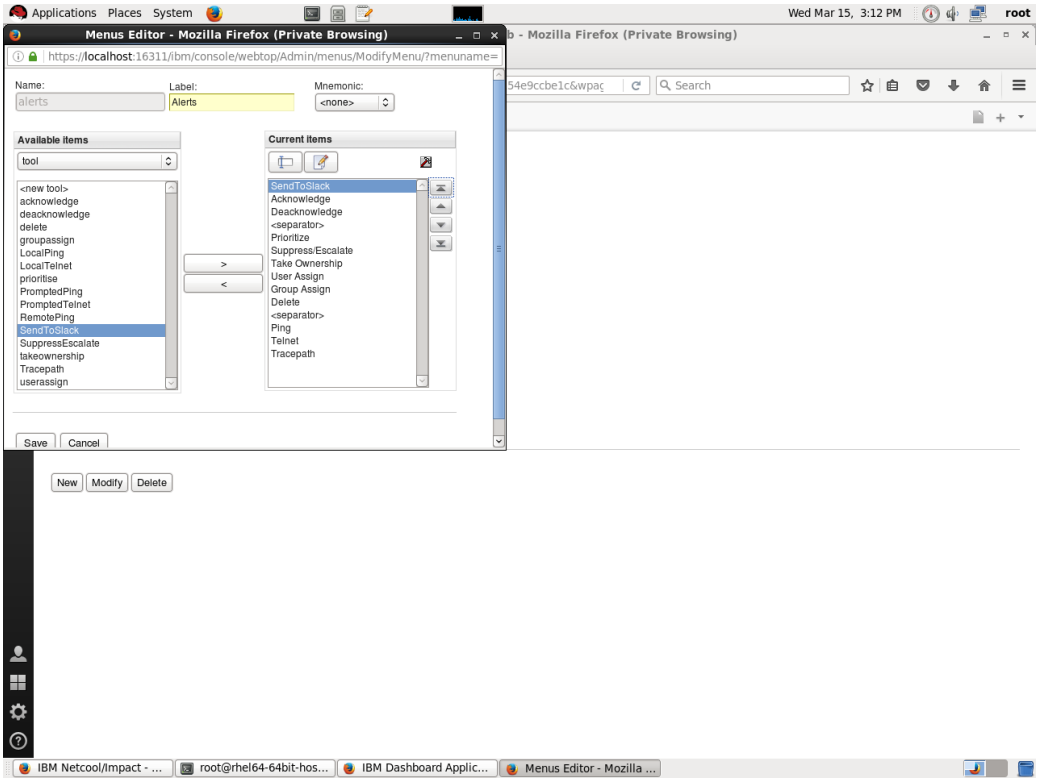
11. Save the tool



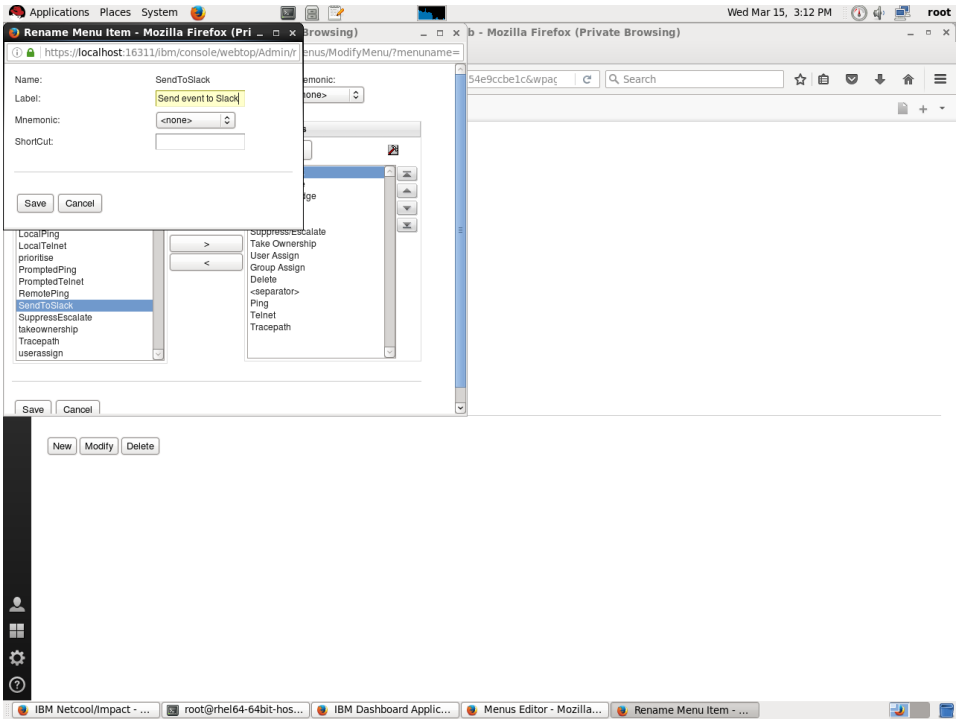
12. Open the "Menu Configuration" tab and choose the "alerts" menu to modify



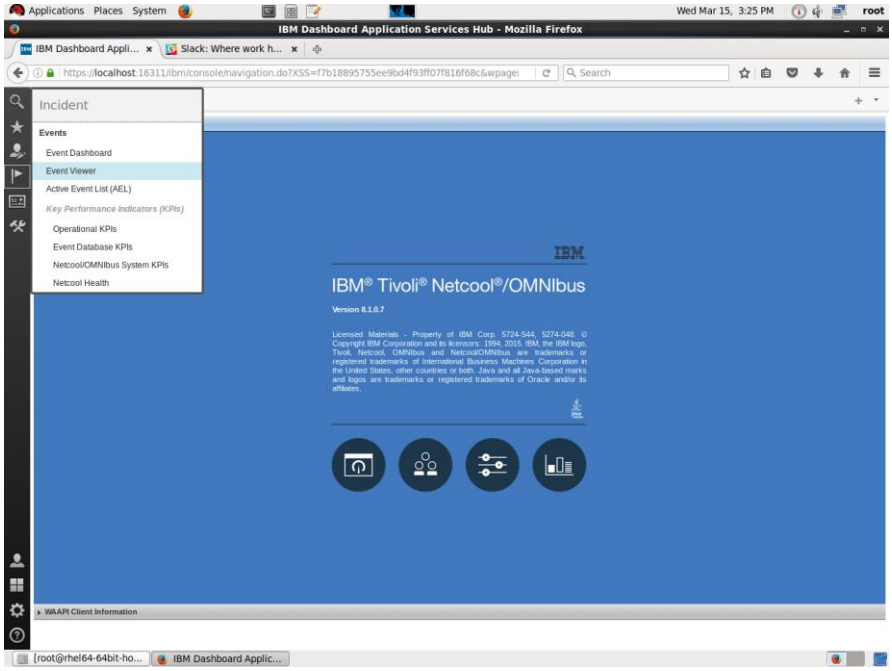
13. Modify the "alerts" menu by moving the SendToSlack tool to the right



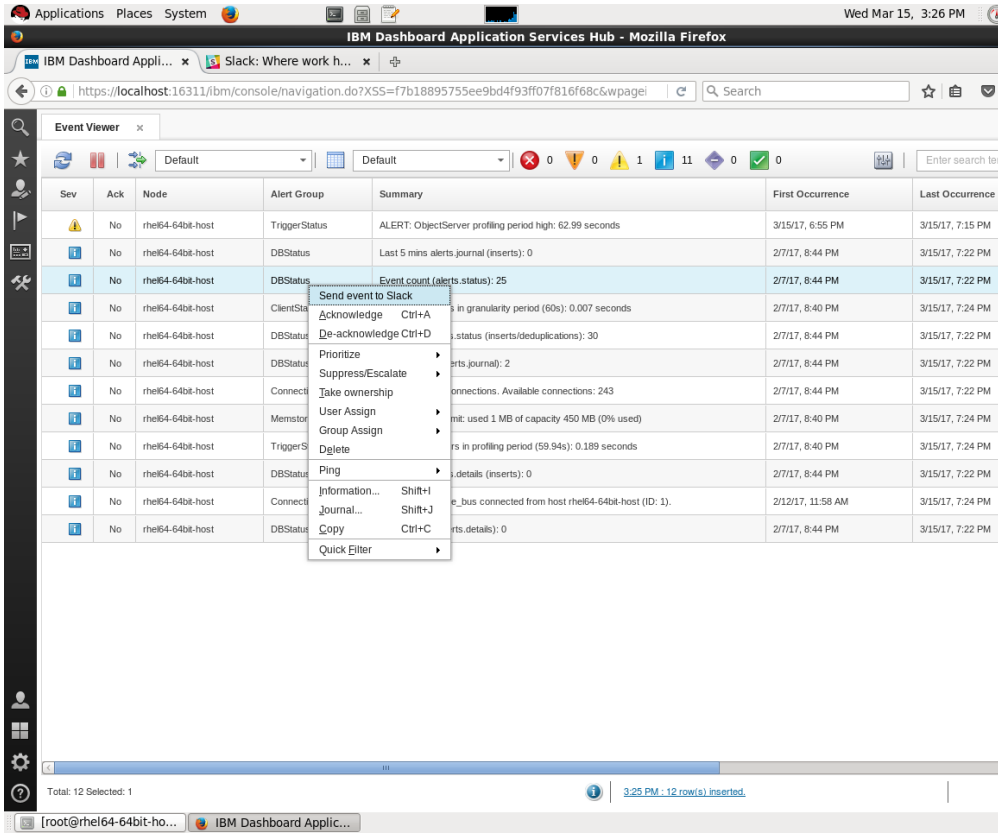
14. Rename the tool in the menu to "Send event to Slack"



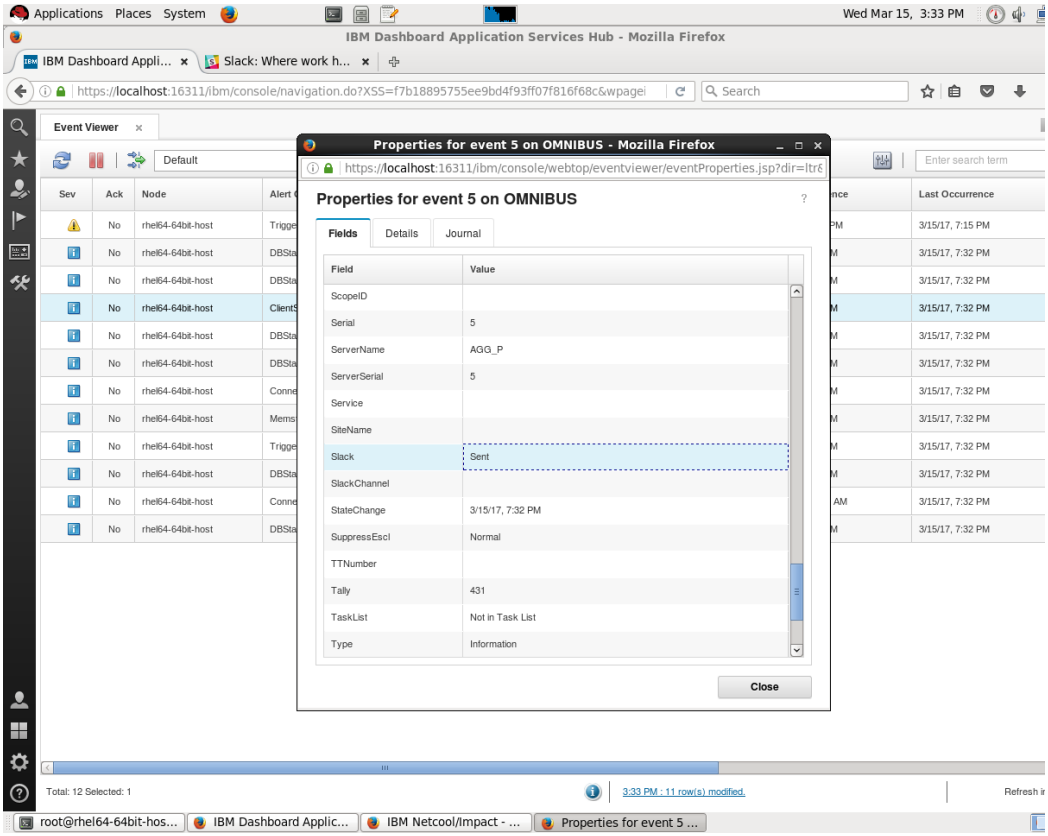
15. Open the event viewer tab



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.



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"
;



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";


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*";          }
elseif
```

```
   (@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_stop -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
```

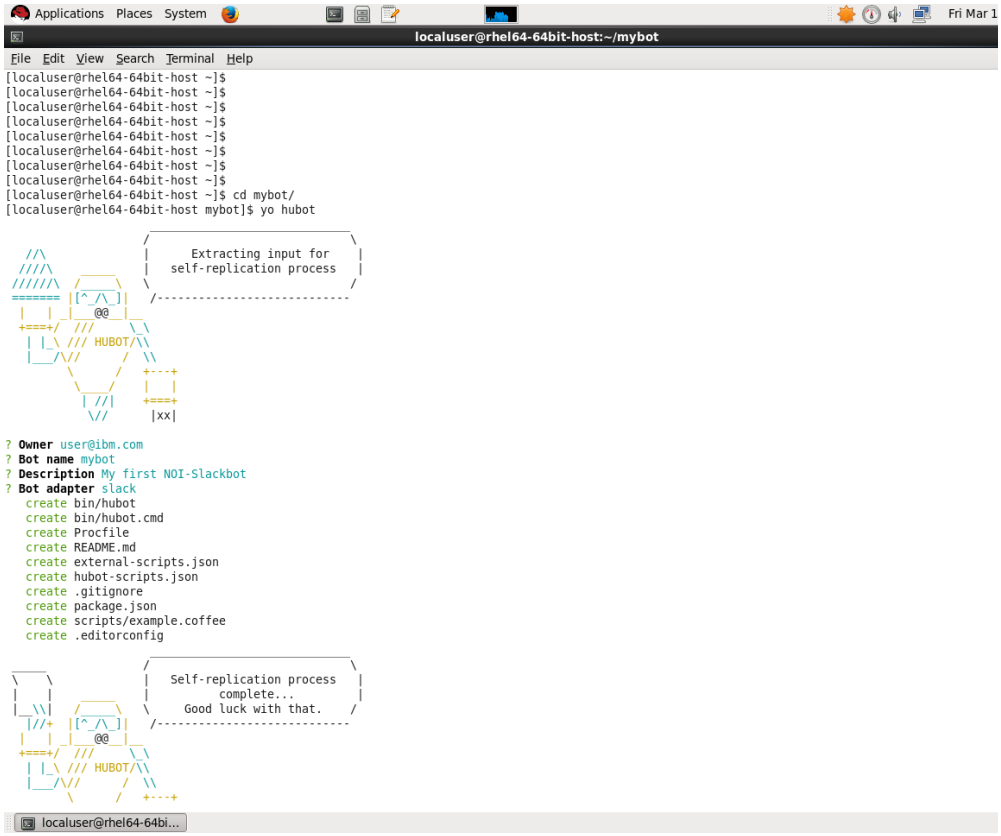
```
cd ~
```

```
mkdir bot
```

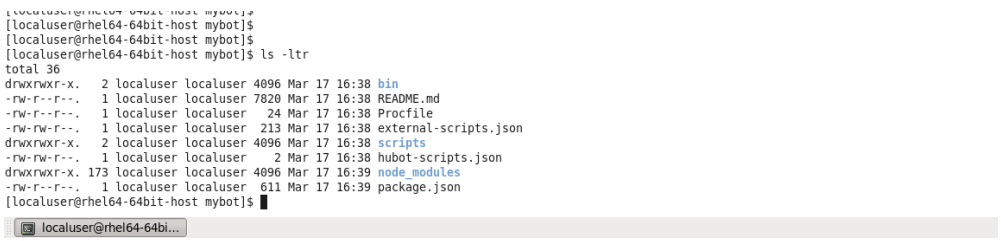
```
cd bot
```

```
yo hubot
```

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



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



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