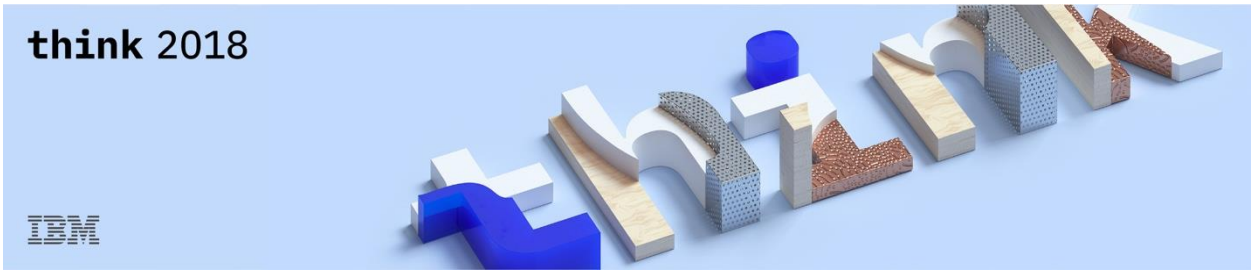


Lab Center – Hands-on Lab

Session 9102

Session Title Cloud Service Management and ChatOps - a new way to solve problems

Robert Barron, IBM, brobert@il.ibm.com

Table of Contents

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for

informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

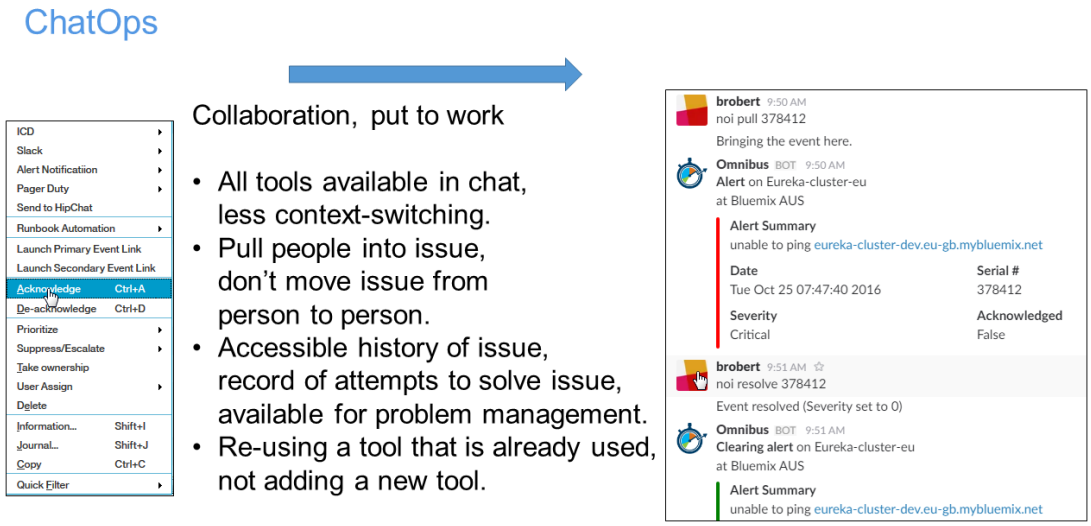
IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current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

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

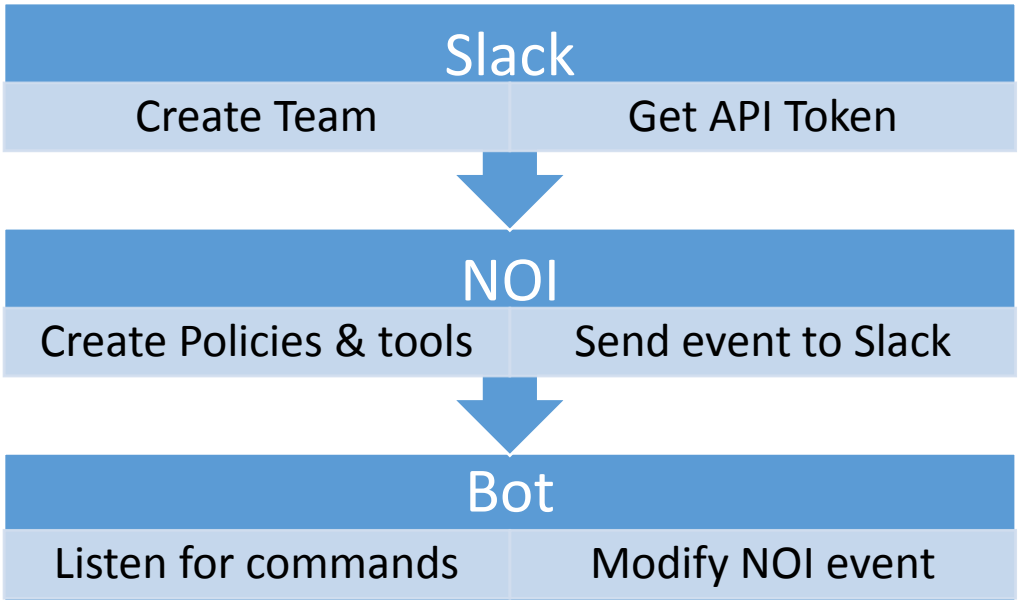
Objectives

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 by creating a Minimal Viable Solution which will showcase capabilities.
For further information and to share knowledge, please reach out to the author of this lab using the email brobert@il.ibm.com



Initial setup

Note that the scripts and pieces of code detailed in the following instructions are available in the github repository at https://github.com/ibm-cloud-architecture/2018-Think-9102 or https://ibm.biz/chatops9102.

You may clone or copy/paste from there instead of this document.

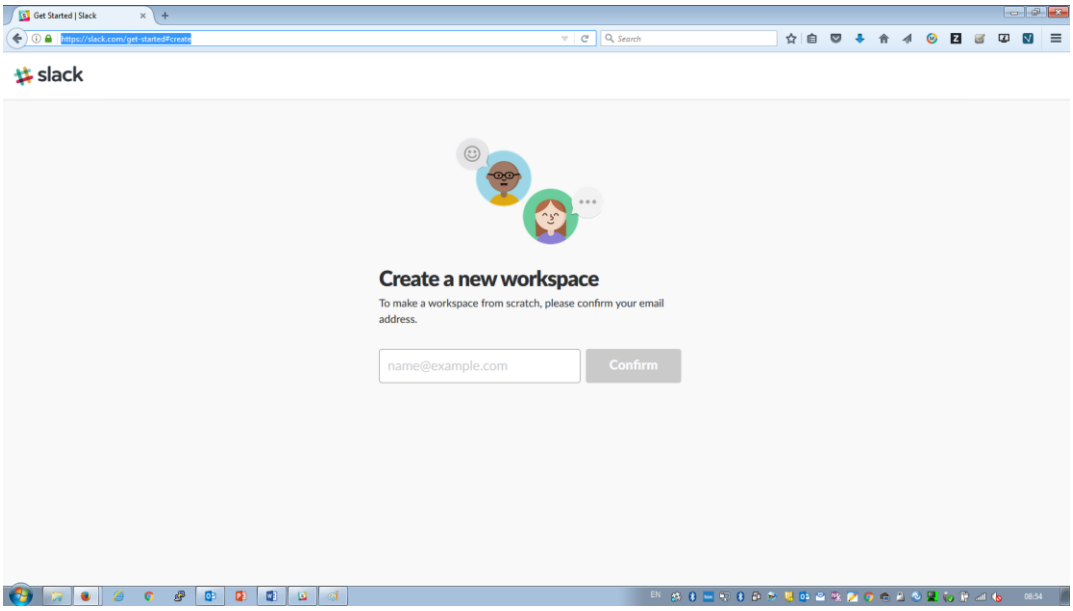
Start the application stack by running the script startup.sh which is on the desktop.

Create the Slack Workspace

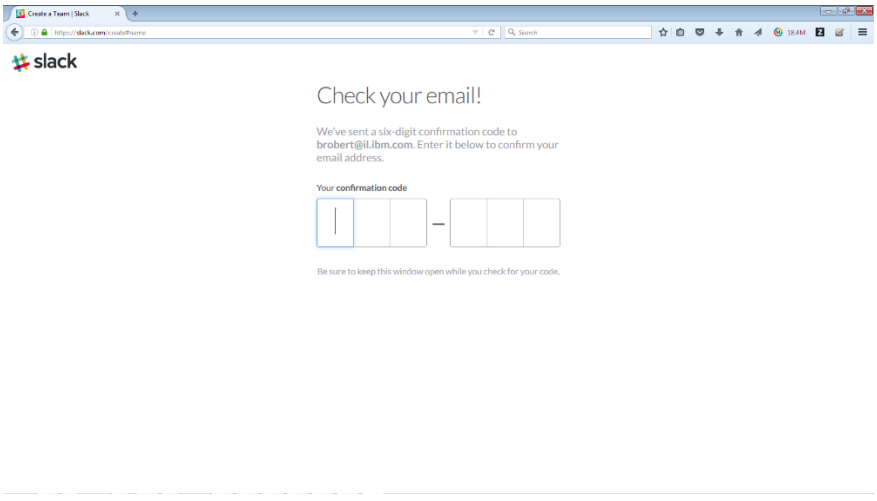
In this section, we will create the Slack workspace and create a token for API access that will be used throughout the lab.

If you have administrative access to an existing Slack workspace, you may skip to step 11 by logging into your Slack.

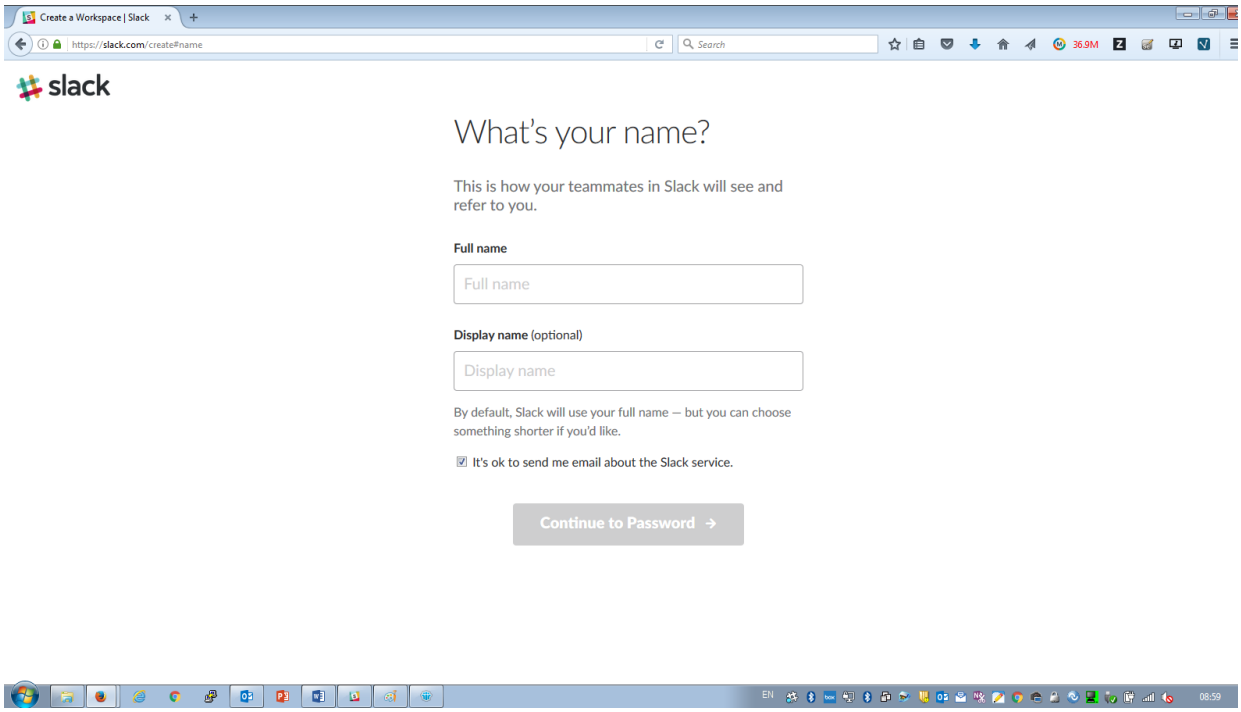
1. Go to the Slack main page https://slack.com/get-started#create and create a new workspace



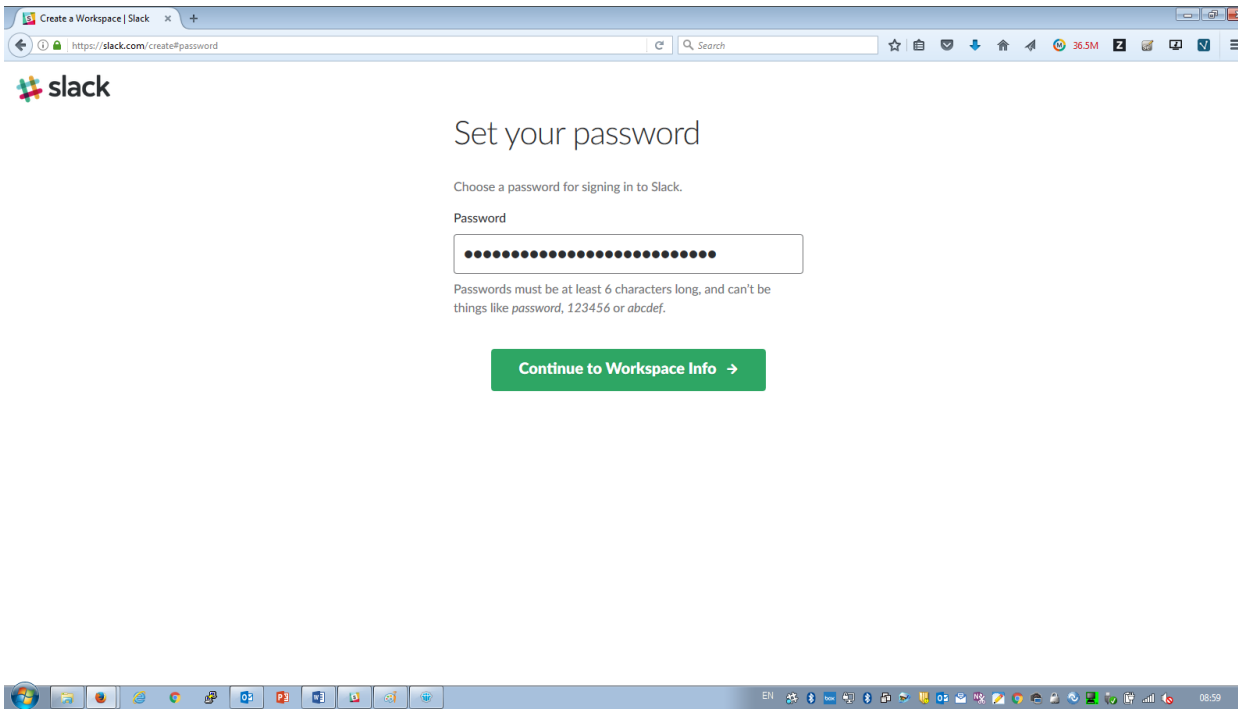
2. Check your email and confirm with Slack



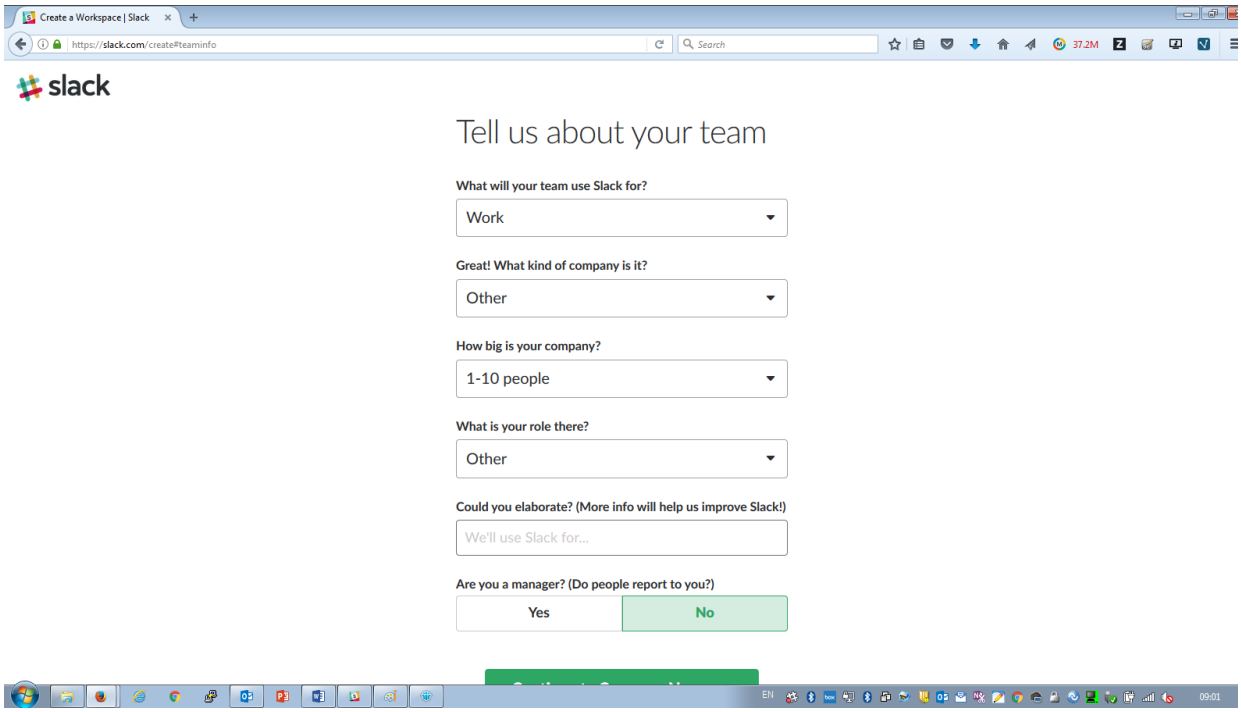
3. Enter your name



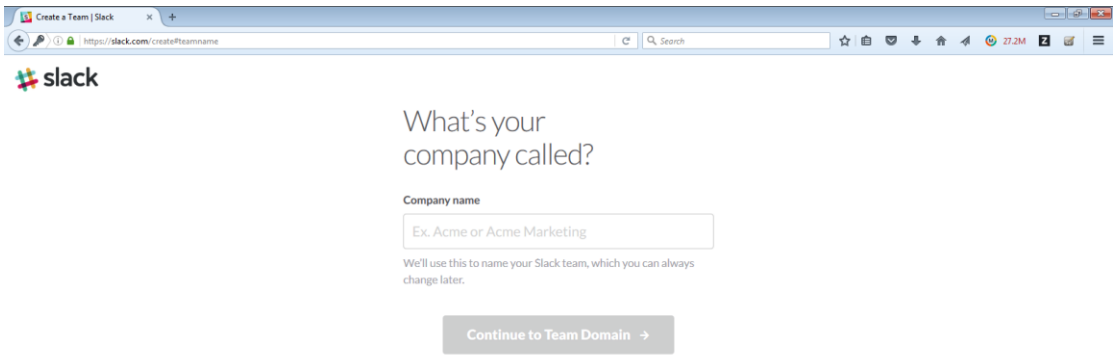
4. Set your password



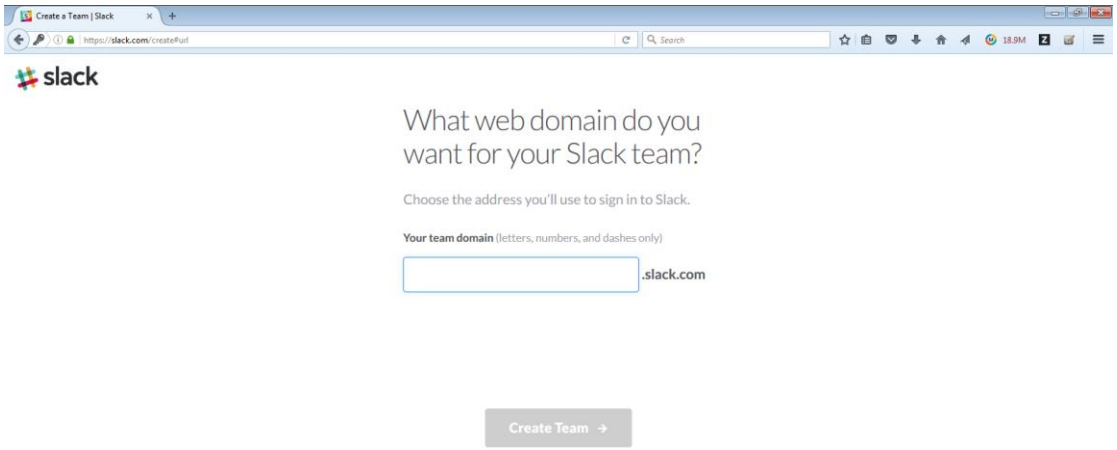
5. Fill in basic details about the workspace



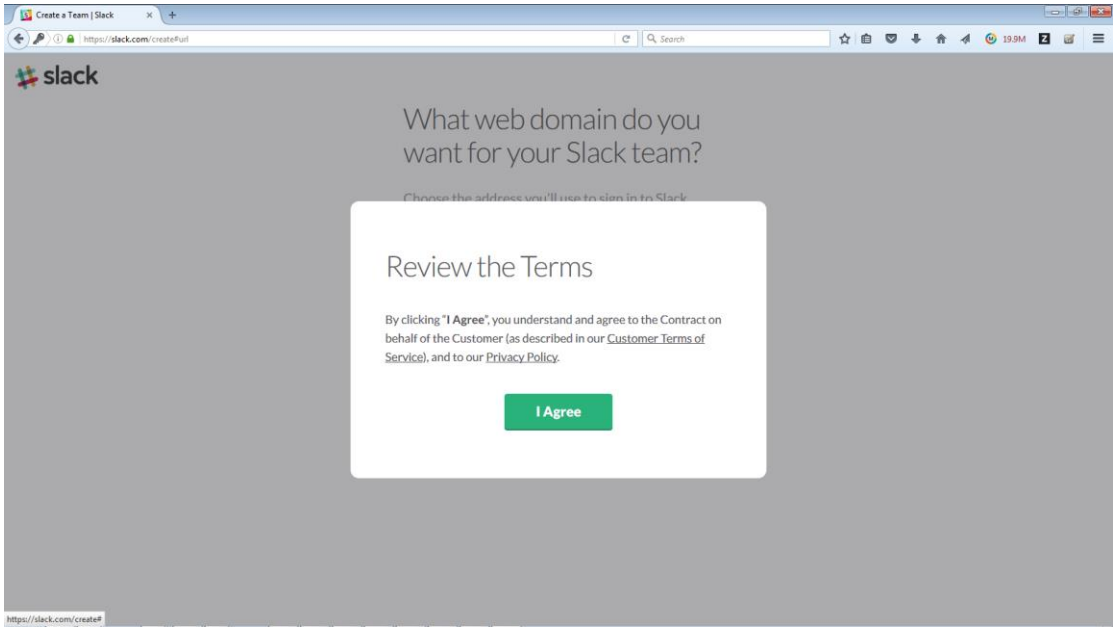
6. Fill in company name



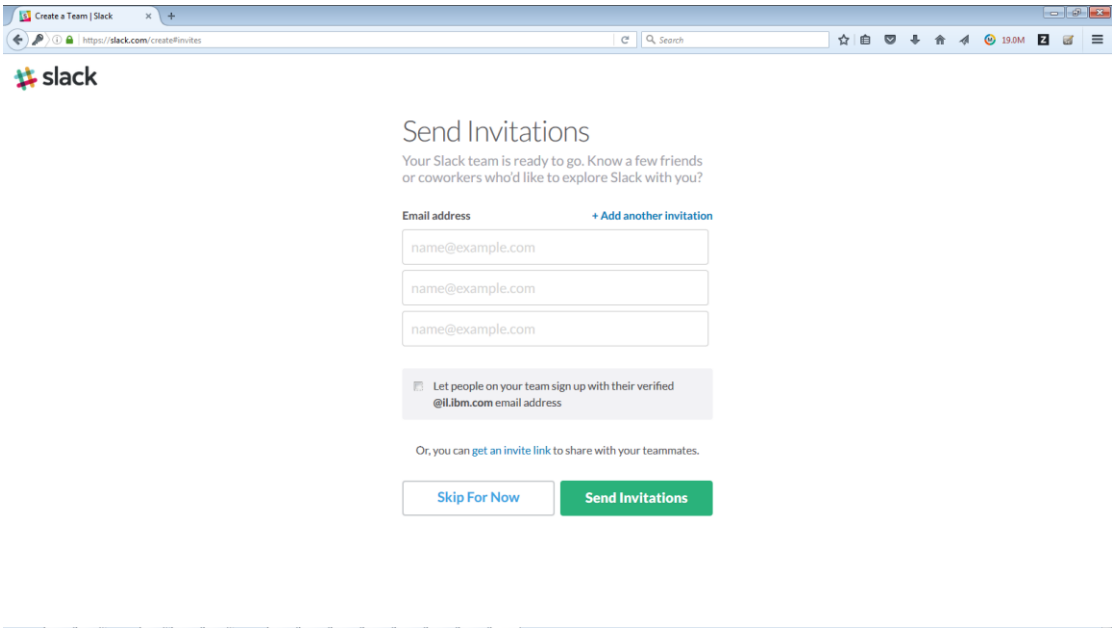
7. Choose your Slack's web domain



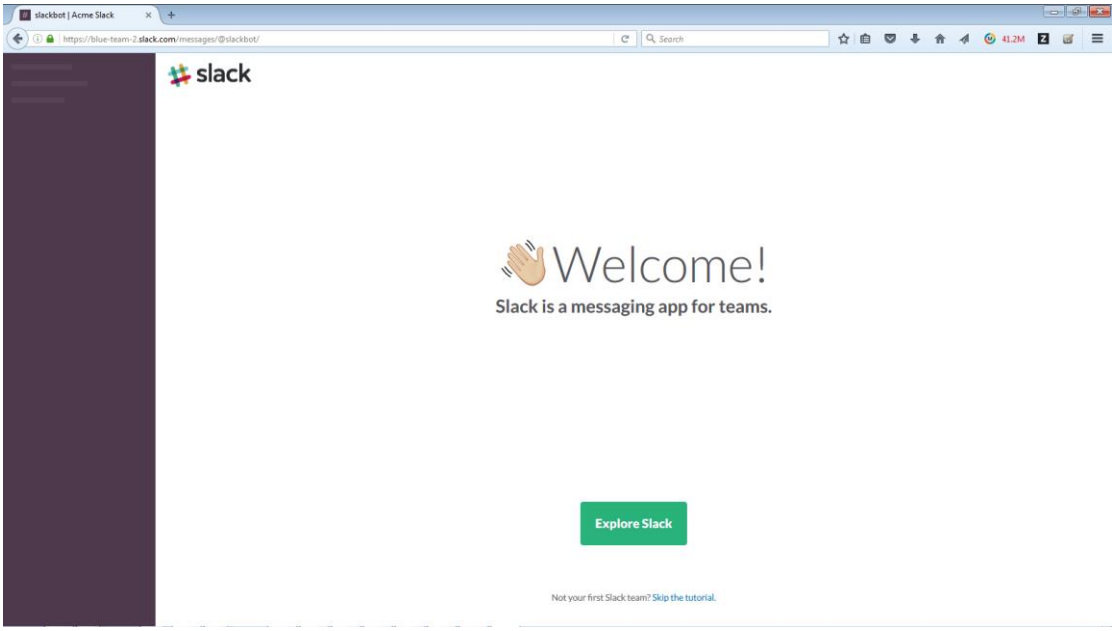
8. Agree to the terms (you have no choice, really!)



9. You may skip this step

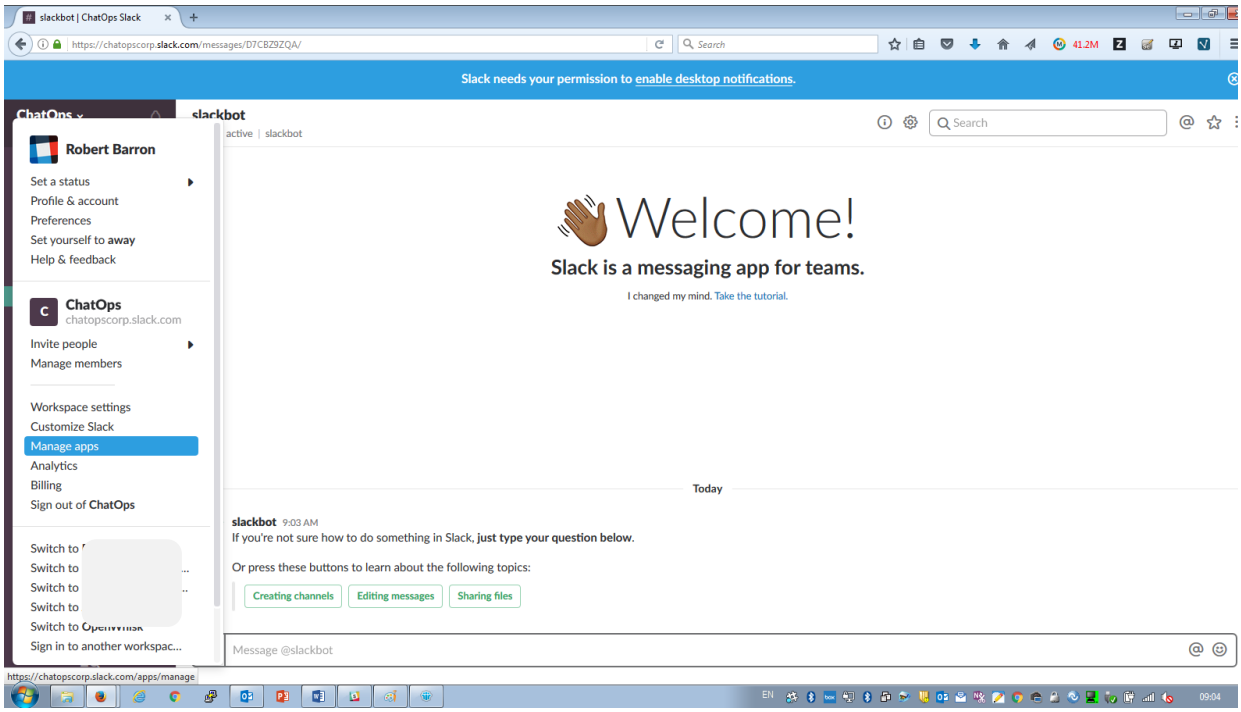


10. You have logged into your new Slack

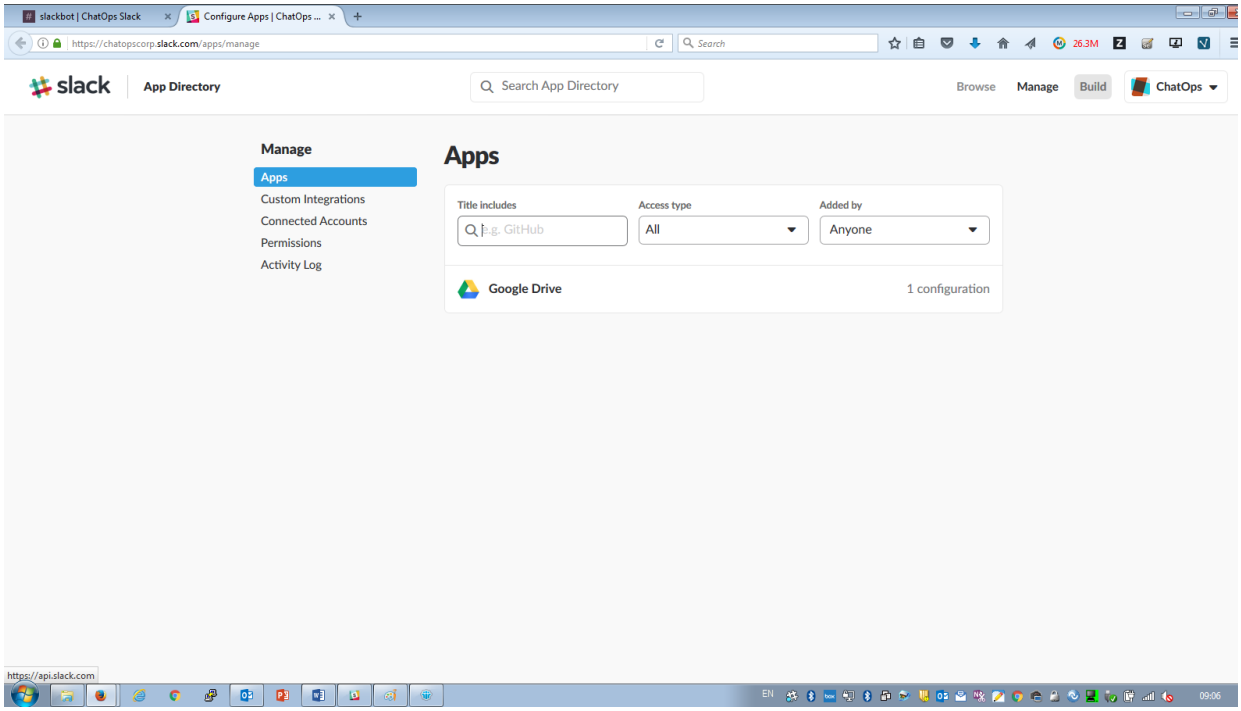


Depending on your familiarity with Slack, you may take or skip the tutorial

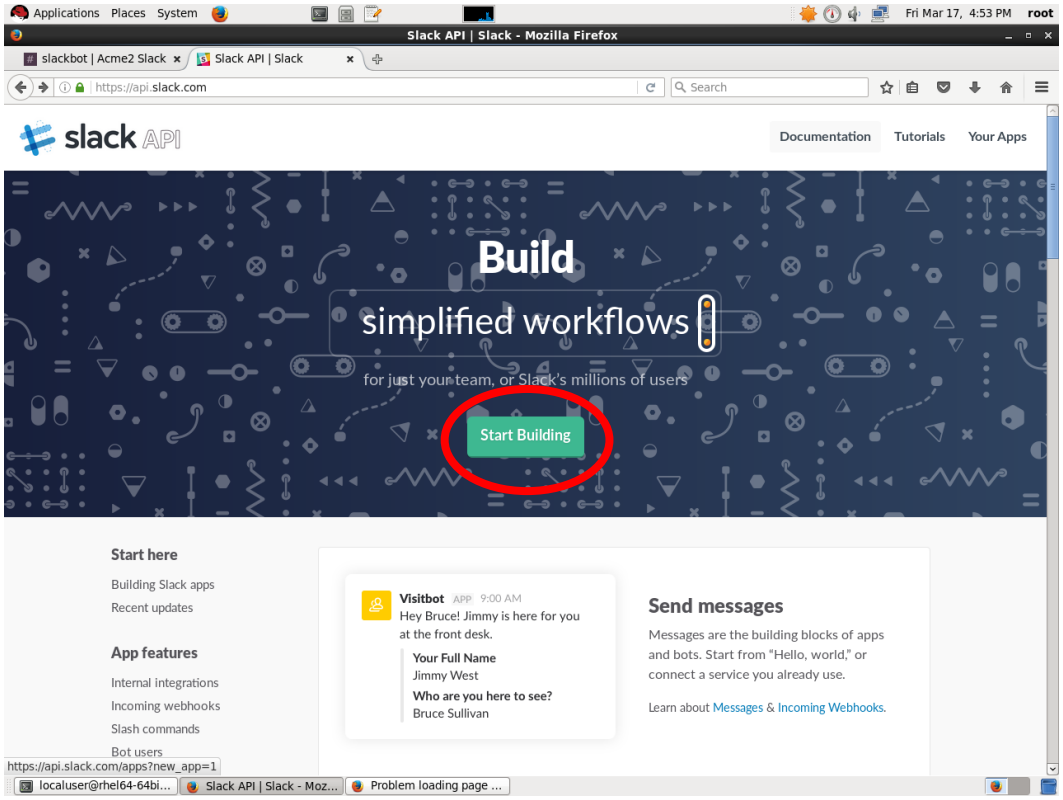
11. Open the menu to "Manage apps"



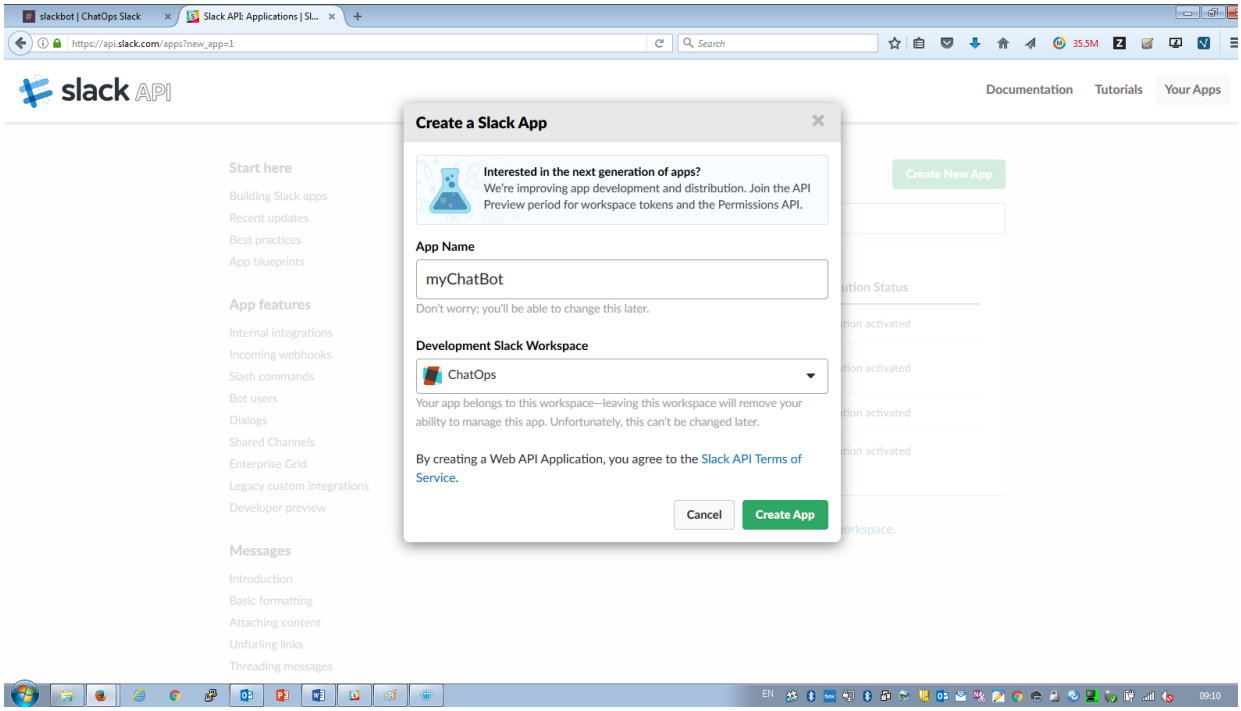
12. In the opened screen, choose the menu option Build



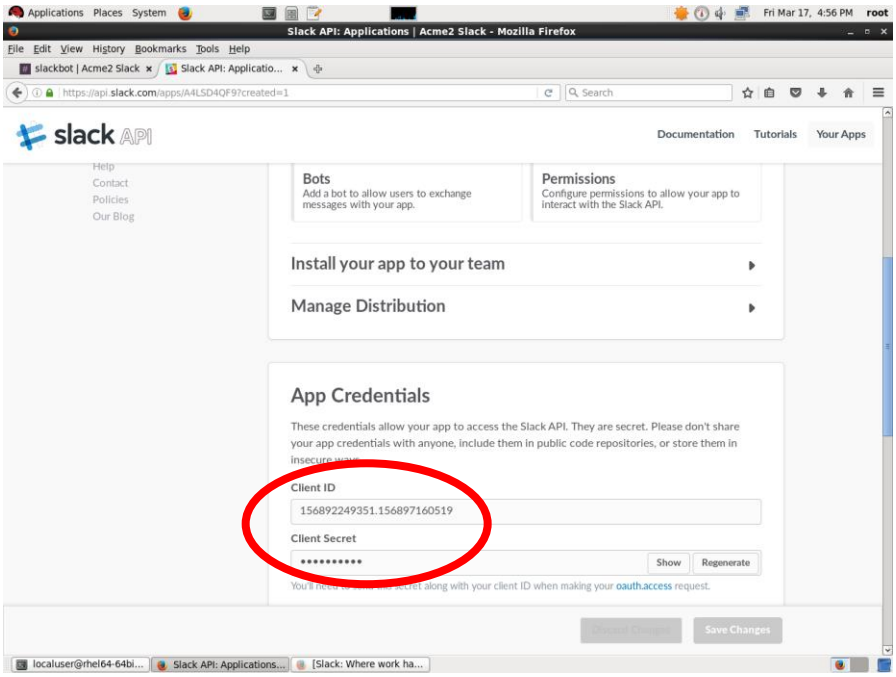
13. Click "Start Building"



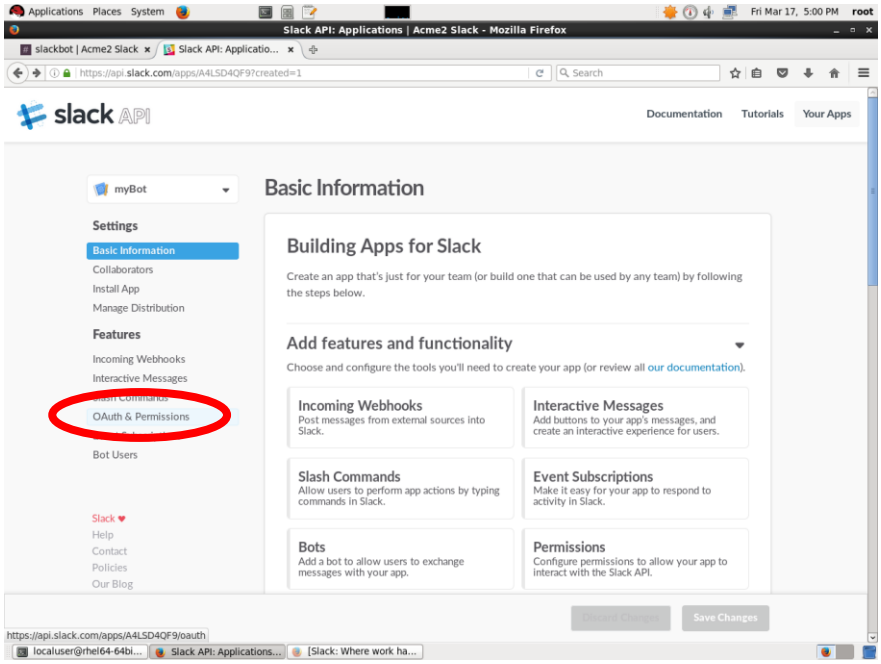
14. Choose a name for your app (myChatBot, for example)



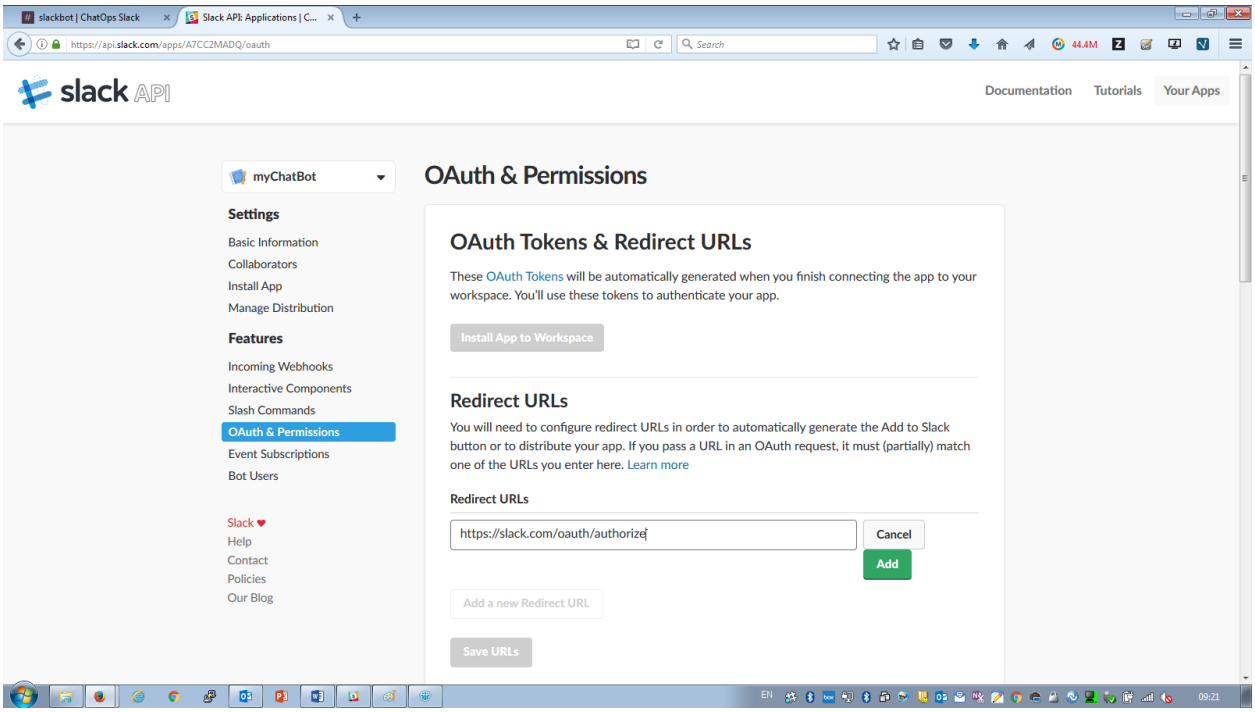
15. Congratulations – you have an app!
    Scroll down and record the Client ID and Client Secret (copy/paste to a file)



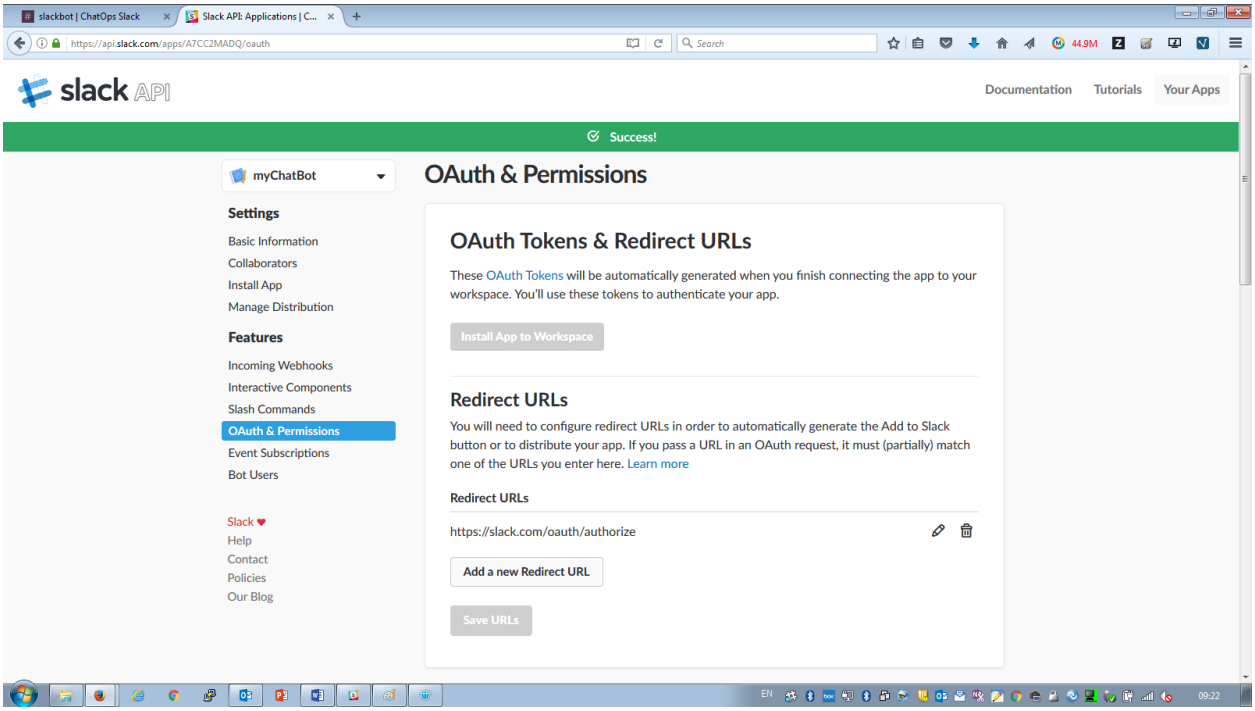
16. Choose OAuth & Permissions from the menu



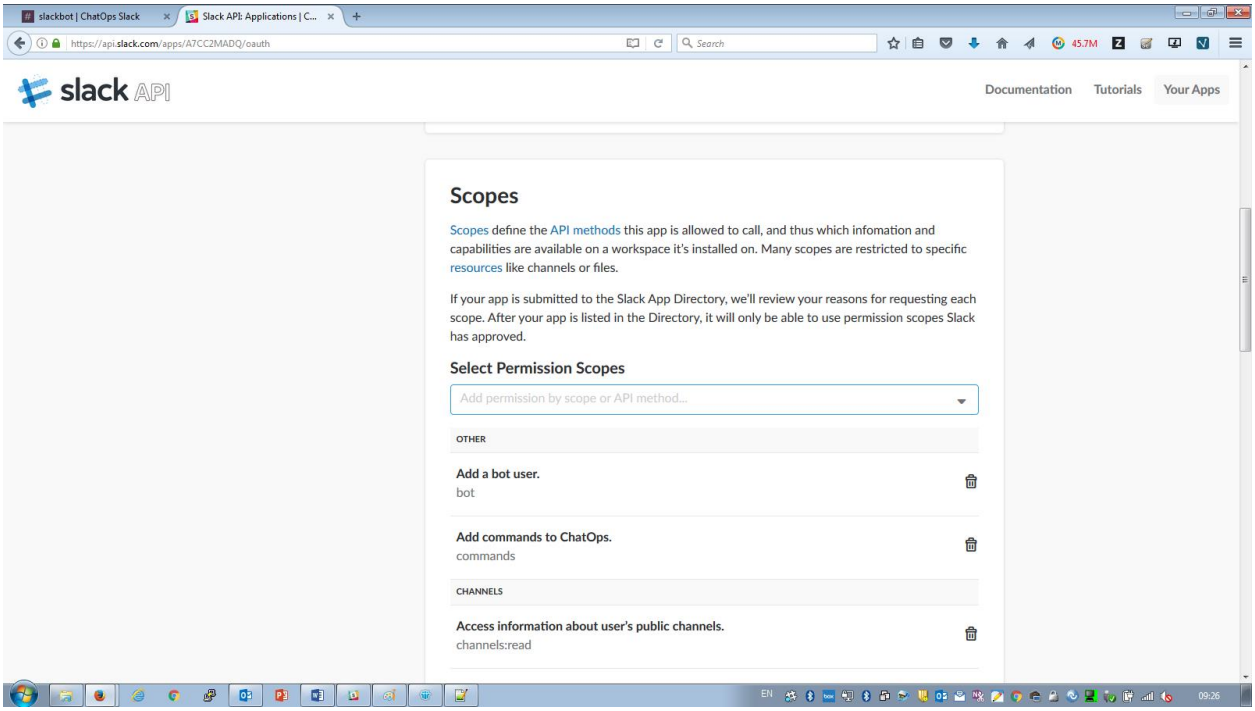
17. Add a new Redirect URL with the target https://slack.com/oauth/authorize



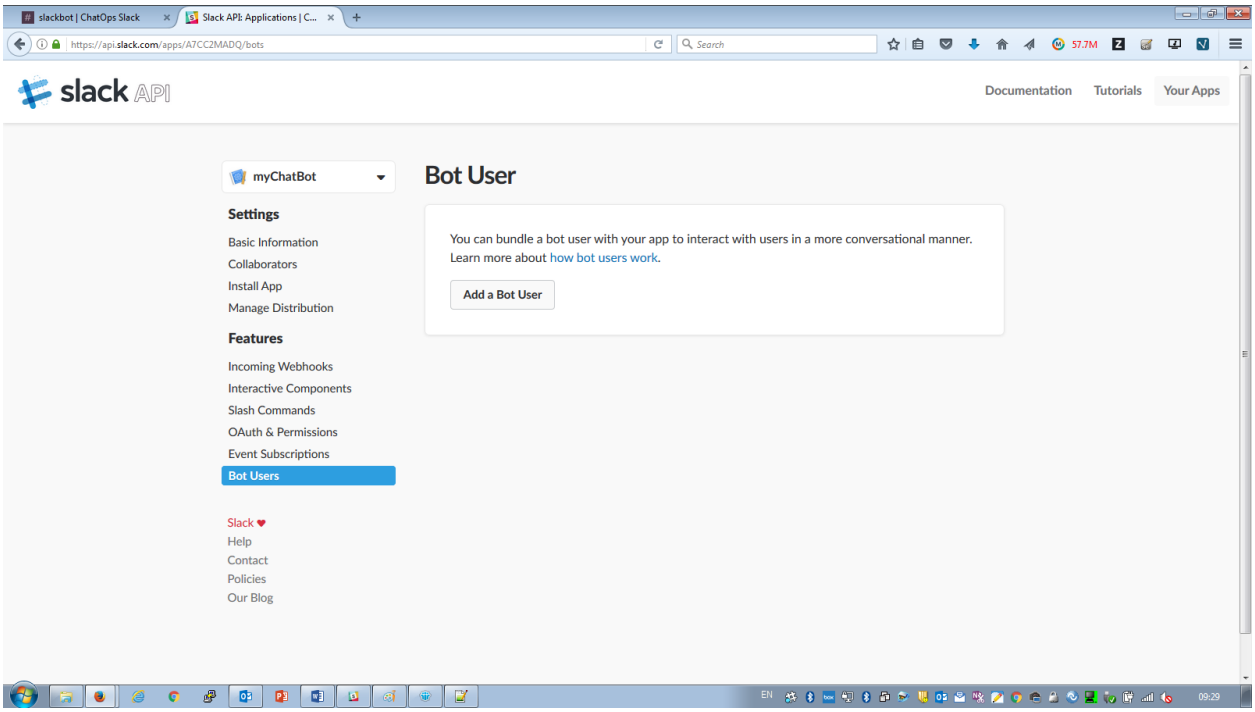
18. Click "Save URLs" to save your change

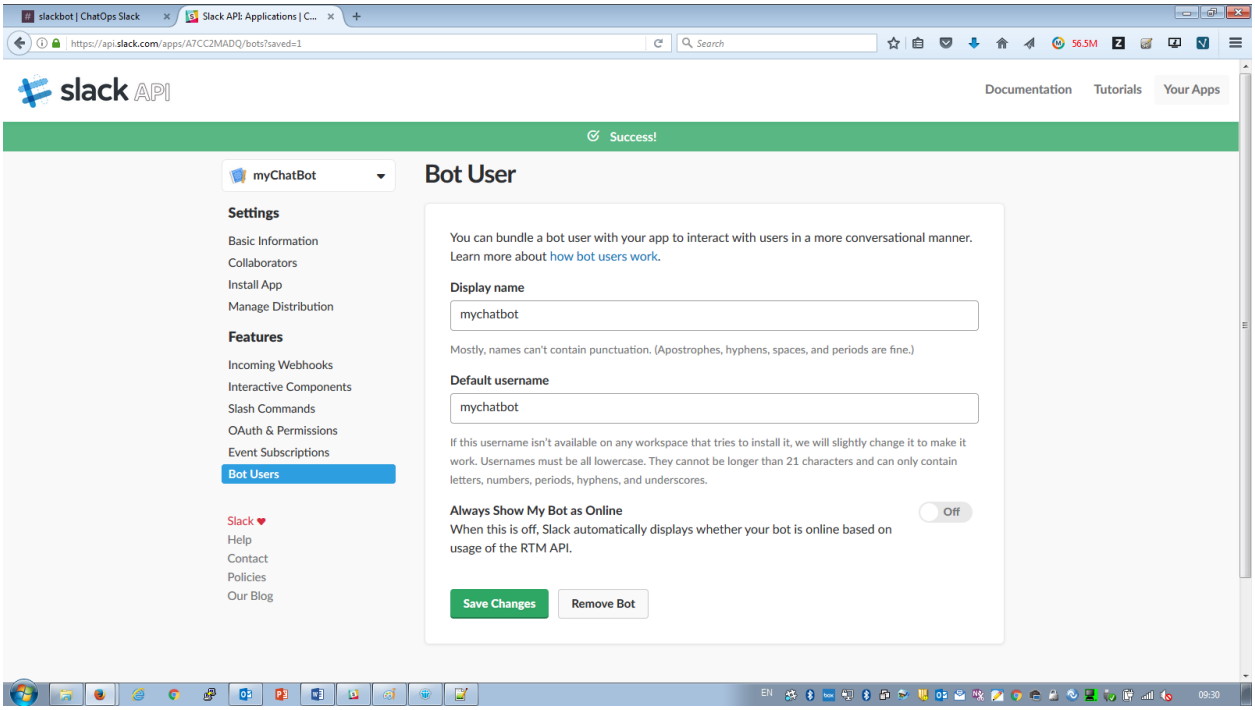


19. Scroll down and add the following Permission Scopes:
    commands, bot, chat:write:user, channels:read, channels:write, groups:read, groups:write, files:write:user, users:read

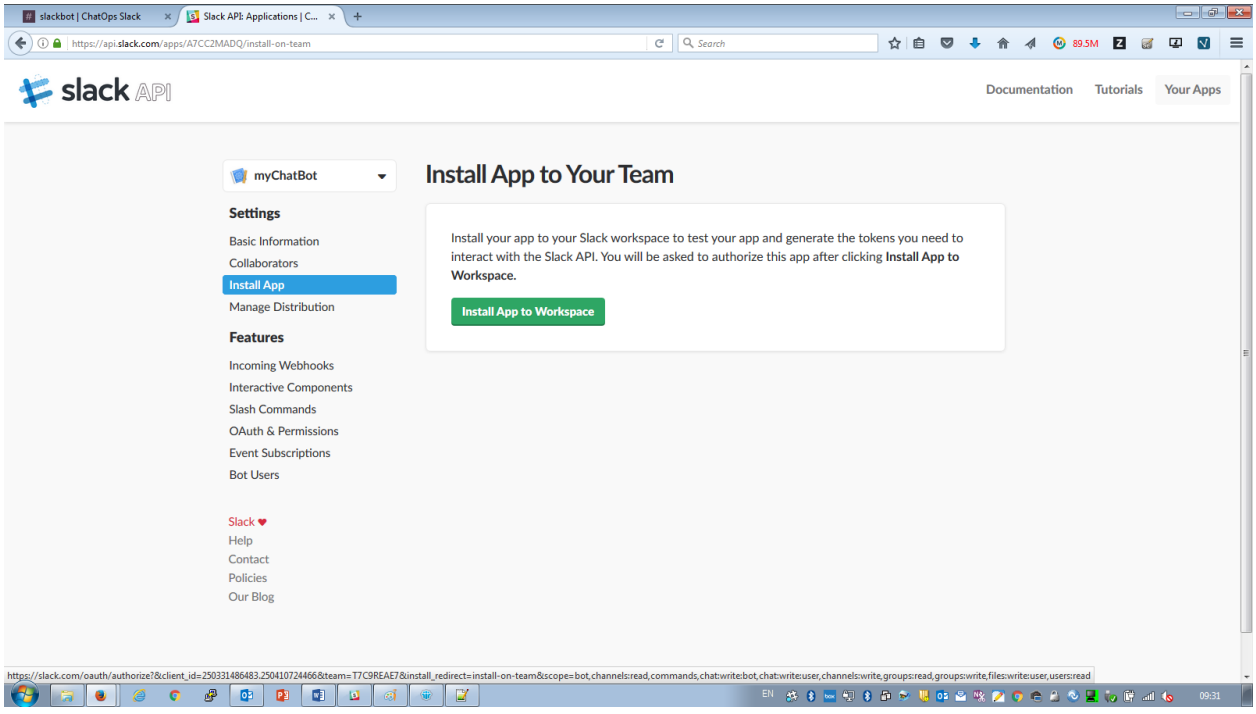


20. Go to the "Bot Users" menu and click "Add a Bot User"

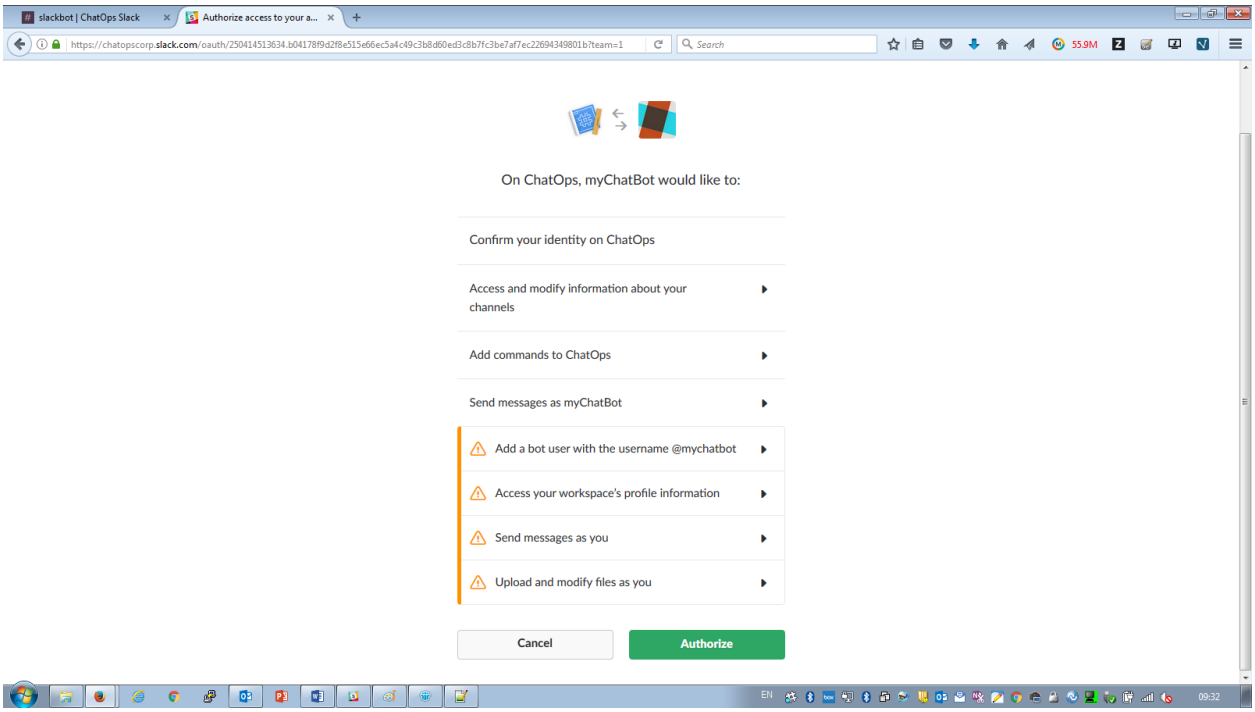




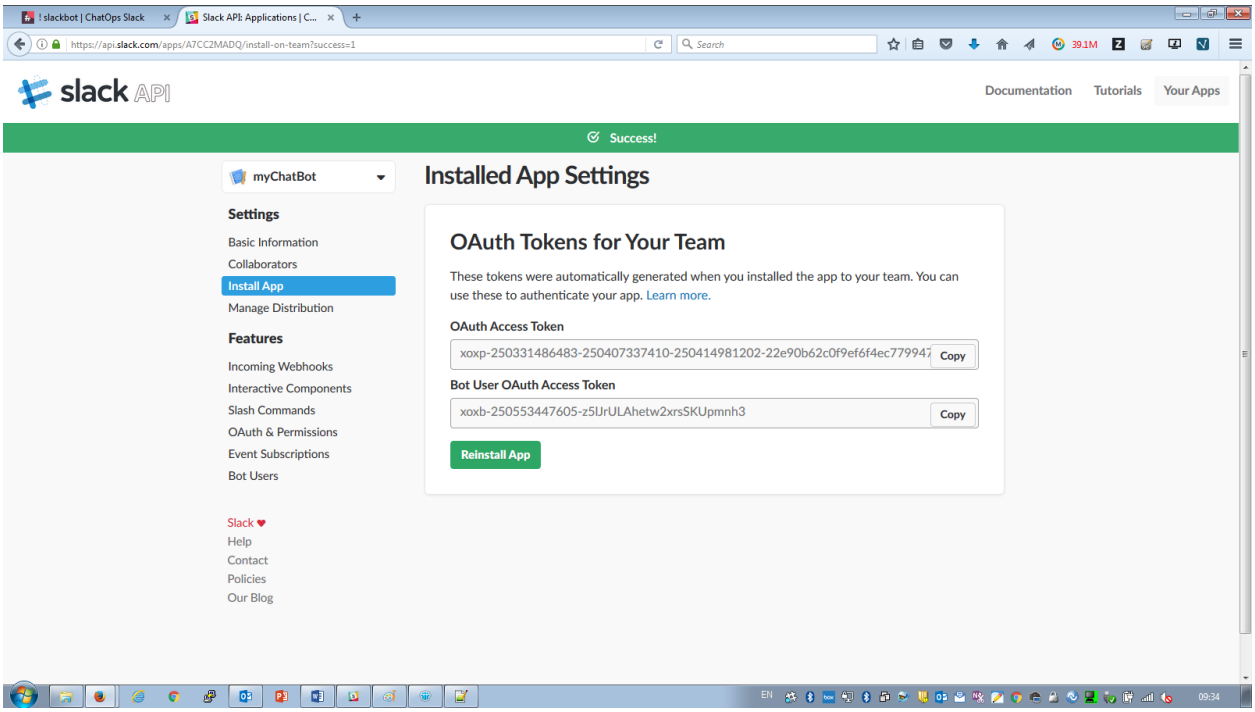
21. Go to the "Install App" menu and click "Add App to Workspace"



22. Authorize the permissions

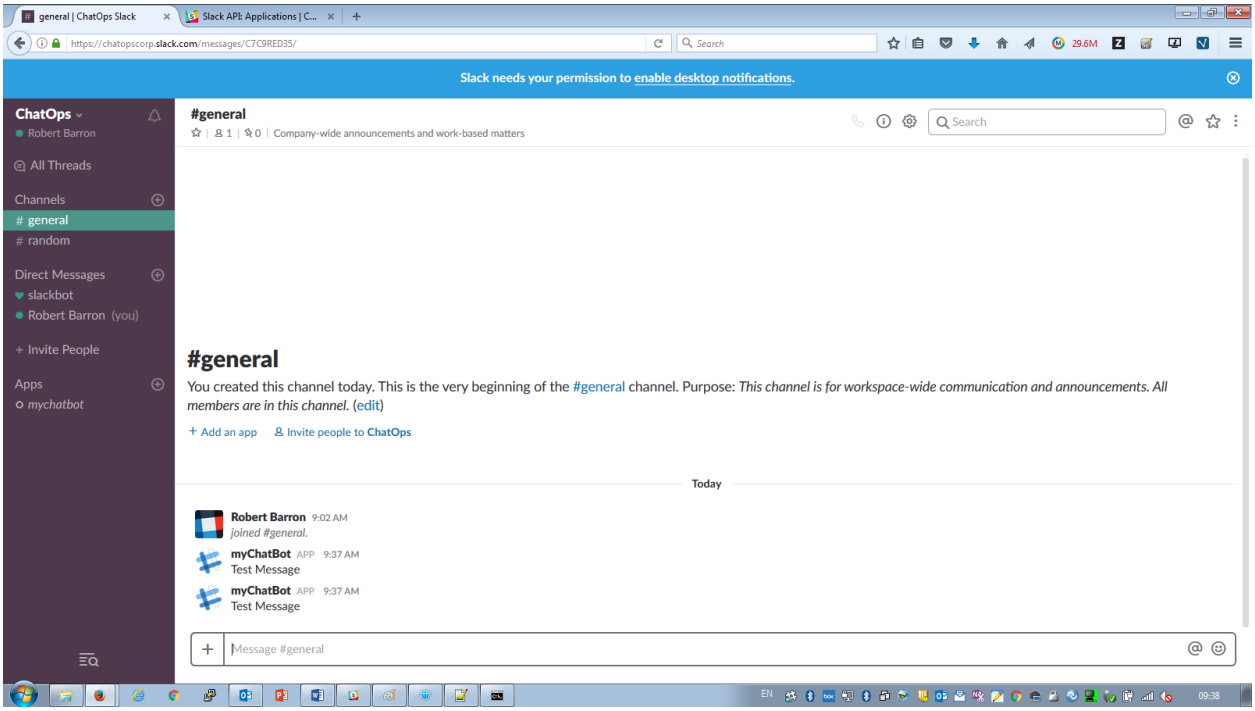


23. Record the OAuth tokens



24. Test your tokens by opening a terminal window and running the following command

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



Sending a test sample message from Impact

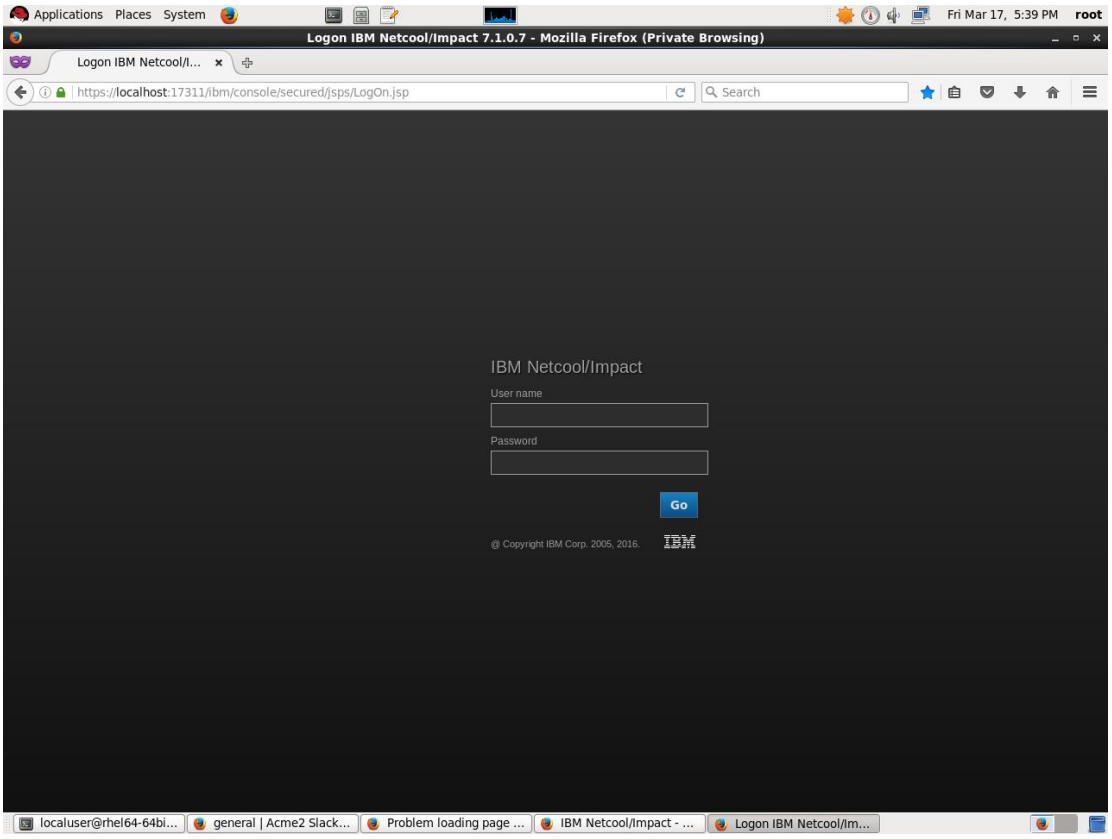
We will use Impact to send the initial event to Slack. Open a terminal window to run the following commands.

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-NOI-ChatOps -file /tmp/slack.cert -keystore /opt/IBM/tivoli/impact/wlp/usr/servers/NCI/resources/security/trust.jks -storepass passw0rd
   ```

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

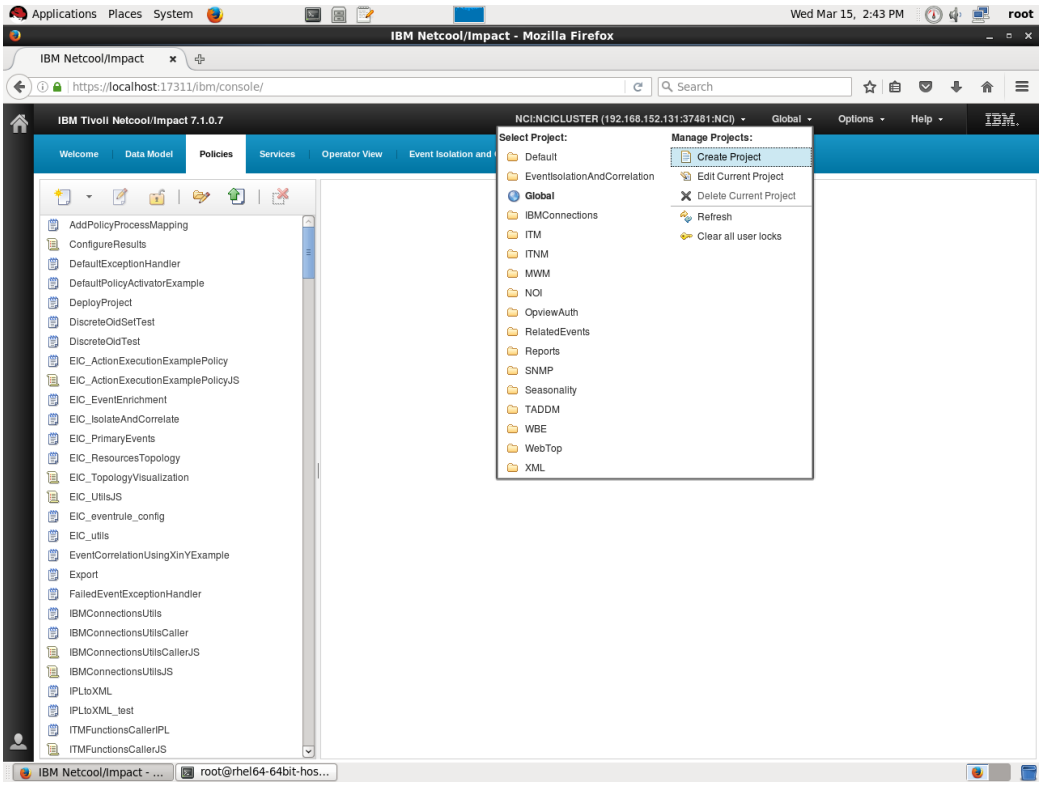
3. Restart Impact using the commands
   ```
   /opt/IBM/tivoli/impact/bin/stopImpactServer.sh
   /opt/IBM/tivoli/impact/bin/startImpactServer.sh
   ```

4. Open a new Firefox tab and login to Impact.
   http://localhost:17310/ibm/console (or the bookmark Logon IBM Netcool/Impact)

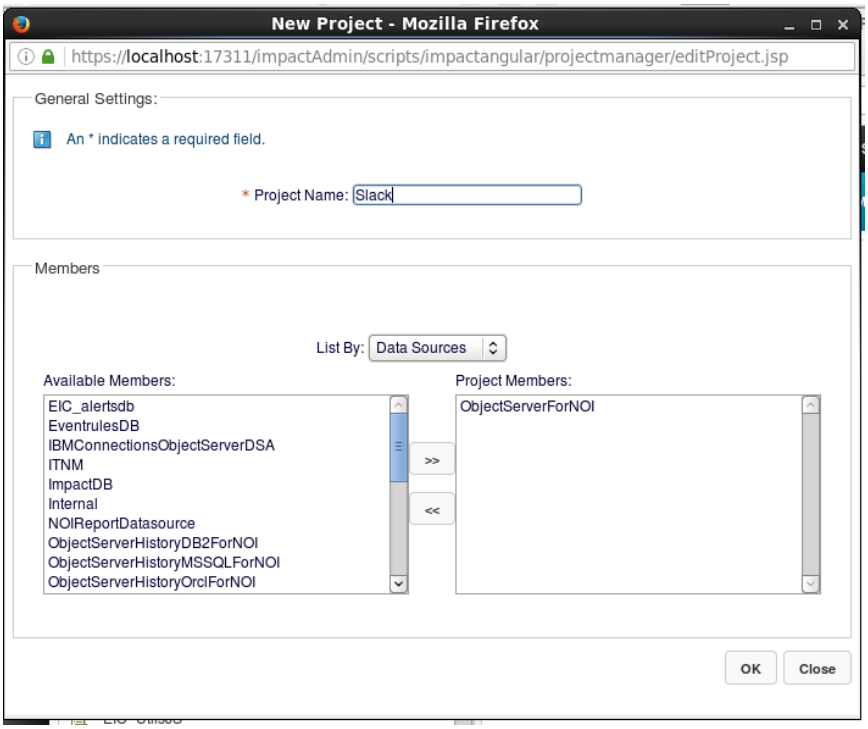


User/password is impactadmin/passw0rd

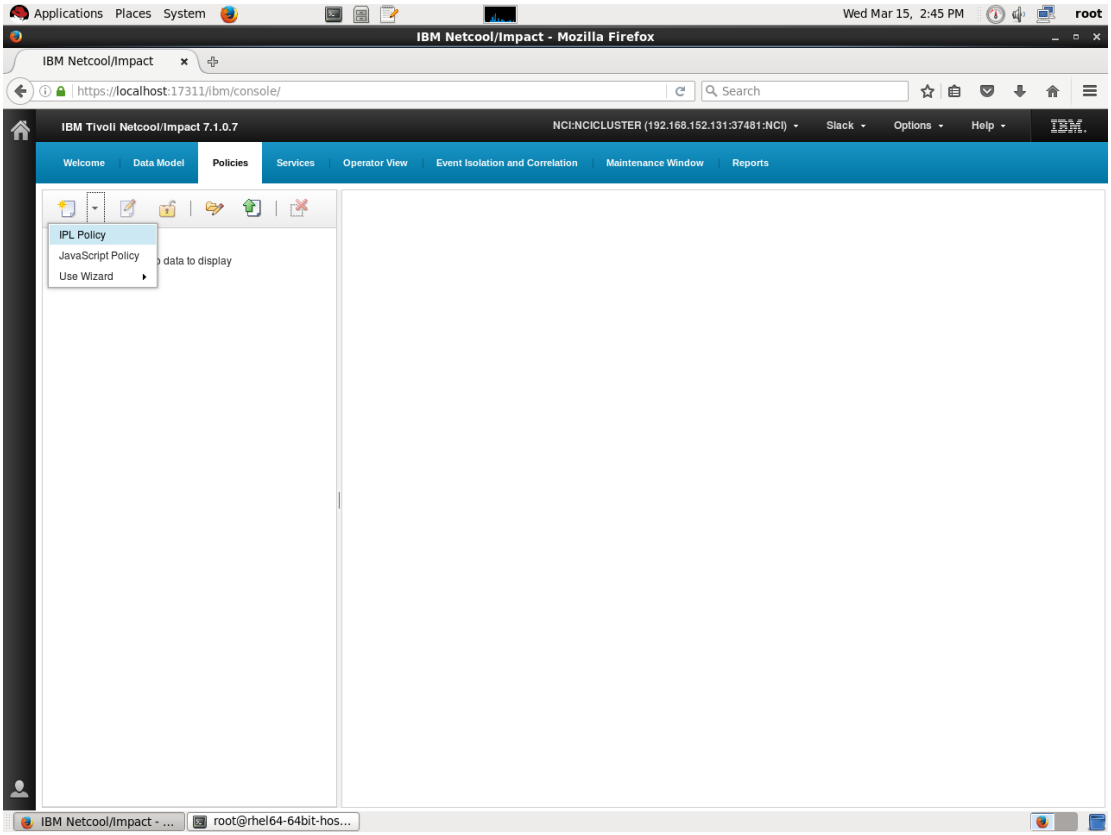
5. Click "Global" and then "Create Project"



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



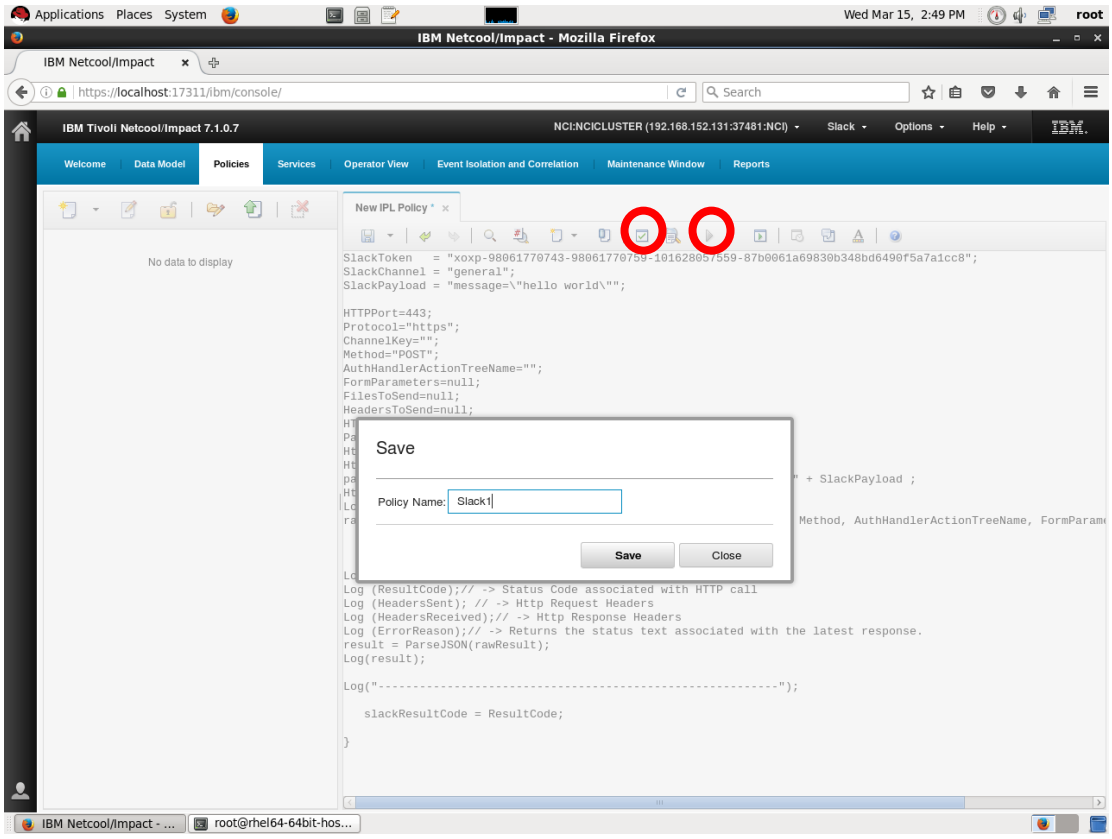
7. Create a new IPL policy


Applications Places System
Wed Mar 15, 2:45 PM
root
IBM Netcool/Impact - Mozilla Firefox
IBM Netcool/Impact
https://localhost:17311/ibm/console/
Search
IBM Tivoli Netcool/Impact 7.1.0.7
NCI:NCICLUSTER (192.168.152.131:37481:NCI)
Slack
Options
Help
IBM
Welcome
Data Model
Policies
Services
Operator View
Event Isolation and Correlation
Maintenance Window
Reports
IPL Policy
JavaScript Policy
Use Wizard
data to display
IBM Netcool/Impact - ...
root@rhel64-64bit-hos...

Enter the following code 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. Save the policy, 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/NCI_policylogger_<PolicyName>.log

Sending a real Omnibus event from Impact

1. Create a file called updateOmnibusSchema.sql 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. Run the following command to execute the sql on the ObjectServer:

/opt/IBM/tivoli/netcool/omnibus/bin/nco_sql -server AGG_P -password passw0rd -user root -input updateOmnibusSchema.sql

(0 rows affected)

(1 row affected)

(1 row affected)

(1 row affected)

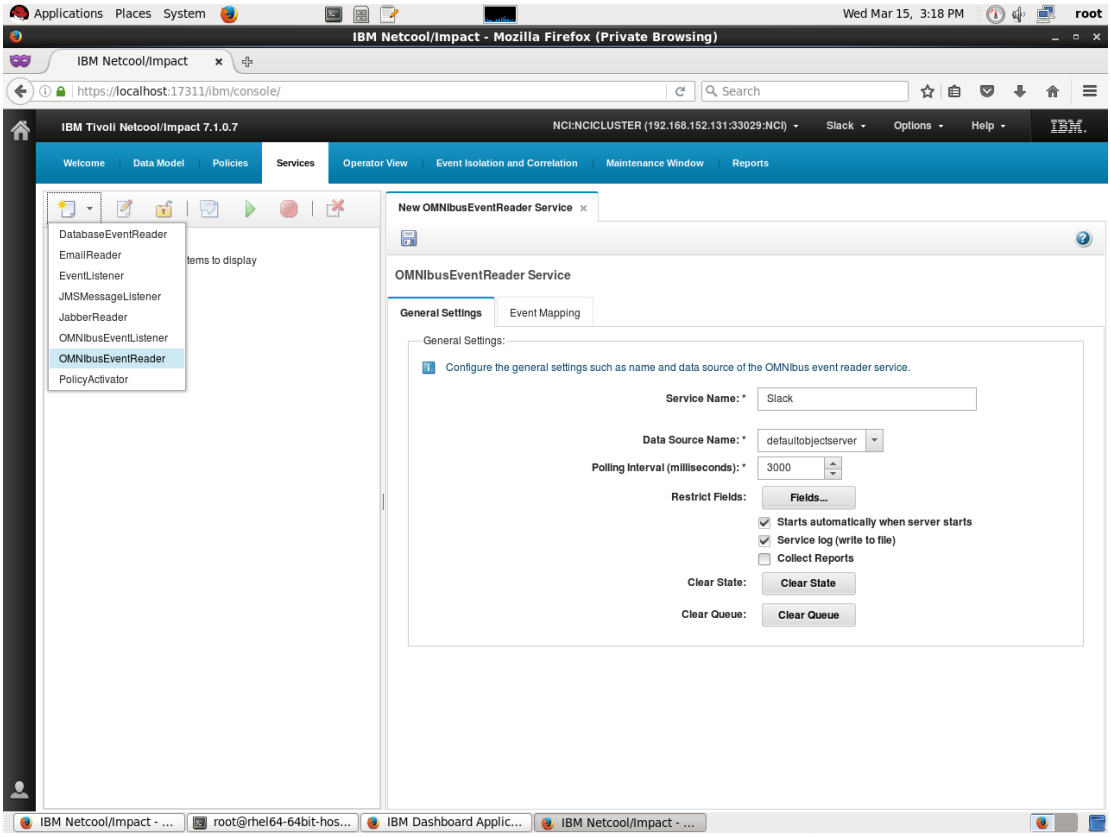
(0 rows affected)

3. Create a new Policy with the following code:

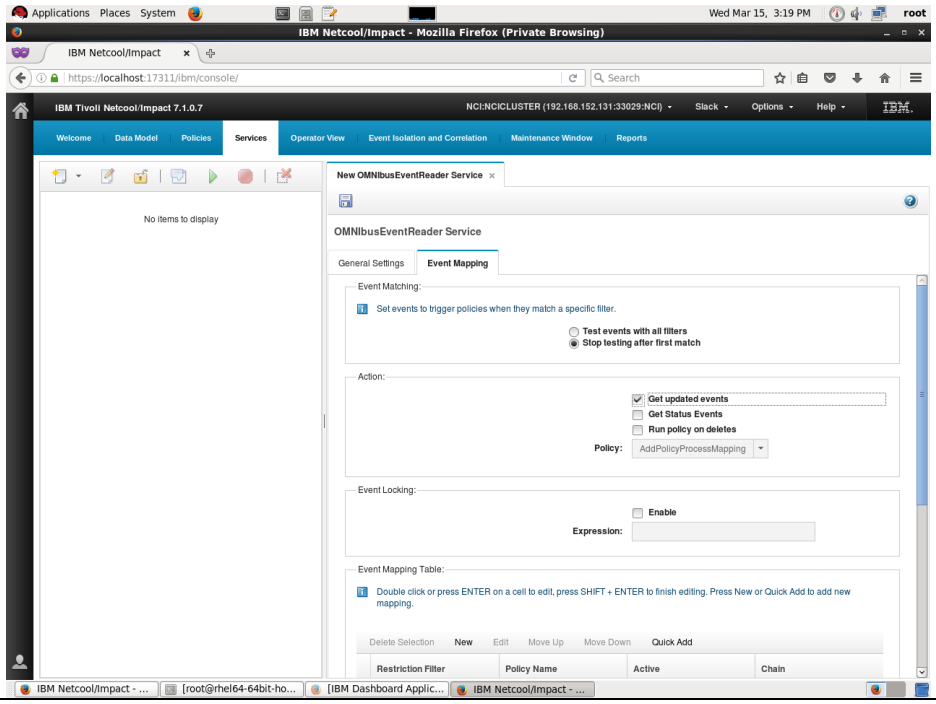
```
SlackToken   = "__INSERT_TOKEN_HERE__";
DASH_Server  = "https://localhost:16311/ibm/console";

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 ) { PreText = "*Clearing alert*"; } elseif
   (@Severity == 1 ) { PreText = "*Information*";    } elseif
   (@Severity == 2 ) { PreText = "*Alert*";          } elseif
   (@Severity == 3 ) { PreText = "*Alert*";          } elseif
   (@Severity == 4 ) { PreText = "*Alert*";          } elseif
   (@Severity == 5 ) { PreText = "*Alert*";          } else
                     { PreText = "*Information*";    }

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

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);
```

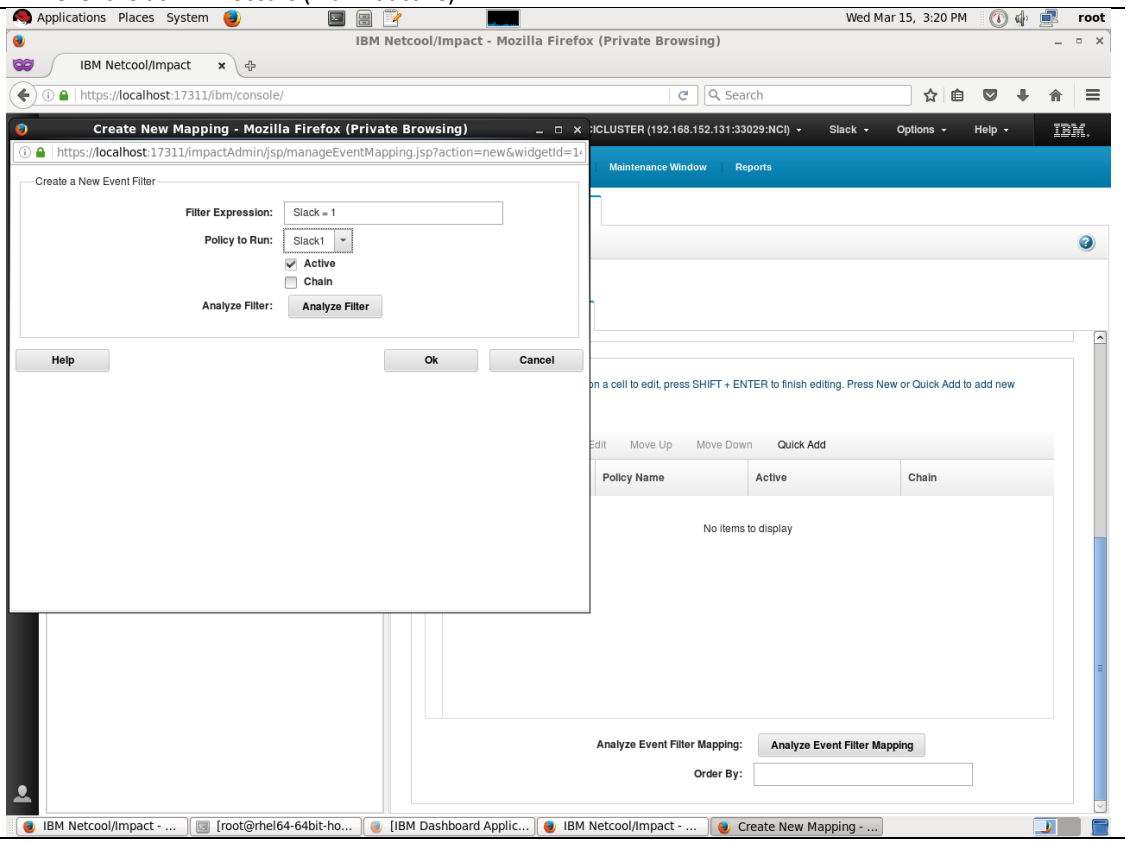
4. open the "Services" tab and create a new service of type OmnibusEventReader called "Slack"



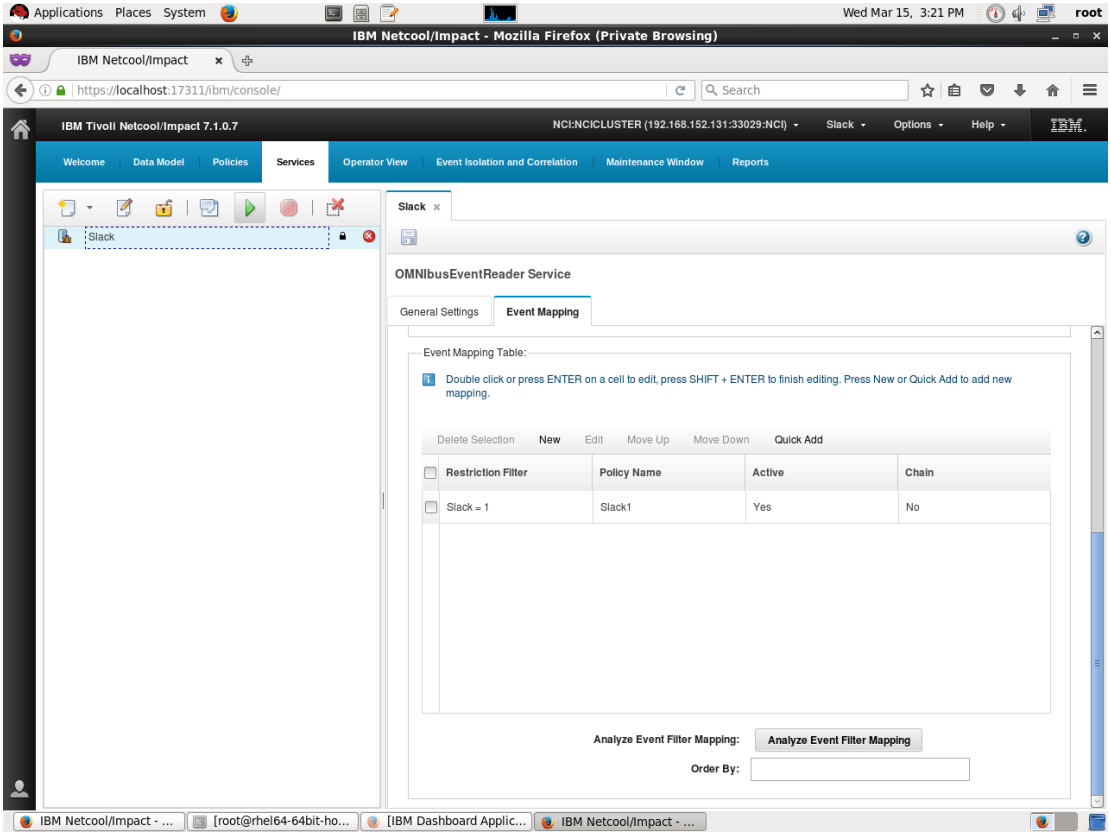
5. Move to the tab "Event Mapping" and check the option "Get Updated Events"



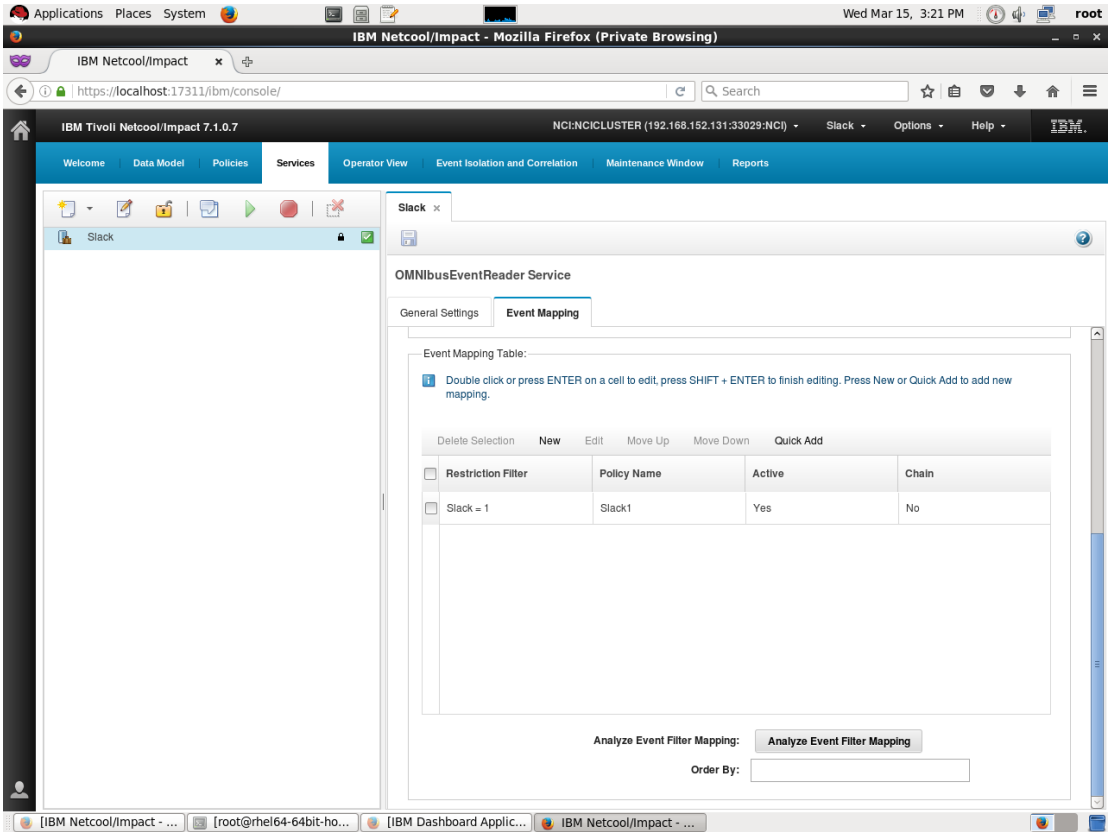
6. Add a new entry to the "Event Mapping Table" which will run the policy you have just created when the event "Slack = 1" occurs (mark it active)



7. Save the service

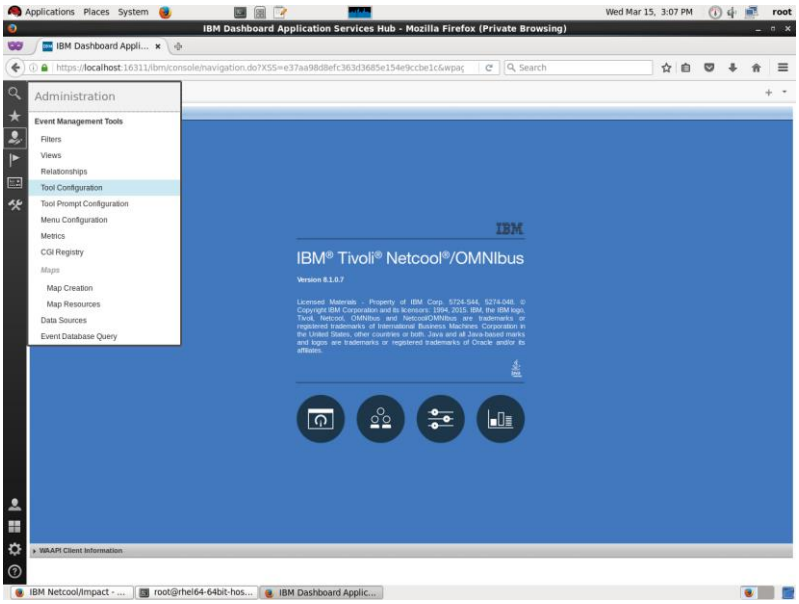


8. Start the Service

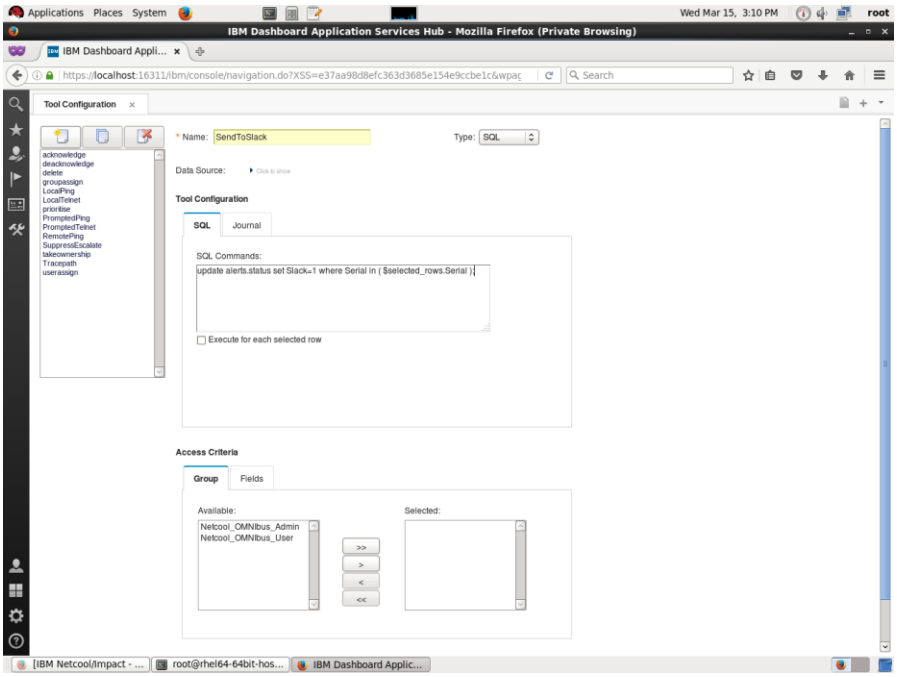


9. Login to DASH at http://localhost:16311/ibm/console with the user/password of ncoadmin/nocadmin (note that you cannot be logged into to DASH and Impact using the same browser session – you will either need to logout of Impact or open a private tab for the DASH session)

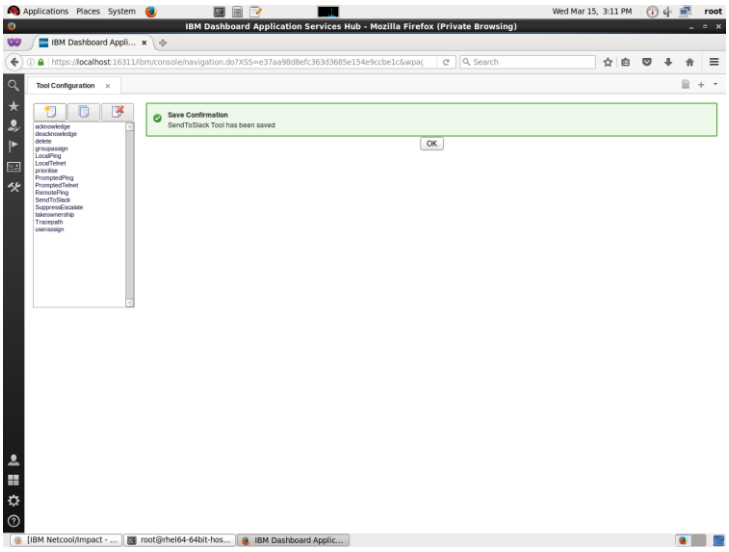
10. Open the "Tool config" menu



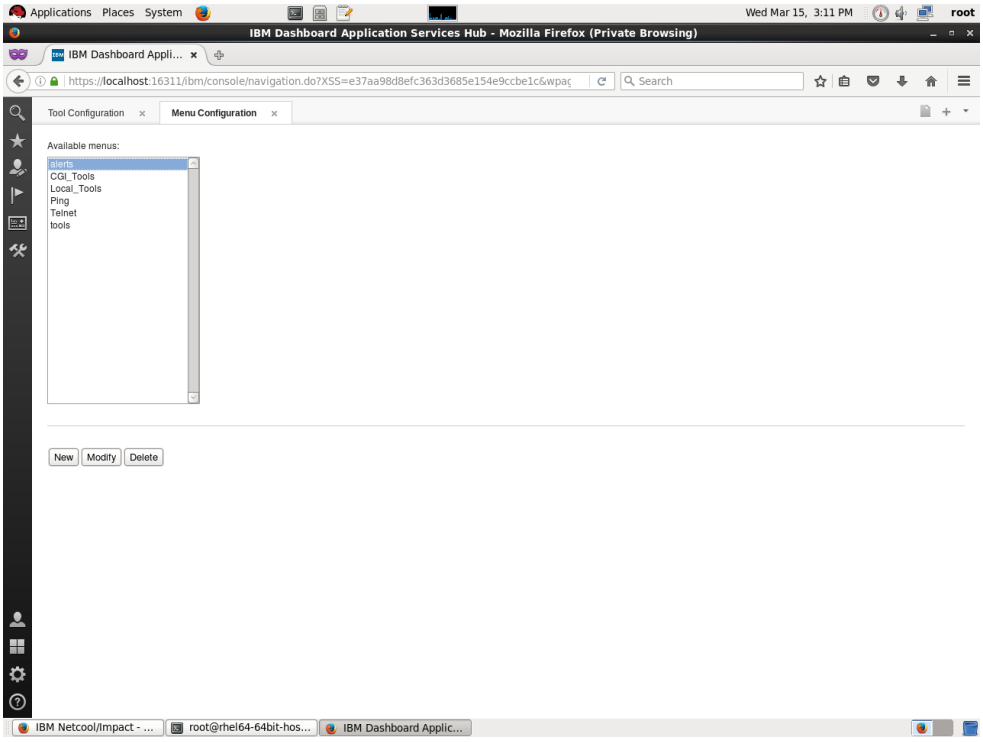
11. Create a new SQL tool called "SendToSlack" and enter the following command : `update alerts.status set Slack=1 where Serial in ($selected_rows.Serial);`



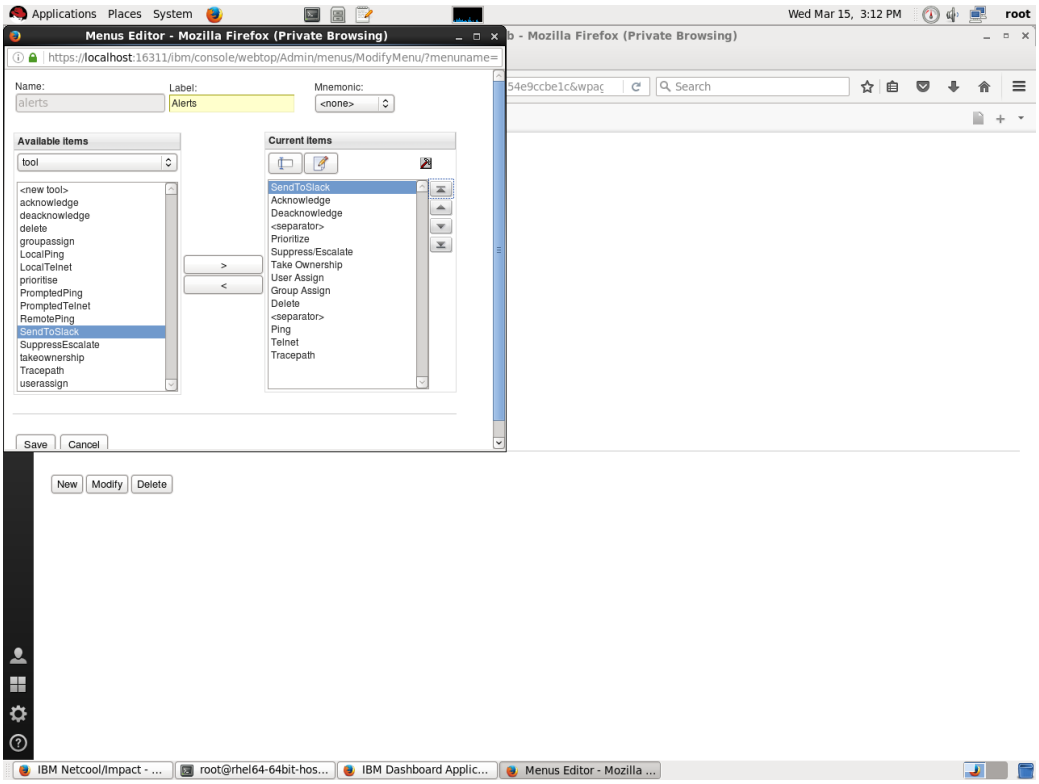
12. Save the tool (you may need to scroll down to see the save button)



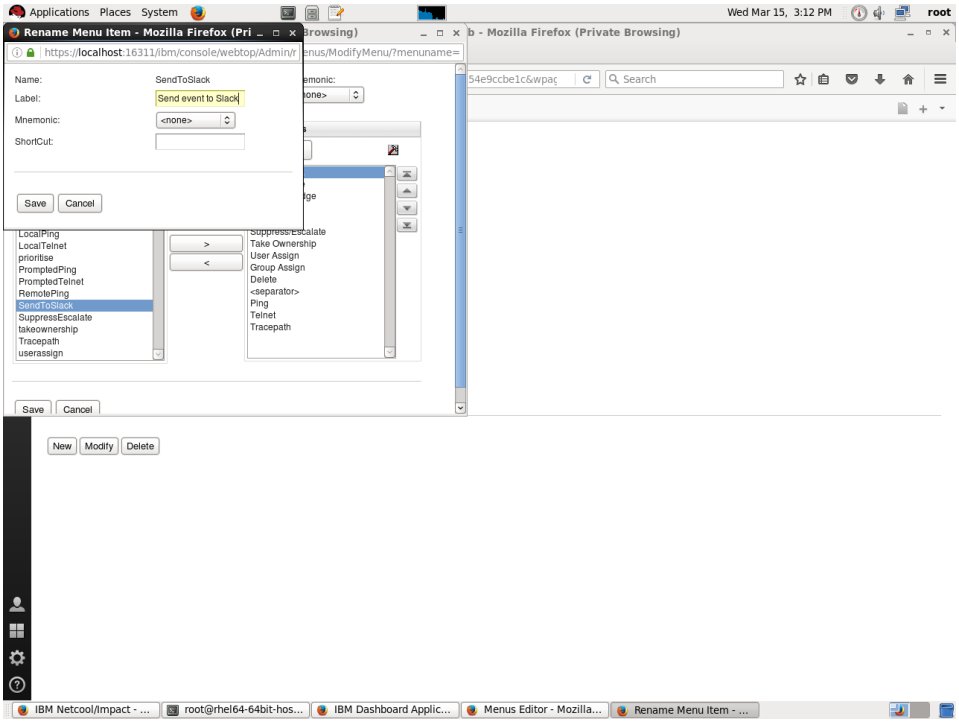
13. Open the "Menu Configuration" tab and choose the "alerts" menu to modify



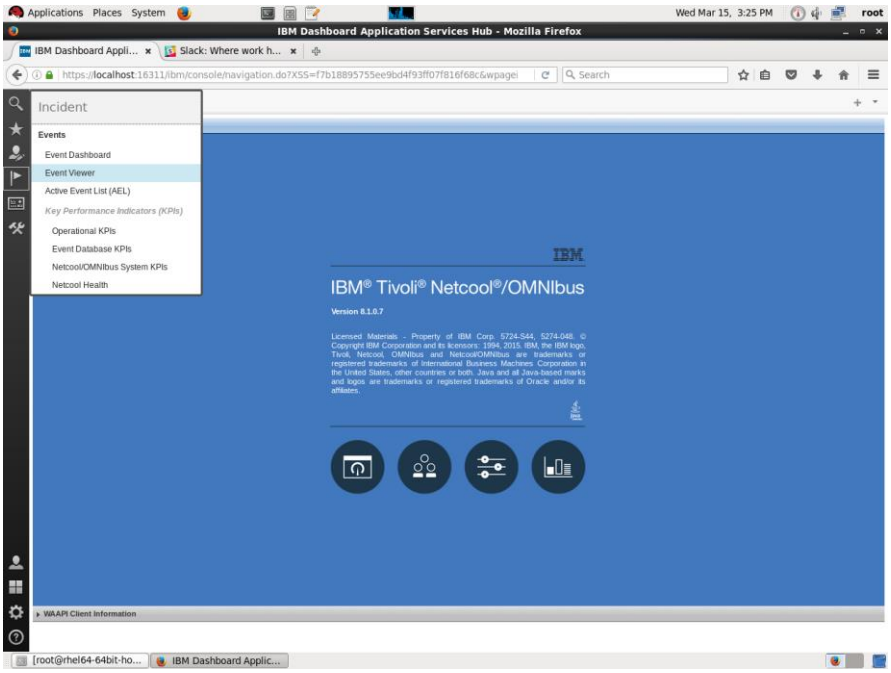
14. Modify the "alerts" menu by moving the SendToSlack tool to the right



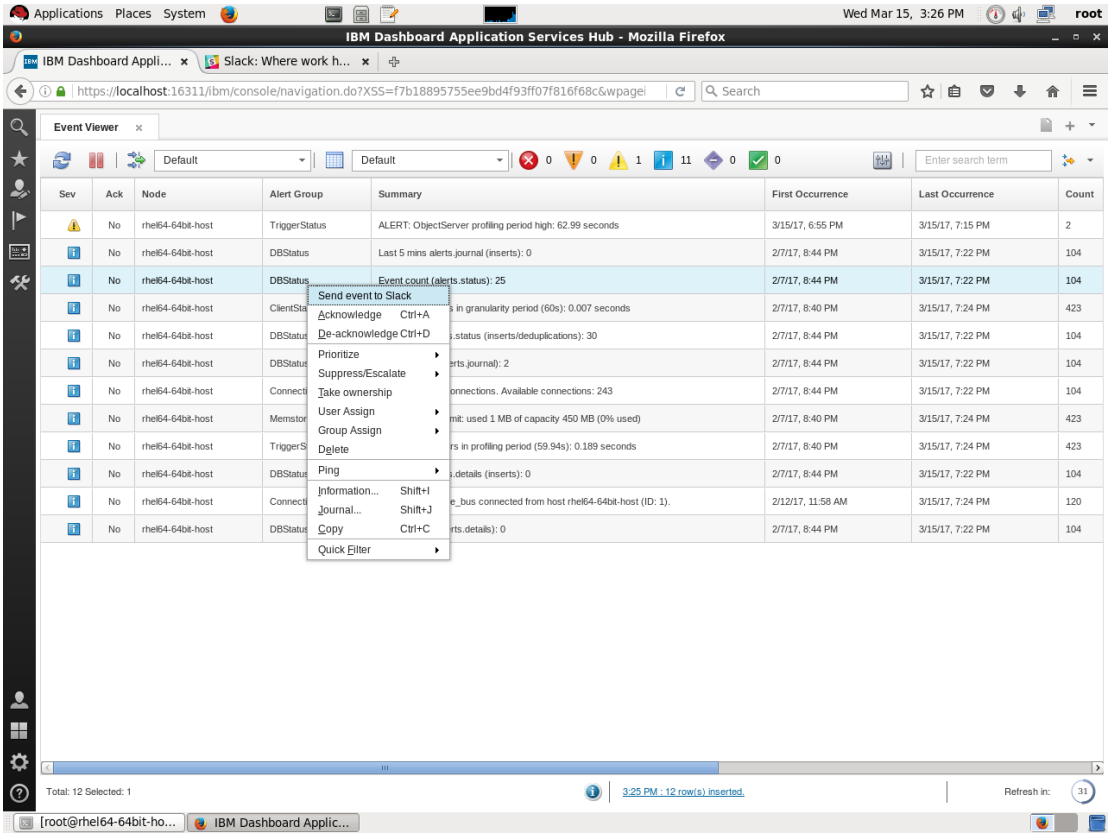
15. Rename the tool in the menu to "Send event to Slack"



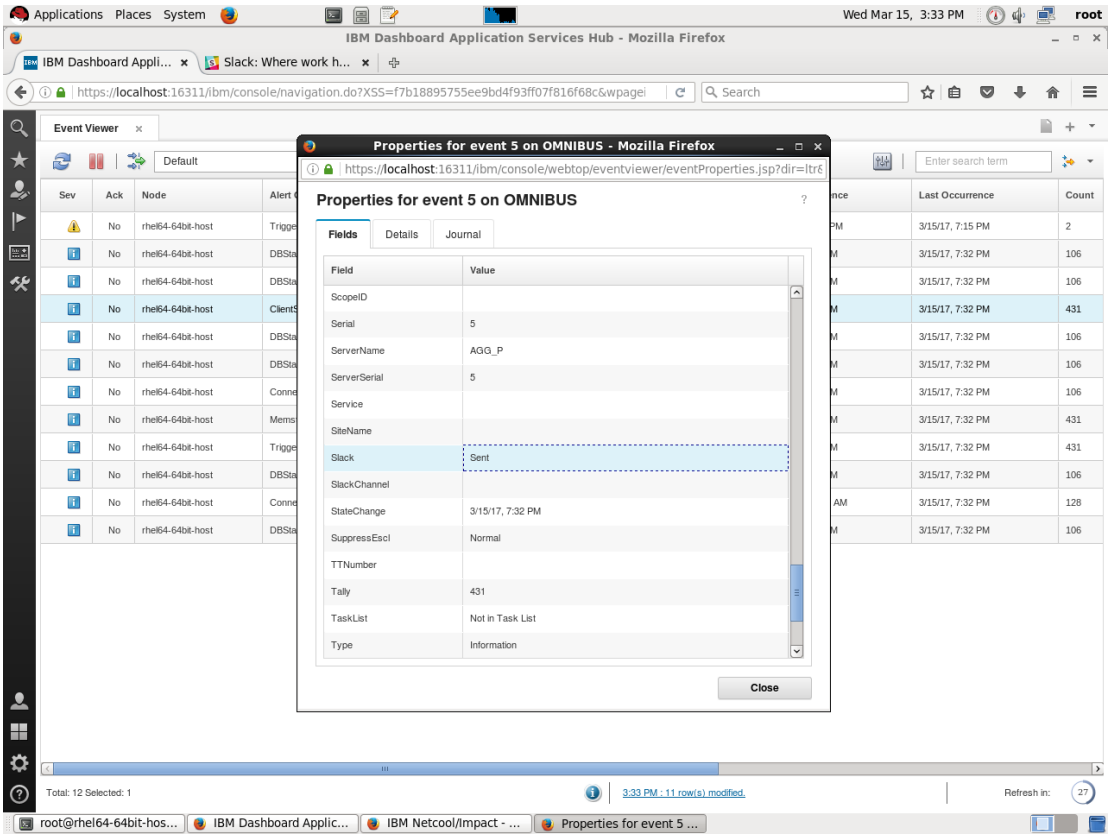
16. Open the event viewer tab



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



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



19. Check Slack and see if your event appears

Format the event sent to Slack.

20. 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);
```

21. Correct the values for the Slack token and the DASH IP.
22. 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: 8080
NRestOS.Enable: TRUE
```

2. Restart the objectserver with the following commands:
   /opt/IBM/tivoli/netcool/omnibus/bin/nco_pa_stop -server host-1_PA -process MasterObjectServer
   /opt/IBM/tivoli/netcool/omnibus/bin/nco_pa_stop -server host-1_PA -process MasterObjectServer
   Use the root password when asked.

3. Test that you have remote access by running the command
   `netstat -nap | grep 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 coffee-script

This command installs a bot "generator".
```

6. Run the following command as root:

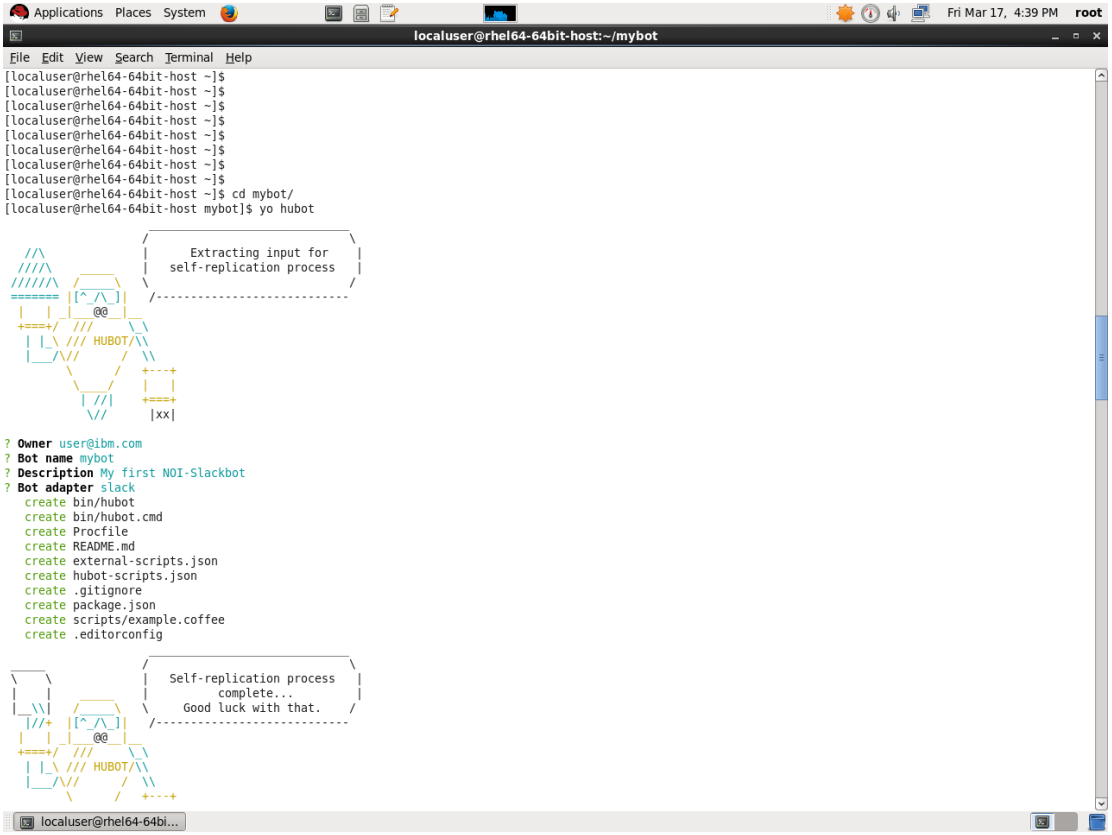
```
npm install -g node-omnibus
```

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 ibmuser

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"
```

11. 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: '7070'

    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"
```

12. Run the bot using the command
    `HUBOT_OMNIBUS_PASSWORD=passw0rd HUBOT_OMNIBUS_HOST=localhost PORT=8080 HUBOT_SLACK_TOKEN=<<YOURTOKENHERE>> ./bin/hubot --adapter slack`
13. From the Slack window, invite the bot to your slack channel with the command /invite <botname>
14. 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.
15. 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"
```

16. 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.
17. 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]
```

18. 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.

19. 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)"
```

20. 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.
21. 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
```

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

We Value Your Feedback!

- Don’t forget to submit your Think 2018 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Access the Think 2018 agenda tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.