# Using IBM DevOps Services & Bluemix Services
## Part 3: Planning, Tracking, & Leveraging a Text Messaging Service

This series of labs demonstrates how easy it is to use IBM DevOps Services and Bluemix together to plan, code, and deploy applications. Each part of the workshop guides you through the steps to experiment with key DevOps Services features

- Part 1: Deploy and make change to a simple application.
- Part 2: Add a data management Bluemix service and enable automatic deployment.
- Part 3: Use the planning capabilities to add a text messaging service, and enable traceability between code changes and work items.
- Part 4: Leverage automated builds and deployments, then explore integration between DevOps Services and Rational Team Concert (RTC)

After completing each part of the workshop, as a bonus, you'll also get applications that you can use to demonstrate DevOps Services and Bluemix capabilities to your colleagues and friends.

### Learning Objectives

In this lab, you will learn how to do the following:

- Use the planning capabilities in IBM DevOps
- Add a text messaging service in Bluemix
- Add links between the stories (plan) and the change sets (code)

### Time Required

This workshop takes about 45 minutes to complete.

### Before you begin

This lab is a continuation of Part 2 - Deploying an App that Uses a Data Management service. If you have not completed part 2, please go to http://bluelabs.myBluemix.net first.
You'll also need one of the following modern browsers:

- Firefox 15 / Chrome 21 / Internet Explorer 10 / Safari 7 or later versions of thses browser

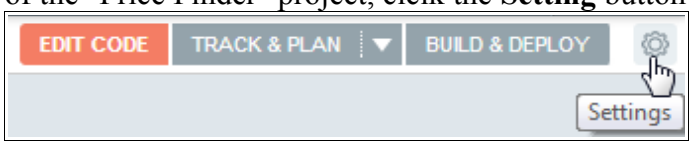
### About the App

In this workshop, you'll continue to work with the apps you built in parts 1 and 2: "Lovely Landscapes" and "Price Finder." You will enhance "Price Finder" application so it has a text messaging capability.

### 1- Enable agile planning

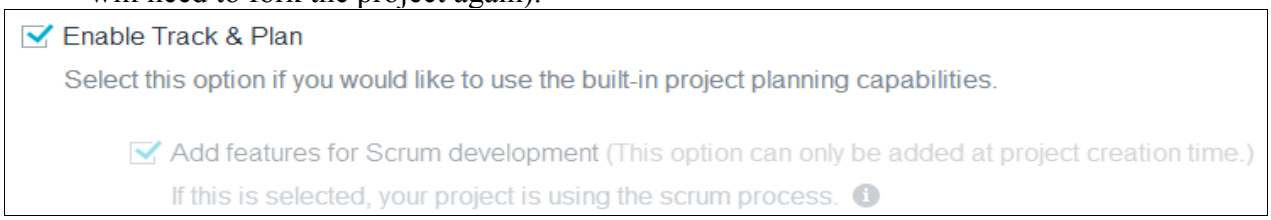
When working with teammates or even by yourself, organizing your ideas into stories and then organizing your stories into a plan can be incredibly valuable.

1. In the web IDE of the "Price Finder" project, clcik the **Setting** button

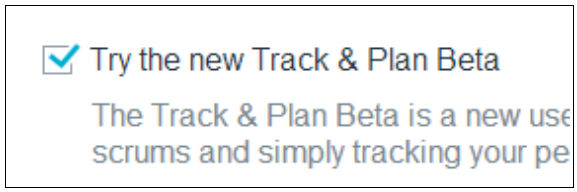



2. On the left pane of the Settings page, click the OPTION menu. Verify that the **Track & Plan** capability is enabled, and the Scrum development feature has been added to the project. (if

not, this is because you forgot to check the option when you forked the project in lab 2. You will need to fork the project again).




3. On the left pane, click the BETA FEATURE menu. Check the "Try the new Track & Plan Beta" option and click **SAVE**.




4. Now that Agile Tracking and Planning is enable, click the TRACK & Plan button on the top of the screen. You are ready to plan your work using the Scrum approach.

**2- Plan your work**

On an agile project, you first need to populate the backlog with stories and epics. Then you select stories that you will deliver in the upcoming sprint.

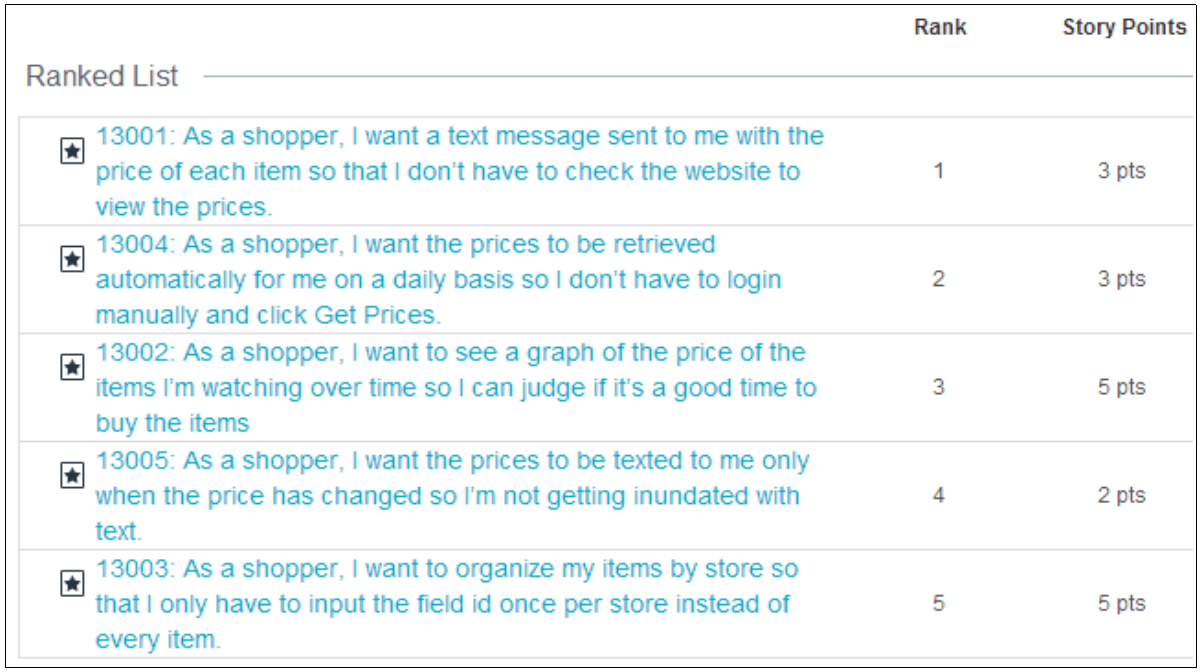
1. On the left pane, select **Sprint Planning**, then click the **Add Sprints** button.
2. Configure the sprints so you have at least two and click **OK**. At any point in time, you can Click the Edit Sprints link on the left pane to reconfigure your Sprints.


3. On the left pane, click **Backlog.** At this point the backlog is empty but you will add new items
4. In the Create a Work Item section, enter: "`As a shopper, I want a text message sent to me with the price of each item so that I don't have to check the website to view the prices.`" Notice that you also have options add additional information to the work item, such as type, description, subscribers, owner, due date, tags, priority, or parent.

5. Click **SAVE**. The work item is created. Its type is automatically set to "Story" as the tool recognize the typical pattern used for stories (As a <type of user>, I want <some goal> so that <some reason>)

6. Repeat similar steps to create four other stories
    - `As a shopper, I want to organize my items by store so that I only have to input the field id once per store instead of every item.`
    - `As a shopper, I want to see a graph of the price of the items I'm watching over time so I can judge if it's a good time to buy the items.`
    - `As a shopper, I want the prices to be retrieved automatically for me on a daily basis so I don't have to login manually and click Get Prices.`
    - `As a shopper, I want the prices to be texted to me only when the price has changed so I'm not getting inundated with text.`

7. In the Ranked List section, move (drag and drop) the first story that you created on the top of the backlog (`"As a shopper, I want a text message sent to me..."`). This story is now ranked as the first one.
8. For this story, change the Story Points to 3. It gives an indication of the effort needed to implement this story.
9. Optionally, you can reorder the other items on the backlog and change their Story Points value. You know have an ordered backlog. You are ready to proceed with Sprint planning.




10. On the left pane, click the **Sprint Planning** link. The backlog and Sprint 1 are displayed on the same page to facilitate planning activities.
11. To assign work to the first sprint, move the firts story from the backlog to Sprint 1. (`"As a shopper, I want a text message sent to me..."`).
12. Change the owner for this story and assign yourself.

13. Change the status of this story to **Start Working**. You are ready to implement the story.

On a real project, you would assign other stories to sprint 1 but we stop here toi simplify this lab.

### 3- Add the Twilio messaging service to your Bluemix application

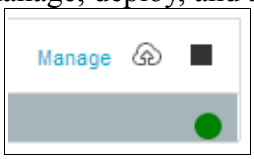
In this section we will implement the story "As a shopper, I want a text message sent to me with the price of each item so that I don't have to check the website to view the prices." The idea of implementing a text messaging service may sound a bit daunting to you.
Fortunately, Bluemix provides a Twilio service we can leverage that will handle sending the text message, so we will be able to implement this feature with only nine lines of code!

If you do not already have an account with Twilio, you'll need to register. To complete this lab, you need to be registered to Twilio. You'll also need your Twilio AccountSID, Auth Token, and phone number.
Warning: Depending on your mobile phone plan, you may be charged by your operator for Twilio text messages sent.

1. Navigate to https://www.twilio.com and log in.
2. Click on your id in the right corner and select **Account Settings**.
3. In the **API Credentials (Live)** section, note your `AccountSID` and `AuthToken`. (You may want to leave this page open as you continue, so you can easily copy the credentials.)

   We'll begin by adding the Bluemix Twilio service to our app.

4. In the DevOps Services web IDE of your "Price Finder" project, click **EDIT CODE**.
5. Scroll drown the right pane to find the Manual Deployment Information table.
6. Single click on the row in the Manual Deployment Information table representing your deployment. It reveals links to manage, deploy, and stop your application.




7. Click the **Manage** link. The Bluemix page for your app is displayed (you may be asked to login to Bluemix).
8. In the Services section, click **ADD A NEW SERVICE.**
9. Scroll down until you see the **Twilio** service (it will likely be in the Mobile section–the catalog is constantly being updated with new services!).
10. Click on the Twilio service. In the Twilio dialog, add your your Twilio `AccountSID` and `AuthToken` (copy them from the API Credentials (Live) section of your Twilio Account Settings page).
11. Click **CREATE**. The Twilio service is added to you Bluemix application
12. When prompted if you want to restart your application now, click **OK**. You will know your application has finished restarting and is running successfully when you see a green status dot in the Health section.

### 4- Add code to use the Twilio service

Now that your application on the cloud contains a Twilio service, you must add code to send a text message whenever a price is retrieved for an item.

The "price Finder" application will use the Twilio module, so you need to we need to add it to requirements.txt.

1. In the web IDE, open requirements.txt (root folder)
2. Add a new line to the end of the file and type “twilio” (without the double quotes). It tells Bluemix to locate the Twilio libraries.
3. Select ***File > Save.***

In order to effectively leverage the Twilio service, you need to connect to it.

4. In the web IDE, open wsgi.py (root folder)
5. In the imports section, around line 55, uncoment (remove character "#") the line about twilio import: `from twilio.rest import TwilioRestClient`
6. As part of the for loop that begins around line 67, uncomment the following two lines (Note: Be sure that identation of the source code (spaces and tabs) is identical to the example below:

```
if decoded_config[key][0]['name'].startswith('Twilio'):
      twilio_creds = decoded_config[key][0]['credentials']
```

7. Your Twilio credentials are now stored in twilio_creds. Beneath the code you just modified, uncomment the following three lines (by removing three quotes at the begenning and the end of the block)

```
twilio_authToken = twilio_creds['authToken']
twilio_accountSID = twilio_creds['accountSID']
twilioClient = TwilioRestClient(twilio_accountSID, twilio_authToken)
```

8. Continuing to work in wsgi.py, uncomment the block (around line 130) that defines the text messaging function. Fake phone numbers must be replaced with your real phone numbers.

```
def sendTextWithMessage(message):
      destination_number="replace-fake-TO-number"
      print "Sending text message to mobile phone: " +
            destination_number
      message = twilioClient.messages.create(to=destination_number,
            from_="replace-fake-FROM-number", body=message)
      print "Text message sent using Twilio to mobile phone: " +
            destination_number
```

9. Replace the **to** number with **your mobile phone number** (the one used to register to twilio). It will receive the text messages
10. Replace the **from** number with **your Twilio phone number** (the one Twilio assigned to you when you registered)
11. In the **getCurrentPrice** function, uncomment (remove #) the line that sends the text message (around line 152)

```
sendTextWithMessage("The current price of %s is %s" % (item["name"],
price))
```

12. Save your changes by selecting ***File > Save.***

That’s it! With about 9 lines of code, you have implemented a text messaging feature in your Bluemix application.

**5- Deploy and test the text messaging feature**

The code of your application has been modified to leverage a text messaging service (Twilio). All you need now is to deploy and test the new feature.

1. Click the **DEPLOY** button.
2. Once your application has finished deploying, click on the link in the green bar to open the root folder page.

3. On the root folder page, scroll down the right pane until you see the Manual Deployment Information section.
4. Click *PriceFinder* in the Manual Deployment Information section to open your application.
5. Click **Get Prices**.
6. Check your mobile phone for a text message!
7. While you are waiting for the text message, go to the DevOps Services Root Server Page Manual Deployment Information section and click the **Logs** link
8. On the Logs page, select the **stdout.log** file to display its content. You can verify that the messaging service has been called. The log contains a message regarding Twilio: *"Sending text message to mobile phone: <your-phone-number>"*

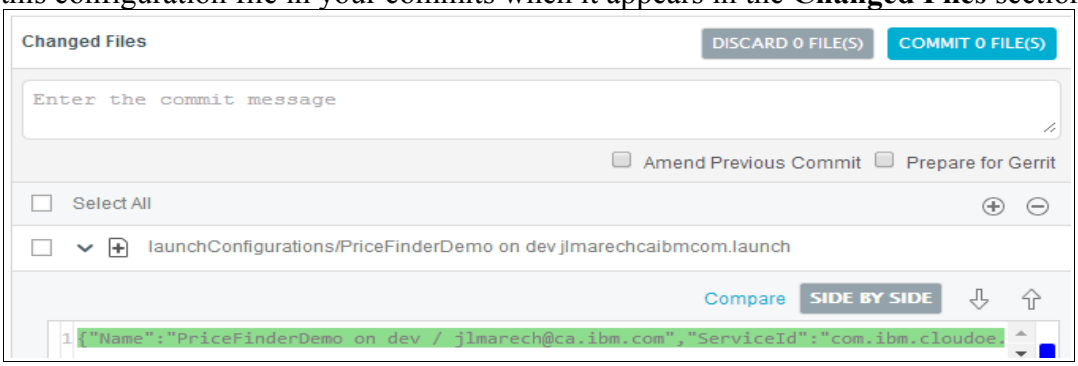
If you'd like, update the prices of the items in the store and get the prices again.
(Be careful, depending on your mobile phone plan, you may be charged by your operator for Twilio text messages)

**6- Push your changes to the repository**

Now that your feature is tested, it's time to push your changes to the Git repository. You will also link your changes to the story in the plan for better traceability.

1. First, go to the DevOps Services project and click the **TRACK & PLAN** button.
2. Select **My Work** and click **IN PROGRESS**. The story you have just implement should be displayed (otherwise, you probably missed a step in *2- Plan your work* and you need to find the story from Sprint 1)
3. Check the story id (number). You will need it soon.
4. Click the **EDIT CODE** button. In the leftmost bar of icons, click the **Git repository** icon.
5. Under the Changed Files section, add a comment that contains the story number,
   eg: `Added Twilio code for story 13001`
6. Check the **Select All** box, then click the COMMIT button.
7. Then PUSH your changes from the OUTGOING section.

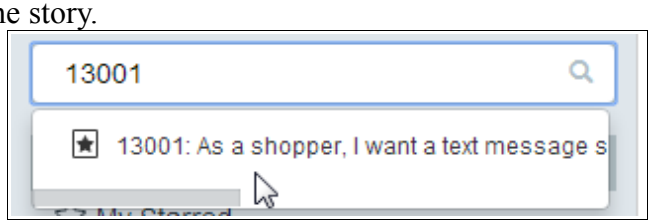
**Note:** The first time that you deploy an application on Bluemix, DevOps Services creates a configuration file, under folder launchConfigurations. This file contains information to facilitate and speed up future deployments.
Include this configuration file in your commits when it appears in the **Changed Files** section.




**7- Report your progresss in the plan**

The new feature has been developed and the code has been delivered to the Git repository. Now it is time to update the story to indicate that the work is completed.

1. In the web IDE, click the **TRACK & PLAN** button.
2. In the upper-left corner, enter the story number in the search box, then click the link to open the story.




3. Change the status for the story by selecting **Complete Development**. Click **SAVE**. The new status is **Implemented**.
4. Because you have already tested the story, change the status again by selecting **Complete Testing**. Click **SAVE**. The new status for the story is now **Done**.
5. Below the story summary, click **LINKS**. Notice that this story is associated to a change set (the changes you pushed to the Git repository)


Congratulation! The Price Finder application has been enriched with a nice text messaging feature.

**Summary**
In this lab, you learned how to:

- Enable the Track & Plan capability on an DevOps Services project.
- Use the Agile planning feature (track and plan)
- Add a text messaging service to a Bluemix application
- Report progress in stories
- Enable traceability between change sets (Git) and stories (plans)

**Next Steps…**
Continue on to PART 4 where you'll learn how to use the Rational Team Concert with IBM DevOps Services. You will also experiment with the advanced deploy option (Delivery Pipeline) to automatically build and deploy a Java application.
http://bluelabs.mybluemix.net