Certified for IBM Cloud and Mobility

IBM Cloud Professional Certification Program

Study Guide Series

Exam C5050-285 - IBM Cloud Platform Application Development V1

Contents

Purpose of Exam Objectives

When an exam is being developed, the Subject Matter Experts work together to define the role the certified individual will fill. They define all of the tasks and knowledge that an individual would need to have in order to successfully perform the role. This creates the foundation for the objectives and measurement criteria, which are the basis for the certification exam.

The certification item writers use these objectives to develop the questions that they write and which will appear on the exam.

It is recommended that you review these objectives. Do you know how to complete the task in the objective? Do you know why that task needs to be done? Do you know what will happen if you do it incorrectly? If you are not familiar with a task, then go through the objective and perform that task in your own environment. Read more information on the task. If there is an objective on a task there is about a 95% chance that you WILL see a question about it on the actual exam.

After you have reviewed the objectives and completed your own research, then take the assessment exam. While the assessment exam will not tell you which question you answered incorrectly, it will tell you how you did by section. This will give you a good indication as to whether you are ready to take the actual exam or if you need to further review the materials.

Section 1: Hosting Cloud Applications

a. **Describe Cloud service models and IBM Cloud offerings**

1. IBM SoftLayer Infrastructure as a Service (IaaS)

   IBM SoftLayer provides self-service deployment of virtual and dedicated bare metal servers in secure data centers around the world. It provides the hosting infrastructure used by IBM Bluemix Public and IBM Bluemix Dedicated services.

2. IBM Bluemix Platform as a Service (PaaS)

   IBM Bluemix Platform as a Service is built on Cloud Foundry open source technology. It makes application development easier by allowing developers to focus on code and data alone. Bluemix Platform as a Service handles managing middleware, operating systems, and hosting infrastructure automatically.

3. Software as a Service (SaaS) and IBM Cloud Marketplace

   The Software as a Service model provides business applications on a ready-to-use and self-service model. The IBM Cloud Marketplace is a place where enterprise cloud customers can discover, learn, try and buy cloud services from IBM and Business Partners.

b. **Describe the different capabilities of IBM Bluemix**

1. IBM Bluemix PaaS provided by Cloud Foundry

   Bluemix PaaS gives instant access to runtimes for applications. Cloud Foundry is an open source PaaS that offers developers the ability to quickly compose their apps without worrying about the underlying infrastructure. Bluemix extends Cloud Foundry with a number of managed runtimes and services, enterprise-grade DevOps tooling, and a seamless overall developer experience.

2. IBM Bluemix Containers using Docker

   IBM Containers allow portability and consistency regardless of where they are run—be it on bare metal servers in Bluemix, your company's data center, or on your laptop. Easily spin up images from our public hub or your own private registry using the native Docker CLI.

3. IBM Bluemix virtual machines powered by OpenStack

   Virtual machines offer the most control over your apps and middleware. Bluemix uses industry-leading OpenStack software to run and manage VMs in a public cloud, a dedicated cloud, or your own on-premises cloud. Key OpenStack services such as Auto Scaling, Load Balancing, and Object Storage can be used in conjunction with Bluemix services to build and run hybrid apps.

Section 2: Planning Cloud Applications

a. **Describe key components of IBM Bluemix PaaS environment**

1. *Runtime* is a set of resources to run an application. Bluemix provides runtime environments as containers for different types of applications. The *runtime environments* are integrated as buildpacks into Bluemix, are automatically configured for use, and require little to no maintenance.

2. *Boilerplate* consists of a combination of runtime and predefined services. Because they contain a runtime and set of services for a particular solution type, they can be used to quickly get an application up and running.

3. *Services* provide ready-for-use functionality for a running application and are represented by two types:
   a. Managed services are listed in the IBM Bluemix PaaS catalog. A Managed Service integrates with Bluemix/Cloud Foundry via a service broker that implements the Service Broker API. The service broker advertises a catalog of service offerings and service plans to Bluemix/Cloud Foundry and receives calls from Cloud Foundry for four functions: create, delete, bind, and unbind.
   b. User-provided services can be used by developers to programmatically define services outside of IBM Bluemix PaaS. This gives a mechanism to provide credentials to applications for service instances which have been pre-provisioned outside of Bluemix/Cloud Foundry.

b. Describe components of IBM Bluemix PaaS architecture based on Cloud Foundry

1. The Droplet execution agent (DEA) manages application instances, tracks started instances, and broadcasts state messages. Application instances live inside Warden containers. Containerization ensures that application instances run in isolation, get their fair share of resources, and are protected from noisy neighbors.

2. The Cloud Controller is responsible for managing the lifecycle of applications. When a developer pushes an application to Cloud Foundry, the Cloud Controller stores the raw application bits, creates a record to track the application metadata, and directs a DEA node to stage and run the application.

3. The Router routes incoming traffic to the appropriate component within the environment. For example, to the Cloud Controller for management of applications in their lifecycle or to a running application on a DEA node.

4. Service Broker advertises a catalog of service offerings and service plans to Bluemix/Cloud Foundry, and receives calls from Cloud Foundry for four functions: create, delete, bind, and unbind. The broker then passes these calls onto the service itself.

5. The interaction between DEA and Health Manager ensures consistency in the running application state. The DEA monitors the state of a started application instance and periodically broadcasts application state messages. These state messages are consumed by the Health Manager. The health manager is responsible for keeping the expected application state consistent with the actual application state.

Reference: https://docs.cloudfoundry.org/concepts/architecture/

c. Explain the process of staging an application in IBM Bluemix PaaS

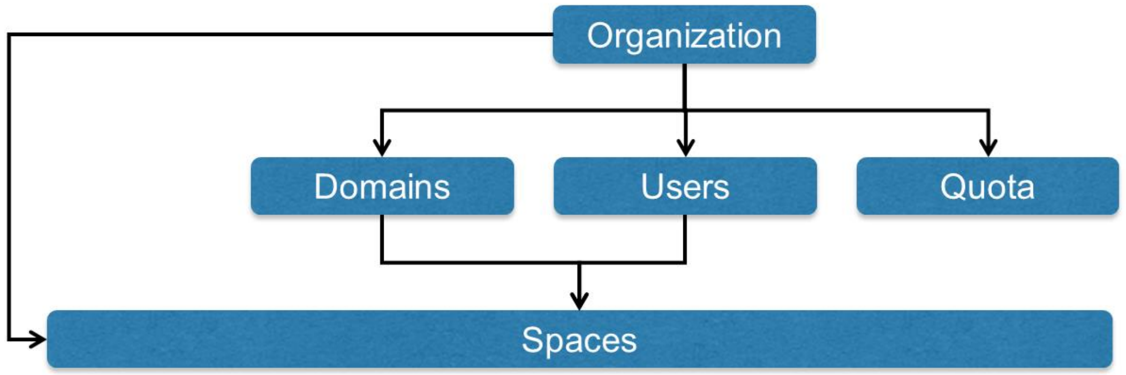
When using the cf cli tool, application staging begins with the push command which creates a series of interactions between the cli, Cloud Controller, and DEA(s) in the environment.

- The cf tool provides information to the Cloud Controller about the application including name, number of instances and buildpack. It uploads the application files to the Controller.
- The Cloud Controller selects a DEA instance from the pool to stage the application.
- The DEA uses instructions in the buildpack to stage the application and streams the output to the terminal running the cli command, allowing the developer to follow progress. Once built, the application is stored as a droplet and pushed to the blobstore.
- The Cloud Controller then selects one or more DEAs based on the desired number of instances and then instructs them to run the droplet.
- As the applications start, the DEAs report status back to the Cloud Controller and begin broadcasting state messages that are monitored by the Health Manager.

Reference: https://docs.cloudfoundry.org/concepts/how-applications-are-staged.html

d. Describe the organization management elements in IBM Bluemix PaaS: Spaces, Users, Domains and Quota

The organization is the key grouping concept in IBM Bluemix. Each organization is composed of several elements.



1. *Spaces* provide a mechanism to collect related applications, services and users. Every organization contains at least one space. All applications and services are associated with a single space.
2. *Users* participate in organizations and have varying capabilities based on assigned role. Users may be members of multiple organizations and can change from one organization to another through the Bluemix dashboard. Users may have different roles in different spaces within an organization, controlling the type of access they have within the space.
3. *Domains* provide the route on the Internet for the organization. An application route is composed of both a hostname which defaults to the application name plus the domain name. A custom domain name can be configured in the organization and then assigned to an application within the organization to form a custom Internet endpoint for the application in IBM Bluemix PaaS.
4. *Quota* defines resource limits for the organization. This represents and controls the number of services and the amount of memory available for applications within all spaces of the organization.

e. Understand IBM Bluemix Regions and how to manage applications in multiple regions

A Bluemix region is a defined geographical territory where applications are deployed. Applications and service instances may be created in different regions with the same Bluemix infrastructure for application management and the same usage details view for billing. Regions allow applications to be deployed closer to customers to reduce application latency or to localize application data to address security and privacy concerns. When deploying applications in multiple regions, if one region goes down, the applications that are in the other regions will continue to run.

When using the IBM Bluemix PaaS user interface, you can switch to a different region to work with the spaces in that region.

When using the cf command line interface, connect to the desired Bluemix region by using the cf API command and specifying the API endpoint of the region. For example, enter the following command to connect to Bluemix Europe United Kingdom region:

```
cf api https://api.eu-gb.bluemix.net
```

When using the Eclipse tools, connect to the desired Bluemix region by creating a Bluemix server and specifying the API endpoint of the region.

For more information on regions and a table of regions and API endpoints see:
https://www.ng.bluemix.net/docs/overview/overview.html#ov_intro

f. Use the Cloud Foundry CLI (cf) tool to manage applications in IBM Bluemix PaaS

1. Specify the region through api endpoint

   The region may be set using either the `cf api <API_URL>` or by using the `-a <API_URL>` option to the login command where `<API_URL>` is the API endpoint of the desired region.

2. Log in to an organization and space using cf

   Command syntax for cf login:
   ```
   cf login [-a API_URL] [-u USERNAME] [-p PASSWORD] [-o ORG]
   [-s SPACE]
   ```
   Where `API_URL` is the API endpoint of the desired region, `USERNAME` is the user, `PASSWORD` is the specified user's password, `ORG` is the organization name and `SPACE` is the space. Any needed argument not provided on the command line will be prompted. For example you may provide the password interactively at the terminal by omitting the `-p` option on the command line.

3. Push an application using cf and understand applicable options

   Command and syntax to push an application:

```
cf push APP [-b URL] [-c COMMAND] [-d DOMAIN] [-i
NUM_INSTANCES] [-m MEMORY] [-n HOST] [-p PATH] [-s STACK] [--
no-hostname] [--no-route] [--no-start]
```

Where `APP` is the application name. This command should be run in the top level directory containing the application and a copy of the manifest.yml file. Some common options in a push are the `-c` to specify a startup command, `-i` to specify number of instances at startup, `-m` to specify memory used by the application instance at startup, and `--no-route` to prevent connecting a route to the application.

4. View logging information using cf

   To view logs from an application use:

   ```
   cf logs APP [--recent]
   ```

   Where `APP` is the application name. When the `--recent` option is specified, the most recent log history is sent to the terminal and the command ends. If specified without this option, the command streams log output to the terminal.

5. Perform scaling (instance, memory, disk) of an application using cf

   The cf scale command may be used to perform horizontal or vertical scaling. For horizontal scaling by increasing the number of instances, use:

   ```
   cf scale APP -i INSTANCES
   ```

   Where `APP` is the application name and `INSTANCES` is the desired number of running instances.

   For vertical scaling by increasing memory capacity, use:

   ```
   cf scale APP -m MEMORY
   ```

   Where `APP` is the application name and `MEMORY` is an integer followed either an **M**, for megabytes, or **G**, for gigabytes.

   For vertical scaling by increasing disk space for instances of the application, use:

   ```
   cf scale APP -k DISK
   ```

   Where `APP` is the application name and `DISK` is an integer followed by either an **M**, for megabytes, or **G**, for gigabytes.

6. cf commands for managing domains, routes, organization and spaces

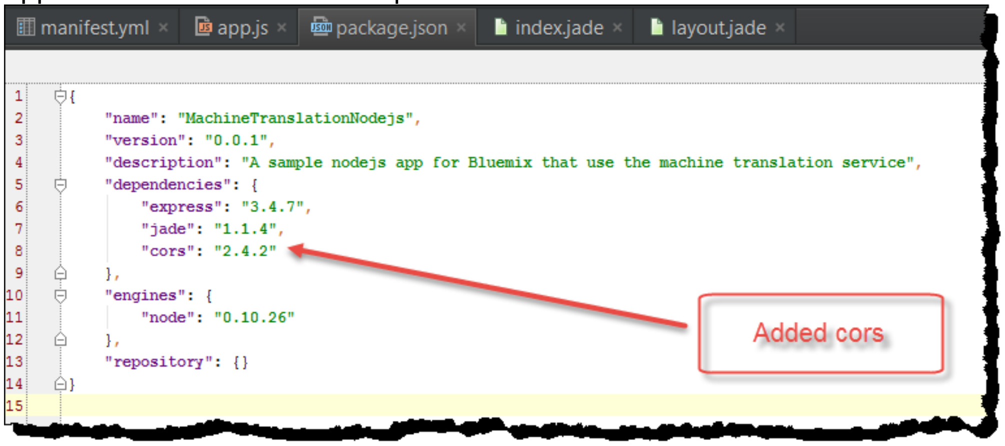
Reference: http://docs.cloudfoundry.org/devguide/installcf/whats-new-v6.html#domains-etc

Section 3: Implementing Cloud Ready Applications

a. **Understand how to design, develop, deploy and manage a IBM Bluemix PaaS application following the Twelve-Factor App methodology** (http://12factor.net/)

Factor 1: One codebase tracked in revision control, with multiple deployments
Use one code base track and revision control with many deploys. Utilize IBM Bluemix DevOps Services or Cloud Foundry deployment tools in combination with tools like IBM UrbanCode Deploy, Jenkins, Gradle, or equivalent.

Factor 2: Explicitly declare and isolate dependencies
IBM Bluemix PaaS deployable units e.g. Java Liberty and Node.js buildpacks manage runtime dependencies. For example, the package.json file for a Node.js application lists all external dependencies:

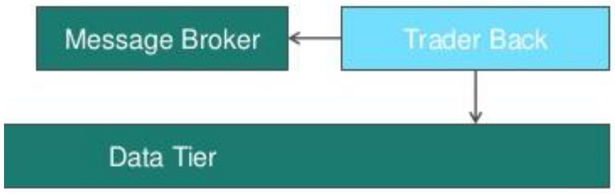


```
{
    "name": "MachineTranslationNodejs",
    "version": "0.0.1",
    "description": "A sample nodejs app for Bluemix that use the machine translation service",
    "dependencies": {
        "express": "3.4.7",
        "jade": "1.1.4",
        "cors": "2.4.2"
    },
    "engines": {
        "node": "0.10.26"
    },
    "repository": {}
}
```

Factor 3: Store configuration in the environment
When running in IBM Bluemix PaaS, access all configuration information through the VCAP_SERVICES environment variable and do not store configuration items, for example a tcp port or API key, as constants in the application code. Services that are bound to an application also populate VCAP_SERVICES with their configuration attributes.

Factor 4: Treat backing services as attached resources
Bind services to applications using the Bluemix PaaS dashboard or use the `cf create-service` and `cf bind-service` commands to attach services to an application. To update to a new service version, unbind the old service and then bind the new service instance to the application.

```
cf push traderback
cf bind-service traderback stsql
cf bind-service traderback stmessaging
```



Factor 5: Strictly separate build and run stages
The IBM Bluemix PaaS implementation separates the build process performed during `cf push` from the immutable container image used when creating an application instance.

Factor 6: Execute the app as one or more stateless processes
When designing applications avoid monoliths and use multiple processes or services as needed. IBM Bluemix PaaS application instances are stateless and do not have

persistent file storage. Avoid dependency on sticky sessions and keep session data in a persistent store to ensure traffic can be routed to other processes without service disruption.

Factor 7: Export services via port binding
IBM Bluemix PaaS provides a process or service with a port for binding and then handles routing traffic to the process over this port automatically. Application code reads the port from the environment and binds to this port accordingly.

Factor 8: Scale out via the process model
Horizontal scaling of application instances in IBM Bluemix PaaS may be explicitly performed using the `cf scale` command or automatically scaled using the Auto-Scaling service.

Factor 9: Maximize robustness with fast startup and graceful shutdown
Use a disposable approach to the design of a process in the application. There should be minimal startup actions required. When a process is terminated, it should be able to go away with minimal housekeeping. This improves robustness and responsiveness to horizontal scaling events.

Factor 10: Keep development, staging, and production as similar as possible
Spaces in IBM Bluemix PaaS provide an effective method to separate different levels of an application. This approach enables agile software delivery and continuous integration.

Factor 11: Treat logs as event streams
In IBM Bluemix PaaS, processes should write log data as an unbuffered event stream to standard out. The IBM Bluemix PaaS Loggregator accumulates log data across various components of the application and Cloud Foundry environment and provides it for viewing using `cf logs` or exporting to a third-party logging service.

Factor 12: Run admin/management tasks as one-off processes
Design tasks that need to run once or occasionally into separate components that can be run when needed instead of adding the code directly into another component. For example, if an application needs to migrate data into a database, place this into a separate component instead of adding it to the main application code at startup.

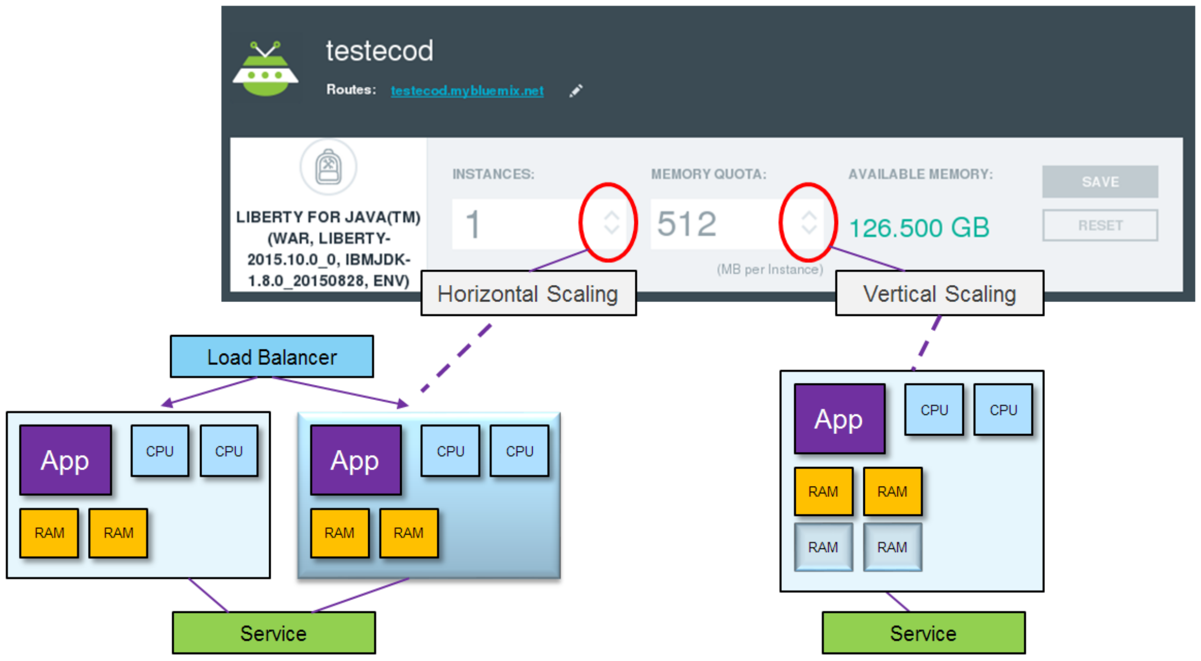
b. **Understand scaling concepts for a Cloud application and steps to scale an application in IBM Bluemix PaaS**

1. Vertical scaling by increasing resources to an application instance
   Vertical scaling increases the resources available to an application by adding capacity directly to the individual nodes — for example, adding additional memory or increasing the number of CPU cores.

2. Horizontal scaling by increasing the number of application instance
   Horizontal scaling is often referred to as scaling out. The overall application resource capacity grows through the addition of entire nodes. Each additional node adds equivalent capacity, such as the same amount of memory and the same CPU. Horizontal scaling typically is achievable without downtime.

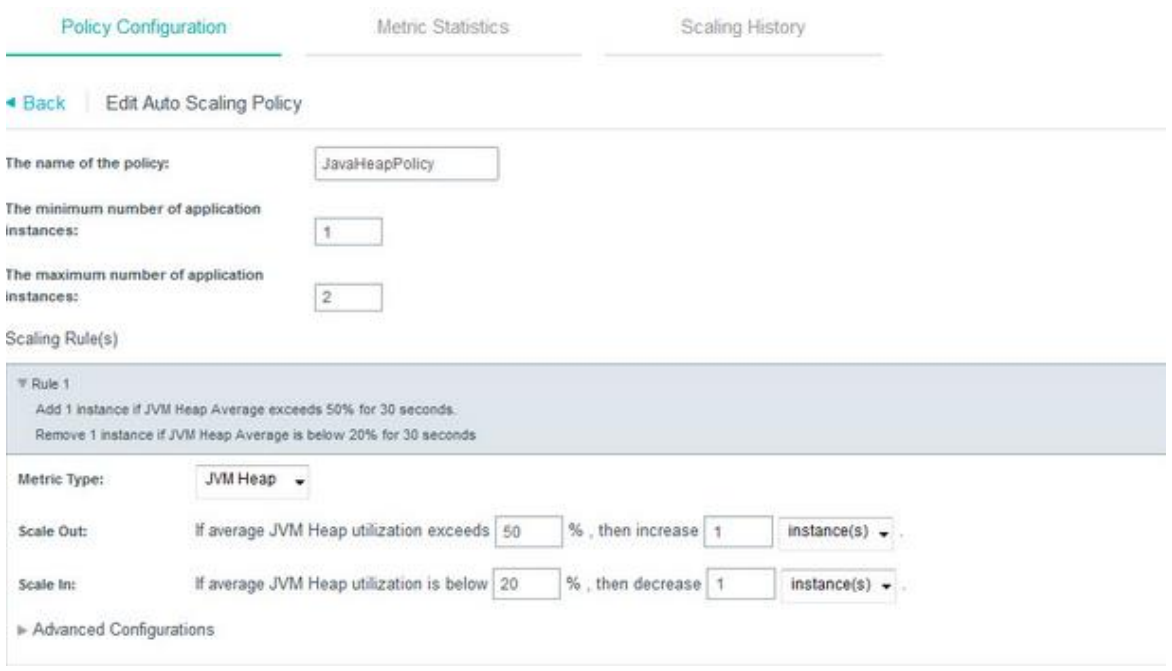
3. Understand how to manually scale applications through IBM Bluemix PaaS dashboard

The IBM Bluemix PaaS UI Dashboard supports both vertical and horizontal scaling through increasing the amount of memory and increasing the number of instances of an application runtime. Both techniques can be applied to the same application:



4. Automatically scaling applications in IBM Bluemix PaaS using the Auto-Scaling service and scaling policy fields and options such as: available metric types for runtimes, breach duration, and cooldown period.

   The Auto-Scaling service has control panels to define scaling policy, view metrics, and view scaling history. A scaling policy is based on thresholds for various metrics such as Memory, JVM Heap, Throughput, and Response time. The breach duration in the policy defines how long a threshold may be exceeded before a scaling event is performed. The cooldown period in the policy is how long to wait after a scaling event before monitoring for exceeded thresholds.



The name of the policy: JavaHeapPolicy

The minimum number of application instances: 1

The maximum number of application instances: 2

Scaling Rule(s)

▾ Rule 1

Add 1 instance if JVM Heap Average exceeds 50% for 30 seconds.

Remove 1 instance if JVM Heap Average is below 20% for 30 seconds

Metric Type: JVM Heap

Scale Out: If average JVM Heap utilization exceeds 50 % , then increase 1 instance(s) .

Scale In: If average JVM Heap utilization is below 20 % , then decrease 1 instance(s) .

Reference: https://www.ng.bluemix.net/docs/#services/Auto-Scaling/index.html#autoscaling

c. **Debug a Cloud application using development mode of IBM Bluemix PaaS**

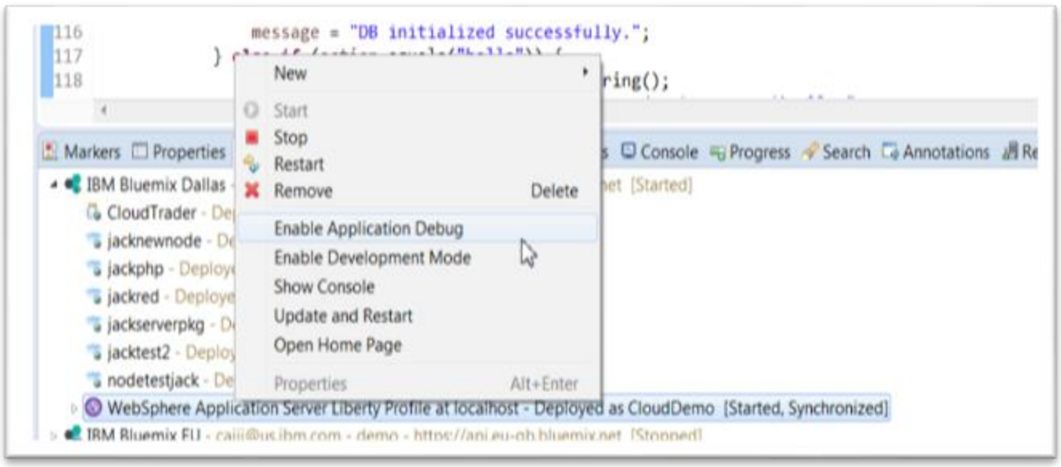
Development mode is a special mode available in Bluemix. It allows the application developer to conduct various operations so that errors can be found and resolved within the application.

1. Using the Eclipse Tools for Bluemix plug-in for development mode with IBM Liberty for Java buildpack applications

   Eclipse Tools supports both development mode and debugging mode. When development mode is activated, it is possible to incrementally publish application changes to Bluemix without redeploying the app.

   When Enable Application Debug is selected, development mode is automatically enabled. When debugging is enabled the developer may create remote debug sessions with the application instance.

   Activating debugging is performed in Eclipse by right-click on the application name under the correct Bluemix server, and selecting Enable Application Debug:

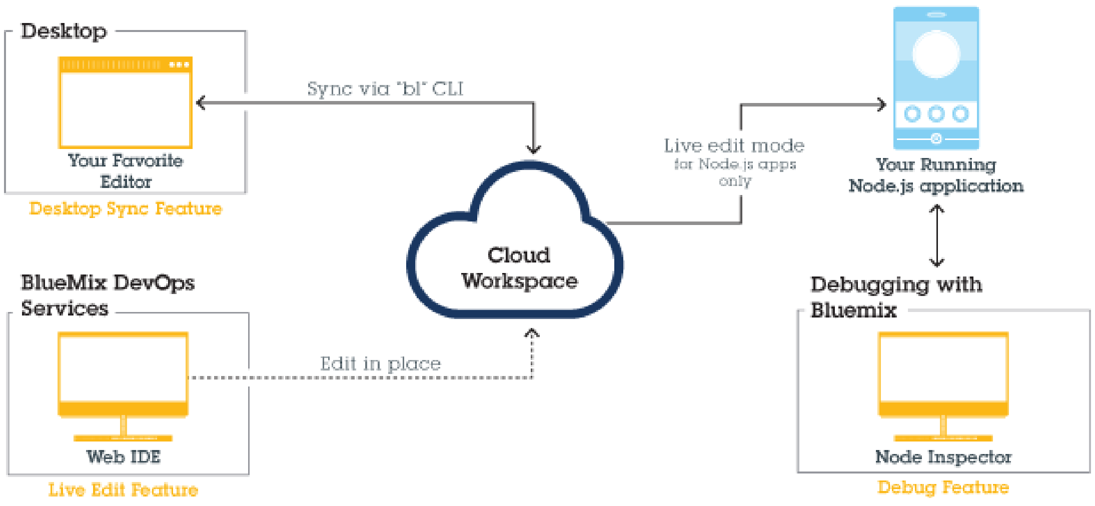


```
message = "DB initialized successfully.";
```

2. Using the Bluemix Live Sync debug feature for development mode with IBM Node.js buildpack applications

   The Bluemix Live Sync feature enables a debug mode that may be used to dynamically edit code, insert breakpoints, step through code, and restart the application, all while the application is running on Bluemix.

   This debug mode may be combined with Bluemix DevOps services Live Edit feature or the Desktop Sync feature to allow a Node.js application source code to be modified during the debug process, without requiring a redeploy of the application.



Requirements for using development mode with debug for Node.js:

- The application must use the IBM SDK for Node.js and not a custom buildpack
- The Chrome browser is required for the node inspector

Enabling Bluemix Live Debug:

a. Allow the buildpack to detect the app start command. The start command must be auto-detected by the buildpack, not set in the manifest.yml file.

- Ensure that the package.json file contains a start script that includes a start command for the app.
- If the app manifest.yml file contains a command, set it to null.

b. Set development mode on in the app manifest.yml file by adding this variable

```
env:
  ENABLE_BLUEMIX_DEV_MODE: "true"
```

c. Increase the memory entry in manifest.yml by adding 128M or more to the memory attribute.
d. Repush the application for the manifest.yml changes to take effect
e. Once the application has been pushed access:

https://[appname].mybluemix.net/bluemix-debug/manage

(where appname is the correct application name) to access the Bluemix debug user interface. From this interface, it is possible to restart the application, connect to a shell on the application, or launch the node inspector to debug the application.

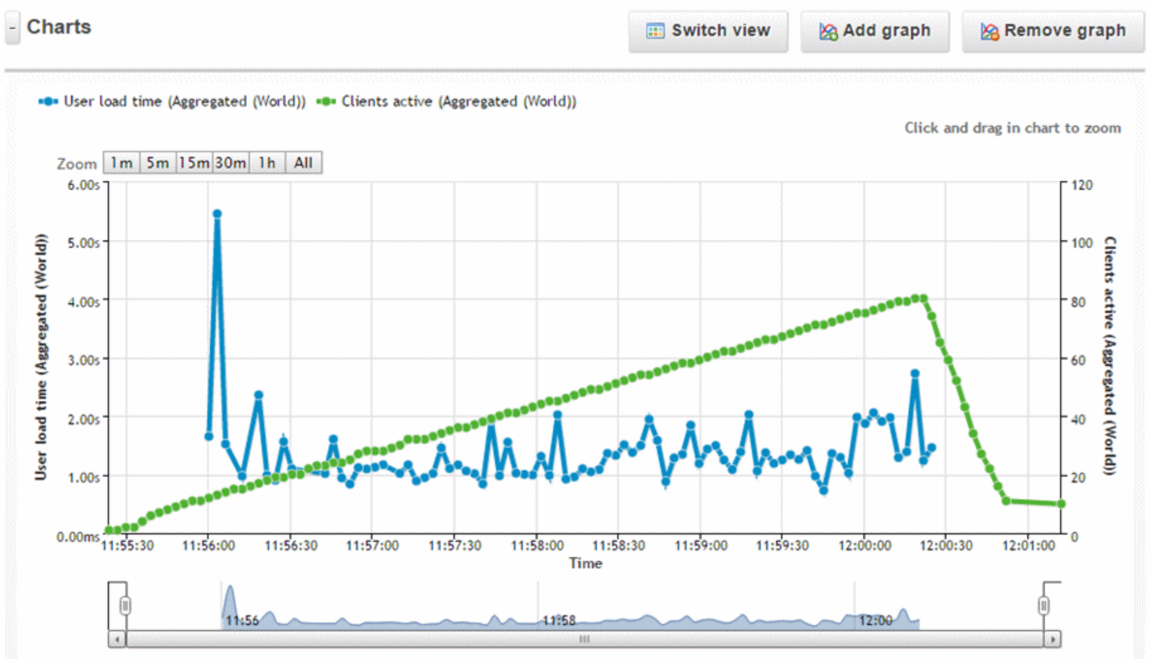
d. **Perform load testing on Cloud applications using simulated loads and describe the benefits of load testing**

1. Use Load Impact or Blazemeter 3rd party load testing services in IBM Bluemix PaaS
   IBM Bluemix PaaS has services to assist in characterizing how an applications responds under a simulated user load. These tests provide insights on performance and can also show if horizontal scaling is responding when the application is under stress.

2. Creating user scenarios
   A virtual user scenario corresponds to a set of actions within the application and can be captured interactively or defined in a script language of the specific load testing tool. These should be based on a typical use-case of a user accessing the application.

3. Defining virtual user load for a test
   The virtual user load run against an application can be focused on investigating different aspects of performance.
   - Measure response time for the application for a specific number of active users. For this type of test, the user workload is ramped to a steady state level and then held for a period time to gather response time statistics.
   - Determine peak scalability of an application. In this test the workload is increased in steps and held constant, or a series of test are performed each with a higher number of simulated users. A key metric such as the performance of a page providing user login or catalog display is monitored to ensure it does not exceed required levels.

4. Analyze results from load tests
   When a load test completes, tools provide graphical and tabular output of information to review. Graphs from load testing will show observed metrics like response time and number of simulated users, graphed as a function of the time into the load test:



Green dots show the virtual user (VU) load, and blue dots the duration of time for a user scenario to complete. In the example shown, there is no strong correlation between the response time and the number of active users. This can be interpreted as the application response time not showing sensitivity to the quantity of simulated users for the duration of the test.

When reviewing results, it is critical to verify that all application responses are successful and not showing error codes. An application generating HTTP 404 or 500 errors may appear to show a quick response time, but it is not operating correctly.

e. **Explain various methods to monitor an application in IBM Bluemix PaaS**

1. Measure application availability, CPU and heap memory usage, response time and throughput by using the Monitoring and Analytics service.

   In the Free plan for Monitoring and Analytics service, the Availability tab provides a view of basic uptime and response time from application pings. The Performance Monitoring tab provides Java Liberty and Node.js applications with views of the historical performance over configurable intervals for CPU usage and memory usage. Additionally, for Java Liberty applications, the thread pool usage and garbage collection statistics are available. For Node.js the throughput (measured in requests per minute) and response time are available. More detailed information for both runtimes is available through the Diagnostics plan through the Request Summary dashboard.

   Reference:
   https://console.ng.bluemix.net/docs/services/monana/index.html#gettingstartedtemplate

2. Monitoring application logs using the cf tool during staging and when the application is running.

   When an application is staging, the `cf push` command streams output from the staging process to the terminal. Simultaneously, the `cf logs` command can be used to display output from the Loggregator. When used with the `--recent` option, the `cf logs` command will display a subset of the past events for the application. To stream log output from the current time forward, omit the `--recent` option. The cf logs command may also be used while the application is running to either view the recent history of log entries or monitor the current log event stream.

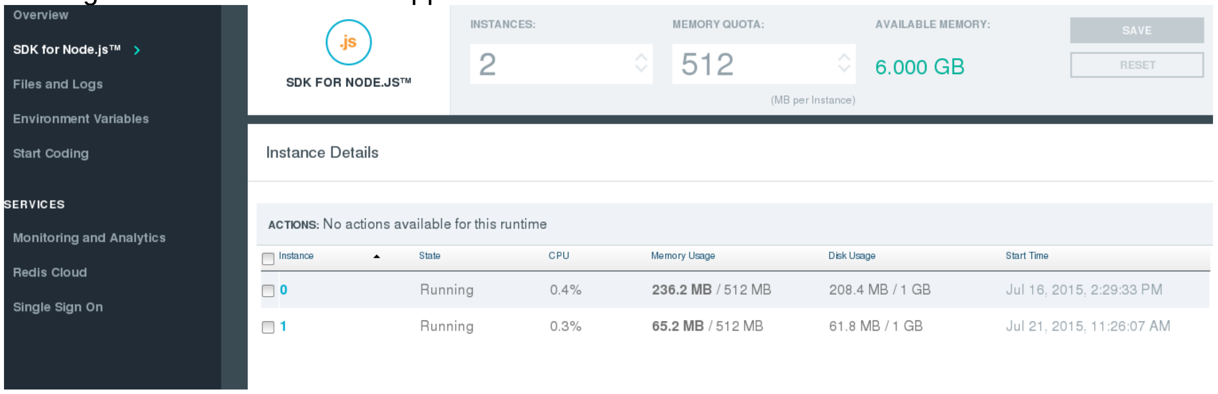
3. Viewing metrics of resource utilization with IBM Auto-Scaling service.

   If an application is using the Auto-Scaling service, there is a metrics tab available from the service that shows the recent history and current value of metrics that are used in scaling policies. The specific metrics available are dependent on the runtime used by the application.

   Reference: https://www.ng.bluemix.net/docs/services/Auto-Scaling/index.html

4. Using instance details panel from the application overview in the dashboard.

   The Instance Details panel provides the current view of running instances for an application, and the CPU and memory utilization of each instance. This panel can be accessed from the Cloud Foundry dashboard by selecting an application to bring up the overview, and then clicking on the runtime for the application:



INSTANCES: 2

MEMORY QUOTA: 512 (MB per Instance)

AVAILABLE MEMORY: 6.000 GB

Instance Details

ACTIONS: No actions available for this runtime

Instance	State	CPU	Memory Usage	Disk Usage	Start Time
0	Running	0.4%	**236.2 MB** / 512 MB	208.4 MB / 1 GB	Jul 16, 2015, 2:29:33 PM
1	Running	0.3%	**65.2 MB** / 512 MB	61.8 MB / 1 GB	Jul 21, 2015, 11:26:07 AM

Section 4: Enhancing Cloud Applications using Managed Services

a. Improve performance and scalabilty of IBM Bluemix PaaS applications with caching

1. Using Data Cache service to store application data
   IBM Data Cache supports distributed caching scenarios for web and mobile applications. Data Cache provides a NoSQL style in-memory data grid for storing key-value objects. It provides linear scalability, predictable performance and fault-tolerance through replication. It supports create, read, update, delete operations on entries in the map through Java and REST APIs.

   As the application runs, statistics are collected, and can be viewed using the Data Cache Dashboard in Bluemix via the **Data Cache Monitoring** capabilities which include visualizing the capacity of the cache, transaction throughput, transaction time, cache hit ratio, etc.

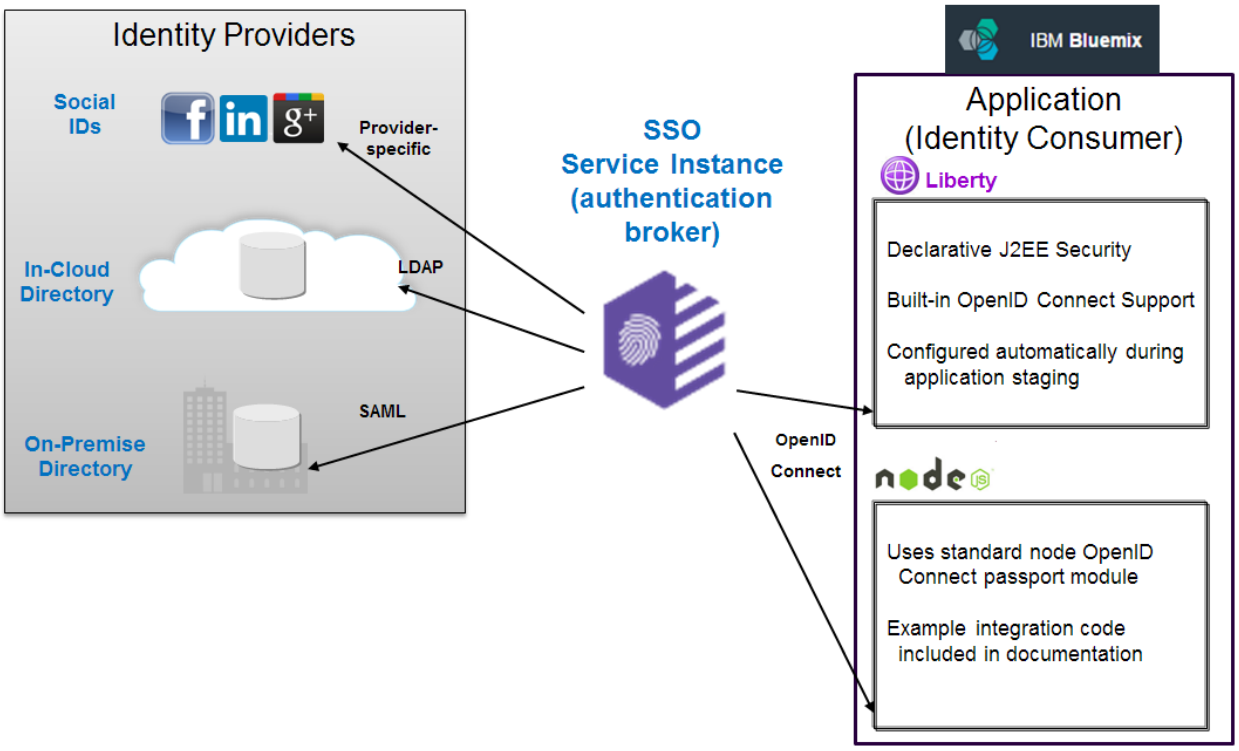
2. Using Session Cache to store and persist HTTP session objects
IBM Session Cache provides a distributed session cache for HTTP sessions. No code changes are requred for applications using J2EE standard HTTP sessions. Session data is replicated to provide fault-tolerance and persistence of session data.

References:
https://www.ng.bluemix.net/docs/services/DataCache/index.html
https://www.ng.bluemix.net/docs/services/SessionCache/index.html

b. Understand how to configure external authentication for IBM Bluemix PaaS web applications with the Single Sign On service (SSO)

1. SSO requires the application to use an OpenID Connect client interface

IBM Single Sign On for Bluemix is an authentication service that provides an easy to embed single sign on capability for web applications. The service may be bound to multiple Bluemix applications to provide a common authentication service. Applications call the SSO service through an OpenID Connect compatible client implementation.


Identity Providers
Social IDs
Provider-specific
In-Cloud Directory
LDAP
On-Premise Directory
SAML
SSO Service Instance (authentication broker)
OpenID Connect
IBM Bluemix
Application (Identity Consumer)
Liberty
Declarative J2EE Security
Built-in OpenID Connect Support
Configured automatically during application staging
node
Uses standard node OpenID Connect passport module
Example integration code included in documentation

2. Applications using SSO can support Cloud directories, Social Media sites and Enterprise directory as identity sources

The SSO service acts as an authentication broker for multiple identity sources. Identity Sources are collections of users, each collection is identified as a realm. The supported identity services are the following:

- Cloud Directory: this is a basic LDAP in the cloud that can be populated with simple username/password authentication credentials and a few other user attributes.
- Social providers: currently supporting Facebook, LinkedIn, and Google. These very commonly used identity providers allow your application to authenticate users and obtain identity information including an email address.
- Enterprise directory identity repositories: this integration uses SAML post single sign on. The on-premise website authenticates users (acting as the identity provider) and then uses SAML to securely transmit that identity information to the SSO Service instance, which is acts in the role of a SAML service provider. A virtual appliance is available to implement an authentication portal to an LDAP server if one is not already configured in the enterprise.

3. Integration requires the implementation of an authentication callback

   When adding the SSO service to an application, only a few steps are required. At a high level, the developer performs the following actions:
   - Add the Single Sign On service to the dashboard
   - Select the identity source(s) to configure
   - Configure settings for identity source
   - Bind SSO service to application and access integrate tab to download Node.js module (if using Node.js)
   - Insert integration code into application (implementing callback method URL)
     - Node.js and Java samples provided, others use an OpenID Connect compatible client library
   - Provide authentication callback URL and specify one or more configured identity sources for the application to use through the service integrate tab

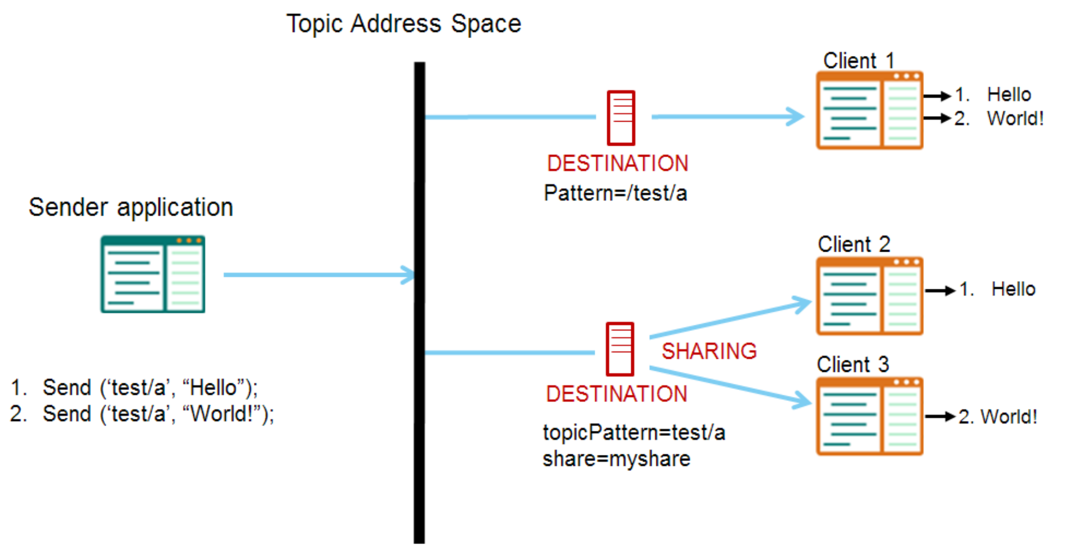
Reference: https://www.ng.bluemix.net/docs/services/SingleSignOn/index.html

c. Enable loosely coupled integration for IBM Bluemix PaaS applications and components by using Messaging Services

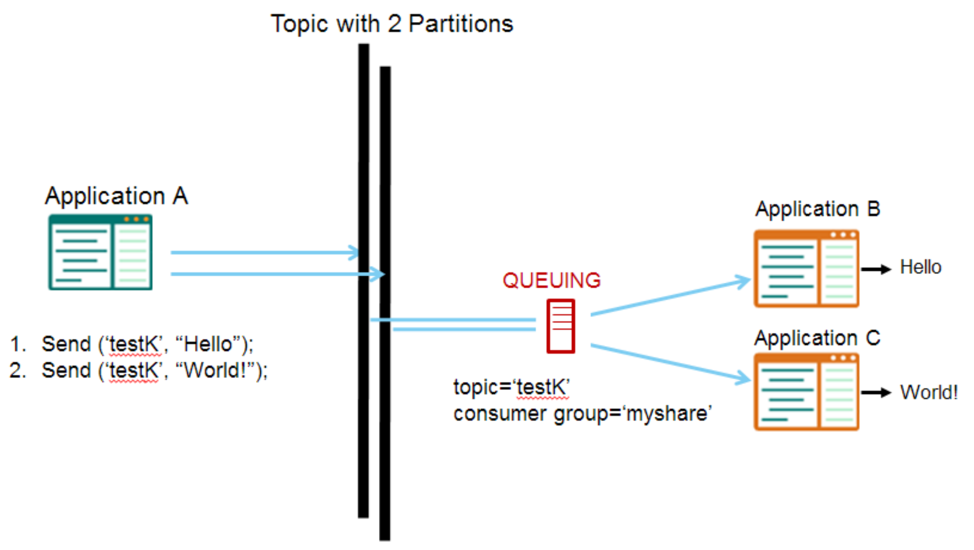
1. Understand messaging use-cases and available APIs in the Message Hub service
   Messaging services provide loose coupling between components of an application in several use-cases. A very common case is asyncronous worker offload of complex tasks allowing the processes handling these tasks to be scaled independently. Messaging provides a natural event-driven service model and avoids polling inefficiencies. Messaging provides a way to delay processing, for example to run a report at a specific time. Messaging can provide responsiveness in an application when integrating with 3rd party or external services by queueing requests. Across many of these use-cases, components using messaging services can be deployed in distributed locations to create hybrid cloud scenarios.

   The IBM Message Hub service is based upon Apache Kafka and supports the use of multiple APIs for messaging. The Kafka API, Kafka REST API and MQ Light APIs may all be used with this service. MQ Light provides a higher level of abstraction than the Kafka API and enables apps to be written quickly and portably in a unified messaging model.

2. Explain how to configure publish-subscribe and worker offload queue topologies using Message Hub
In a publish-subscribe model every message receiver (or consumer) gets a copy of messages published by the sender (or producer). In a queue model, messages are distributed among a pool of receivers (or consumers). When using the MQ Light API to create a queue model, each receiver will join a destination with sharing enabled.



When using the Kafka API (or Kafka REST API), to configure a queue, each consumer should subscribe using the same consumer group and the number of partitions on the topic should be greater than or equal to the maximum number of consumers.



3. Explain rationale of the cf option `–no-route` when using the worker offload pattern

An application component implementing the worker offload pattern through messaging services should not be configured with an application route in an envrionment like IBM Bluemix PaaS. To avoid this, you may use `cf push` with the `–no-route` option to prevent the Cloud Foundry envrionment from creating a route on application startup.

4. Understand benefits and usage of MQ Light API for topic hierarchies, fault tolerance, and QoS
   To support a flexible subscription model, the MQ Light API supports the use of hierarchy and wildcards in the topic structure. Levels in a topic are separated by the / character and two different wildcard types are available. The # wildcard matches any number of levels within a topic and the + character matches at a single level.

   MQ Light buffers messages based upon message and destination time-to-live. When an application subscribes to a topic it may specify a destination time-to-live which will cause MQ Light to store messages at the destination in the event of a receiving application crash. After the destination time-to-live expires, MQ Light will discard all held messages and stops storing new messages. For QoS, MQ Light messaging provides two models for delivery assurance, **at most once**, and **at least once**. The first is appropriate when occasional loss of data caused by a communication disruption is acceptable, and the second is required when there can be no loss of data. When using **at least once,** duplicate messages may appear around communication disruptions and they should to be programmatically discarded.

Reference: https://console.ng.bluemix.net/docs/services/MessageHub/index.html#messagehub
https://developer.ibm.com/messaging/mq-light/docs/api-reference/

d. Describe cognitive capabilities to process unstructured data and images in IBM Bluemix PaaS

1. Alchemy Language API services include sentiment analysis, entity extraction, relationship extraction, concept tagging, text extraction, language extraction, and microformat parsing

   Alchemy Language API's text analysis functions include:

   - **Entity extraction** - identify the proper nouns, i.e. people, companies, locations, etc.
   - **Sentiment analysis** - determine the overall sentiment or sentiment specific to a keyword or entity.
   - **Keyword extraction** - extract the important terms.
   - **Concept tagging** - identify the overall concepts of the text.
   - **Relation extraction** - extract subject-action-object relations.
   - **Taxonomy Classification** - automatically categorize your text, HTML or web-based content into a hierarchical taxonomy.
   - **Author extraction** - identify the author of a blog post or news article.
   - **Language detection** - detect 97+ languages.
   - **Text extraction** - pull out only the import content from a web page.
   - **Feed detection** - extract the ATOM or RSS feeds from a web page.
   - **Microformats Parsing** - Automatically detect and parse the microformats embedded within a webpage.

- **Linked Data Support** - AlchemyAPI supports Linked Data in the web thereby exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF

For example the Taxonomy Classification API service provides an easy way to categorize a public web page, uploaded text or uploaded html. To perform taxonomy classification of a public web page use:

API endpoints: URLGetRankedTaxonomy
Output: detected language and topic categories identified in data
Parameters: (apikey, text, url, outputMode, callback, baseURL)
Response: (status, url, detected category, score, statusInfo)

Using the endpoint http://access.alchemyapi.com/calls/url/URLGetRankedTaxonomy and providing the url: http://www.nytimes.com/2013/07/13/us/politics/a-day-of-friction-notable-even-for-a-fractious-congress.html?_r=1

The service returns a response object of the form (when the outputMode is specified as json):

```
{"status": "OK","usage": "By accessing AlchemyAPI or using information
generated by AlchemyAPI, you are agreeing to be bound by the AlchemyAPI
Terms of Use: http://www.alchemyapi.com/company/terms.html","url":
"http://www.nytimes.com/2013/07/13/us/politics/a-day-of-friction-
notable-even-for-a-fractious-congress.html?_r=2", "totalTransactions":
"1",","language": "english","taxonomy": [{"label": "/law, govt and
politics/politics","score": "0.769694" },{"label": "/law, govt and
politics/government","score": "0.613808" },{"label": "/law, govt and
politics/legalissues/legislation","score": "0.512705"}]}
```

The API endpoints for uploading text or html are TextGetRankedTaxonomy, HTMLGetRankedTaxonomy and provide similar taxonomy classification services.

Another popular AlchemyAPI is SentimentAnalysis that provides an easy way to identify positive/negative sentiment within any document or web page. You can compute document-level sentiment, user-specified sentiment, entity-level and keyword-level sentiments. To perform sentiment analysis of a public web page use:

API endpoints: URLGetTextSentiment
Output: Identifies the attitude, opinion and feeling towards a person/organization/product/location etc.
Parameters: (apikey, text, url, outputMode, callback, showSourceText)
Response: (status, url, detected category, score, statusInfo)

Using the endpoint http://access.alchemyapi.com/calls/url/URLGetTextSentiment and providing the url: http://www.politico.com/

The service returns a response object of the form:

```
URL sent to URLGetTextSentiment AlchemyAPI --> http://www.politico.com/
<?xml version="1.0" encoding="UTF-8"?><results>
```

```
<status>OK</status>
<usage>By accessing AlchemyAPI or using information generated by AlchemyAPI,
you are agreeing to be bound by the AlchemyAPI Terms of Use:
http://www.alchemyapi.com/company/terms.html</usage>
<url>http://www.politico.com/</url>
<totalTransactions>1</totalTransactions>
<language>english</language>
<docSentiment>
<mixed>1</mixed>
<score>-0.437706</score>
<type>negative</type>
</docSentiment></results>
```

The API endpoints for uploading text or html are TextGetTextSentiment, HTMLGetTextSentiment and provide similar sentiment analysis services.

2. Alchemy Vision API services include imaging tagging, link extraction and face detection/recognition

   Alchemy Vision function includes the following:

   - **Image Link Extraction** - perform image link extraction on Internet-accessible URLs and posted HTML files.
   - **Image Tagging** - perform image tagging on your Internet-accessible URLs and posted image files.
   - **Face Recognition** - perform face detection and recognition on your Internet-accessible URLs and posted image files.

   For example, the Image Tagging API provides an easy way to scan a provided URL and find the most prominent image which can be tagged. It also provides an endpoint to provide image tagging on uploaded images. To extract image keywords from an URL use:

   API endpoints: URLGetRankedImageKeywords
   Output: find the most prominent image, classify and tag it
   Parameters: (apikey, text, url, outputMode, callback, baseURL)
   Response: (status, url, detected category, score, statusInfo)

   Using the endpoint
   http://access.alchemyapi.com/calls/url/URLGetRankedImageKeywords and providing the http://farm4.staticflickr.com/3726/11043305726_fdcb7785ec_m.jpg

   The service returns a response object of the form:

```
URL sent to URLGetRankedImageKeywords AlchemyAPI -->
http://farm4.staticflickr.com/3726/11043305726_fdcb7785ec_m.jpg
<?xml version="1.0" encoding="UTF-8"?><results>
<status>OK</status>
```

```
<usage>By accessing AlchemyAPI or using information generated by AlchemyAPI,
you are agreeing to be bound by the AlchemyAPI Terms of Use:
http://www.alchemyapi.com/company/terms.html</usage>
<url>http://farm4.staticflickr.com/3726/11043305726_fdcb7785ec_m.jpg</url>
<totalTransactions>4</totalTransactions>
<imageKeywords>
   <keyword>
      <text>cat</text>
      <score>0.998887</score>
    </keyword>
    <keyword>
      <text>animal</text>
      <score>0.768525</score>
    </keyword>
    <keyword>
       <text>pet</text>
       <score>0.574443</score>
    </keyword>
 </imageKeywords></results>
```

The API endpoint for uploading an image is ImageGetRankedImageKeywords, and provides similar image keyword tagging.

Reference: http://www.alchemyapi.com/api

e. Understand how to store and retrieve files using the IBM Object Storage service in Bluemix

1. Creation of a container in object storage service

   Note that IBM Bluemix PaaS offers two types of Object Storage services. Object Storage version 1 is based on SoftLayer object storage and provides a service that can be bound to a Bluemix application and is also used in some boilerplates. Object Storage version 2 provides a flexible model to use Object Storage on IBM on-premise clouds or IBM public cloud. This section focuses on usage specific to Object Storage version 1

   Before using any API services of Object Storage version 1, an authentication token must be obtained by calling the api_uri endpoint using Basic Authentication using credentials provided in VCAP_SERVICES to the application. Once obtained, containers and objects can be managed through REST API calls to the storage url returned with the token. For example, a successful authentication will have a response of the form:

```
HTTP/1.1 200 OK
X-Backside-Transport: OK OK
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: application/json:charset=utf-8
Date: Fri, 30 Jan 2015 18:33:12 GMT
Set-Cookie:
connect.sid=s%3AnPo52qjvdynxlvCGEAmJfs7d.HuYPqYVmtQqeImm73i
Unubk2r9T0S5WIQVOL5edX08U; Path=/; HttpOnly
--> X-Auth-Token: AUTH_tk924cd97c2af7475c8e6cb25d2adaccf8
```

```
X-Cf-Requestid: 969d6f00-8b85-471e-5ab5-80c36031f9fb
X-Powered-By: Express
--> X-Storage-Url:
https://dal05.objectstorage.softlayer.net/v1/AUTH_0c6bf5d5-
b5e4-43aa-93f1-ea7065837fb8
X-Client-IP: 129.42.208.182
X-Global-Transaction-ID: 770597619
```

To create a container in Object Storage, use a PUT request to the storage url and append the new container name in the request. Using the example above:

```
curl -i -H "X-Auth-Token:
AUTH_tk924cd97c2af7475c8e6cb25d2adaccf8" -X PUT
https://dal05.objectstorage.softlayer.net/v1/AUTH_0c6bf5d5-
b5e4-43aa-93f1-ea7065837fb8/myNewContainer
```

2. Perform create, upload, and list operations on an object using containers

   Once the container is created, objects are manipulated by http requests. To create a file or upload with new data to an existing file, use the PUT command specifying the container in the path (simplified command omitting `-H` option shown):

   - `curl -X PUT [...] storageurl/container1 -T object.jpg`

   To list all of the objects in a container, use the GET command specifying the container name (simplified command shown):

   - `curl -X GET [...] storageurl/container1`

   To download an object, use the GET command specifying the object name (simplified command shown):

   - `curl -X PUT [...] storageurl/container1/object.jpg`

3. Purpose of metadata to store information about files in object storage

   For objects that are stored in object storage, information about the object is available through metadata. Object metadata includes some standard types like: last modified, length, md5 checksum, expiration date, and also supports custom data through the X-Object-Meta-{name} element. Reading metadata from an object is done using the HEAD request:

   - `curl -X HEAD [...] storageurl/container1`

   Object metadata is created or updated with the http POST command. Selected standard metadata are modifiable and X-Object-Meta-{name} items may be created as needed.

References:
https://www.ng.bluemix.net/docs/#services/ObjectStorage/index.html#ObjectStorage
http://developer.openstack.org/api-ref-objectstorage-v1.html

Section 5: Using DevOps Services & tools to Manage Cloud Applications

a. Describe capabilities of IBM Bluemix DevOps Services

1. Agile Planning: Planning and tracking features to manage collaborative work in agile teams
   - Teams can create stories, tasks, and defects to describe and track project work, and use agile planning tools to manage backlogs, plan releases, and plan sprints.
2. Web code editor: A browser-based Integrated Development Environment (IDE) for cloud development
   - Using the Web IDE, teams can import, create, modify and debug source code from a web browser. The environment also provides color coding and content assist capabilities to facilitate development activities.
3. Source control management: Parallel development and versioning features through Git, Jazz SCM, or GitHub
   - Each project gets a shared repository where team members check in changes, associate code changes with work items, and view a history of recent updates
4. Delivery pipeline: Continuous Integration and Continuous Delivery features to rapidly deploy cloud applications
   - The Delivery Pipeline allows automatically building and deploying applications to IBM's cloud platform.

b. Plan and track work for agile team collaboration

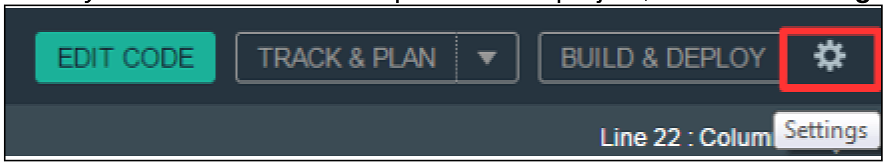
1. Bluemix DevOps Services - Track & Plan supports typical activities conducted during agile projects, such as backlog management, sprint planning, and daily scrums
2. Understand work items in Track & Plan like epic, story, task, and defect

   On an agile project, features to implement are captured in a product backlog through work items. Common work items in Track & Plan are epic, story, task and defect. The team identifies the work for the next Sprint in the Sprint Backlog. A set of hands-on steps are provided to show some key concepts in Track & Plan:

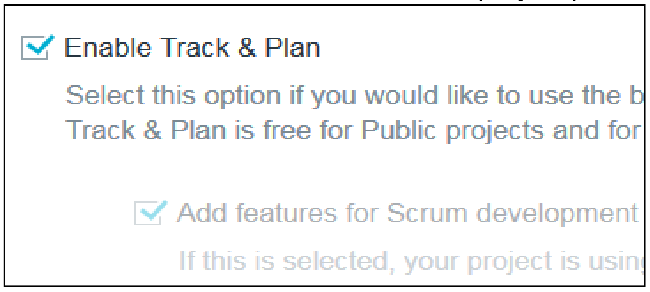
   a. Enable Agile Planning and Tracking on a project:

   Depending on how IBM Bluemix DevOps Services projects have been created, the Agile Planning and Tracking capability may not be enabled. First verify that agile planning and tracking is enabled for your project.

   - On your IBM Bluemix DevOps Services project, click the **Settings** button



   - On the left pane of the Settings page, click **OPTIONS**. Verify that the **Track & Plan** capability is enabled, and that the Scrum development feature has been added to the project. (if not, this is because the option was not selected when the project was created or forked from another project.):



☑ Enable Track & Plan

Select this option if you would like to use the b
Track & Plan is free for Public projects and for

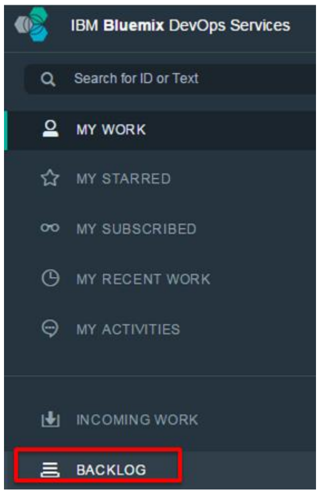
☑ Add features for Scrum development

If this is selected, your project is usin

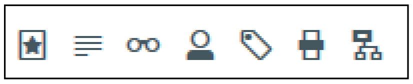
- When Agile Tracking and Planning is enabled, click the **TRACK & PLAN** button on the top of the screen. You are now ready to plan your work using the Scrum approach.

b. Prepare the Product Backlog
The product backlog is a prioritized features list containing short descriptions of all functionality desired in the product.

Add work items to the Backlog
- On the left pane of the Track & Plan page, select **Backlog.**



- At this point the backlog is empty and needs to be populated with work items Stories.
- Create the first story for your project. You can use the description of your choice or reuse the following example.
- In **Create a Work Item** field, enter: "As a user, I want be able to search for food suggestions on Twitter so that I benefit from others' experiences." Notice that you also have the option to add additional information to the work item, such as type, description, subscribers, owner, due date, tags, priority, or parent.



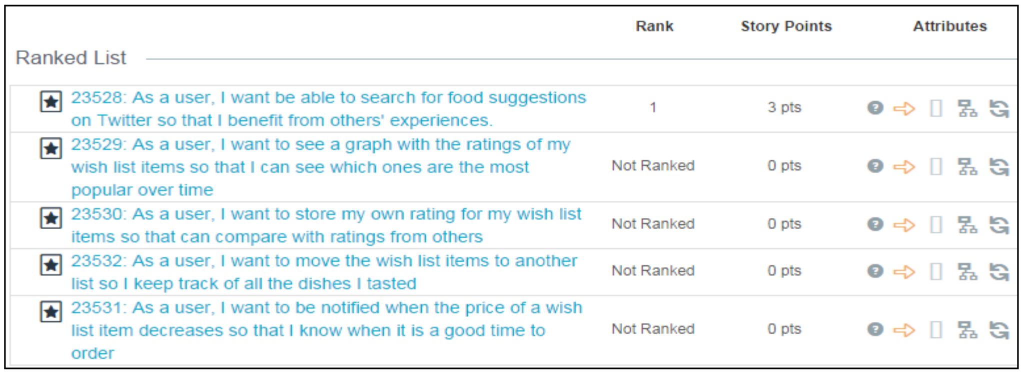
- Click **CREATE**. The work item is created. Its type is automatically set to "Story" as the tool recognizes the typical pattern used for stories (As a <type of user>, I want <some goal> so that <some reason>).

- Repeat similar steps to create four other stories (or enter the description of your choice):
    - As a user, I want to see a graph with the ratings of my wish list items so that I can see which ones are the most popular over time.
    - As a user, I want to store my own rating for my wish list items so that I can compare with ratings from others.
    - As a user, I want to be notified when the price of a wish list item decreases so that I know when it is a good time to order.
    - As a user, I want to move the wish list items to another list so I keep track of all the dishes I tasted.
- All work items are listed in the **Recently Created** section.

Define priorities and estimate effort

Move the most importtant stories to the top of the backlog (ranking), and define story points to represent the estimated effort to implement a story.

- Refresh your browser. The work items are no longer in the Recently Created section but in the **Ranked List** one.
- Move (drag) the first story that you created to the top of the backlog ("As a user, I want to be able to search for food suggestions on Twitter so that ..."). This story is now ranked as the first one.
- For this story, change the Story Points to **3 pts** (click 0 pts to select another value). It gives an indication of the effort needed to implement this story (see the image below)



Ranked List

	Rank	Story Points	Attributes
23528: As a user, I want be able to search for food suggestions on Twitter so that I benefit from others' experiences.	1	3 pts	
23529: As a user, I want to see a graph with the ratings of my wish list items so that I can see which ones are the most popular over time	Not Ranked	0 pts	
23530: As a user, I want to store my own rating for my wish list items so that can compare with ratings from others	Not Ranked	0 pts	
23532: As a user, I want to move the wish list items to another list so I keep track of all the dishes I tasted	Not Ranked	0 pts	
23531: As a user, I want to be notified when the price of a wish list item decreases so that I know when it is a good time to order	Not Ranked	0 pts	

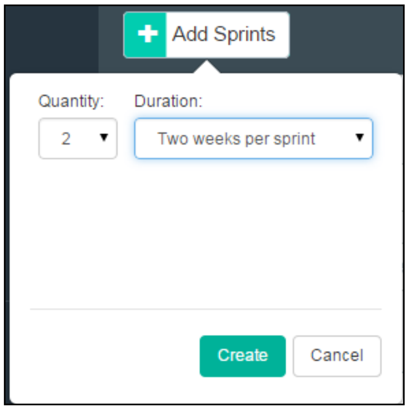
- Optionally, you can reorder the other items on the backlog and change their *Story Points* value. Now that you have an ordered backlog, you are ready to proceed with Sprint planning.

c. Define new Sprints for the project

Now that the Product Backlog is populated, it is time to define how many Sprints you will have in your project and the duration of these Sprints. This varies according to the project objectives and the release dates.

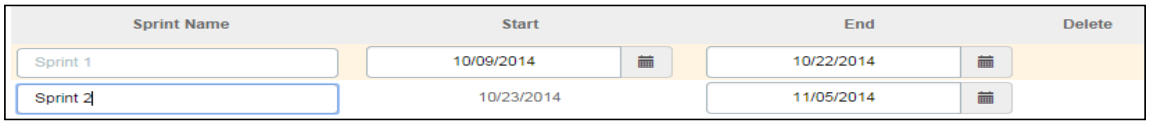
- On the left pane, click **Sprint Planning,** then click the **Add Sprints** button.

- Configure the sprints so you have at least two, and specify a duration of two weeks per sprint.



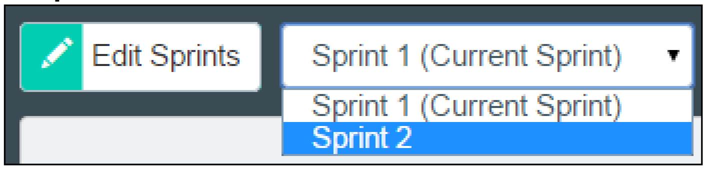
Quantity: 2
Duration: Two weeks per sprint

- Click **Create**. The sprints are created. At any point in time, you can click **Edit Sprints** to reconfigure your Sprints.



Sprint Name	Start	End	Delete
Sprint 1	10/09/2014	10/22/2014	
Sprint 2	10/23/2014	11/05/2014	

- On the left pane, click the **Sprint Planning** link.
- Select **Sprint 2** in the list box

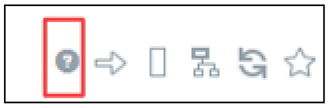


- The backlog and Sprint 2 are displayed on the same page.

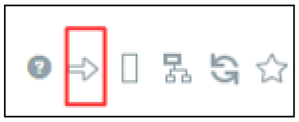
d. Assign some backlog items to the first Sprint
Select **Sprint 1 (Current Sprint)**. The backlog and Sprint 1 are displayed side by side to facilitate planning activities.

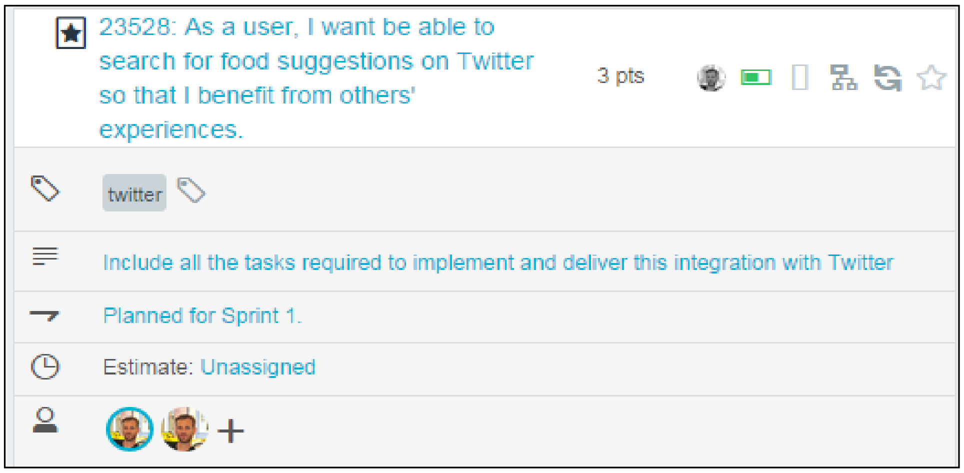
- To assign work to the first sprint, move (drag and drop) the first story from the backlog to Sprint 1. ("As a user, I want be able to search for food suggestions on Twitter so that ...").
- Note that on an Agile project, the team might decide to decompose stories into tasks and track the work at the task level. To simplify the scenario, you do not create child tasks in this lab.

- Change the owner for this story by assigning yourself.



- Change the status of this story to **Start Working**.





23528: As a user, I want be able to search for food suggestions on Twitter so that I benefit from others' experiences.

3 pts

twitter

Include all the tasks required to implement and deliver this integration with Twitter

Planned for Sprint 1.

Estimate: Unassigned

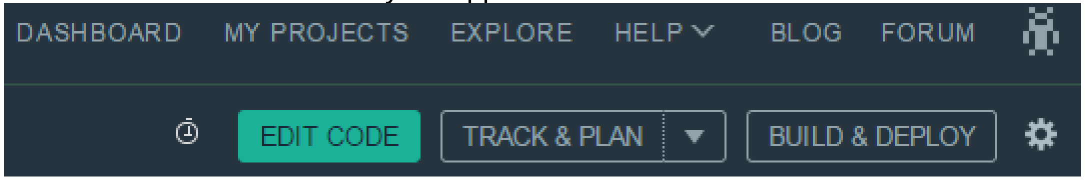
- You are ready to implement the story. When the story is completed, you change the status again to reflect progress on the project.

c. Edit and debug Cloud applications using IBM Bluemix DevOps Services Web code editor

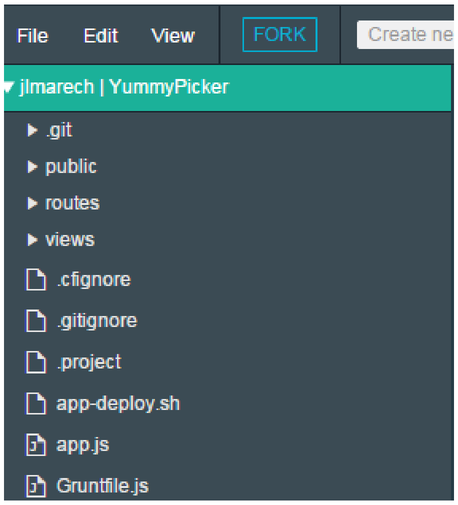
1. Understand basic functionality of the Web code editor

   A set of hands-on steps are provided to show some key concepts for the Web IDE.

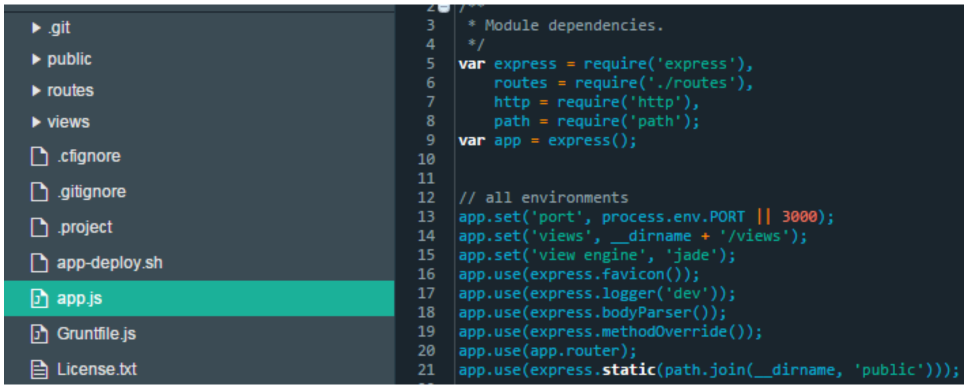
   Start Web Editor: In the IBM Bluemix DevOps Services project page, click **EDIT CODE** to access the source code of your application.



   The files for your application source code are displayed on the left pane of the screen. The structure of the code depends on your programming language.

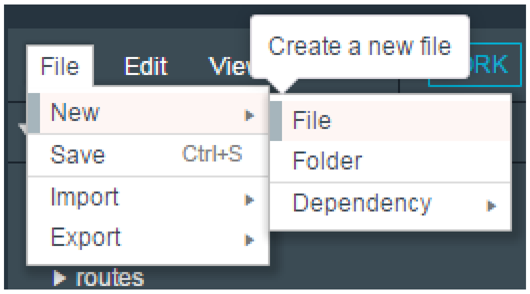


Edit existing code: To edit existing code, simply select the file from the left pane to open it. You can use the menu option (File --> Save) to save changes, but note that the web IDE also include an auto-save feature.

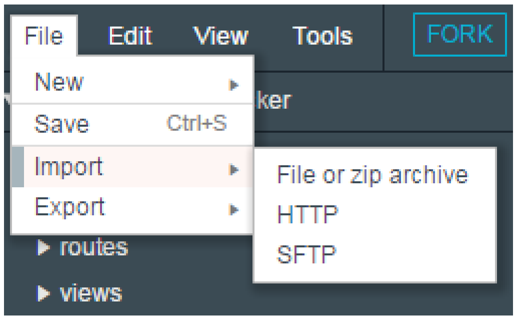


```
 * Module dependencies.
 */
var express = require('express'),
    routes = require('./routes'),
    http = require('http'),
    path = require('path');
var app = express();


// all environments
app.set('port', process.env.PORT || 3000);
app.set('views', __dirname + '/views');
app.set('view engine', 'jade');
app.use(express.favicon());
app.use(express.logger('dev'));
app.use(express.bodyParser());
app.use(express.methodOverride());
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));
```

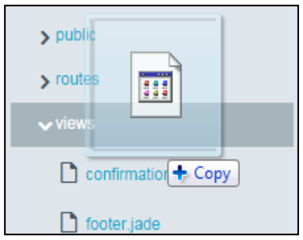
Create new file: To create new files in your project, select the **File > New > File** option. Then start adding code to this file. You also create folders to organize and group source files.



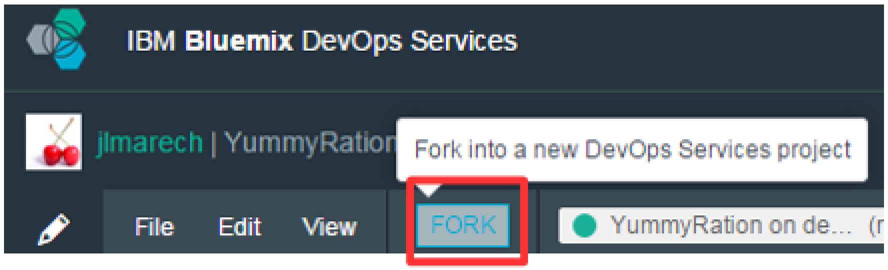
Import existing code: To import code from your local environment or from another project, click on **File -> Import** and select one of the import options from the cascading menu, upload file or zip archive, http or sftp transfer.



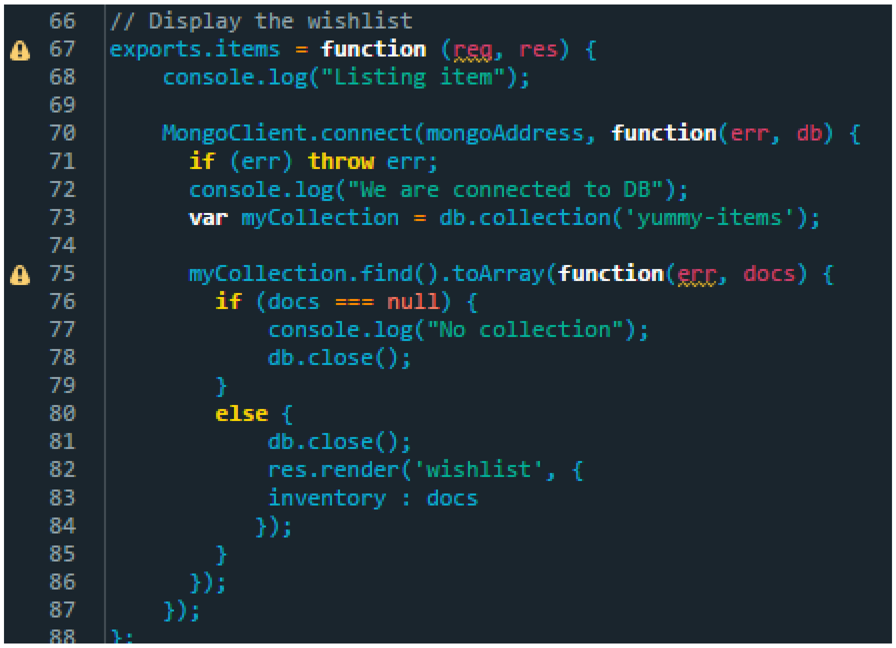
You can also drag and drop files from your local disk to the IBM Bluemix DevOps Services project:



Fork existing projects: Another powerful option, if you don’t want to start development from scratch, is to leverage the Fork option. You can browse public Bluemix DevOps Services projects and if you want to reuse one of them, click the Fork button in order to create a copy in your environment.


IBM Bluemix DevOps Services
jlmarech | YummyRation
Fork into a new DevOps Services project
File
Edit
View
FORK
YummyRation on de...

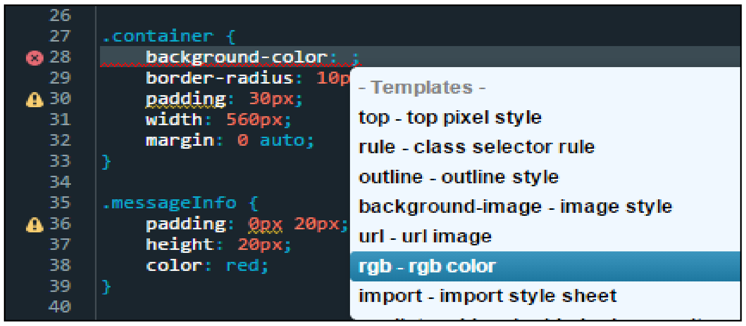
Code editor syntax highlighting: The IBM Bluemix DevOps Services web editor provides syntax coloring support for multiple languages, including HTML, CSS, JavaScript, Ruby, and Python.



```
// Display the wishlist
exports.items = function (req, res) {
    console.log("Listing item");

    MongoClient.connect(mongoAddress, function(err, db) {
      if (err) throw err;
      console.log("We are connected to DB");
      var myCollection = db.collection('yummy-items');

      myCollection.find().toArray(function(err, docs) {
        if (docs === null) {
            console.log("No collection");
            db.close();
        }
        else {
            db.close();
            res.render('wishlist', {
            inventory : docs
           });
        }
      });
    });
};
```

For some languages, such as JavaScript, the Web IDE also supports syntax checking and code completion, both for standard language constructs and for the services that Bluemix provides. First type **Ctrl-Space** to activate code assist, then you can choose code snippets from the provided list.

The Web IDE provides a control for modifying the Launch Configuration of the application in Bluemix. Changing the Launch Configuration allows the developer to specify a different application / hostname for the code when deployed and also to change the region for deployment.

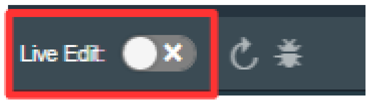


```
.container {
    background-color: ;
    border-radius: 10p
    padding: 30px;
    width: 560px;
    margin: 0 auto;
}

.messageInfo {
    padding: 0px 20px;
    height: 20px;
    color: red;
}
```

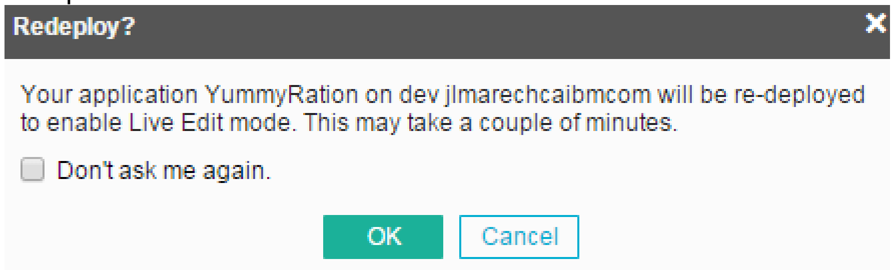
2. Using Live Edit to quickly make changes without redeploying an application

   For advanced debugging and quick changes to application code, IBM Bluemix DevOps Services provides specific support for Node.js applications (only for Node.js at this time)

   - The first step to use the debug tools is to enable the Live Edit capability. Simply click the Live Edit button on the project page



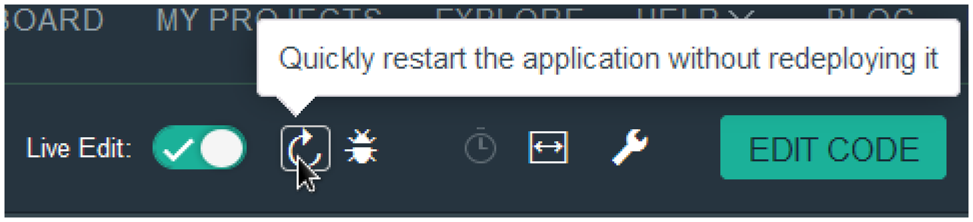
- When activating Live Edit, you are asked to re-deploy your application. Click OK to accept.


Redeploy?

Your application YummyRation on dev jlmarechcaibmcom will be re-deployed to enable Live Edit mode. This may take a couple of minutes.

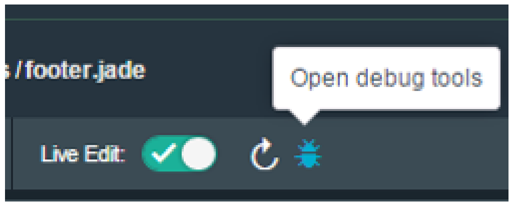
Don't ask me again.

- After the application redeploys, you can edit the application code in the web editor and push it into the running instance with a quick restart (instead of a full redeployment)

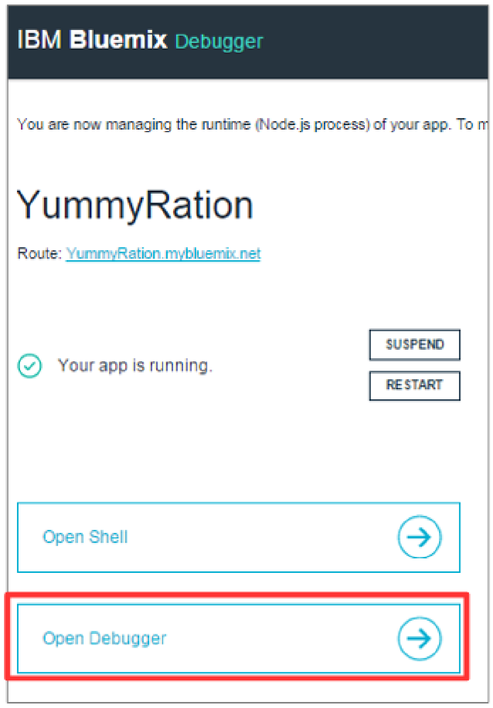


3. Enable debug mode to troubleshoot an application running in IBM Bluemix PaaS When in Live Edit mode you can also bring up an interactive debugger for Node.js applications. (Requires Chrome browser)

   - Click the debug button to enter the debugging page (you may be asked to log in again using your Bluemix credentials).



   - On the Bluemix Debugger page, click the **Open Debugger** button.



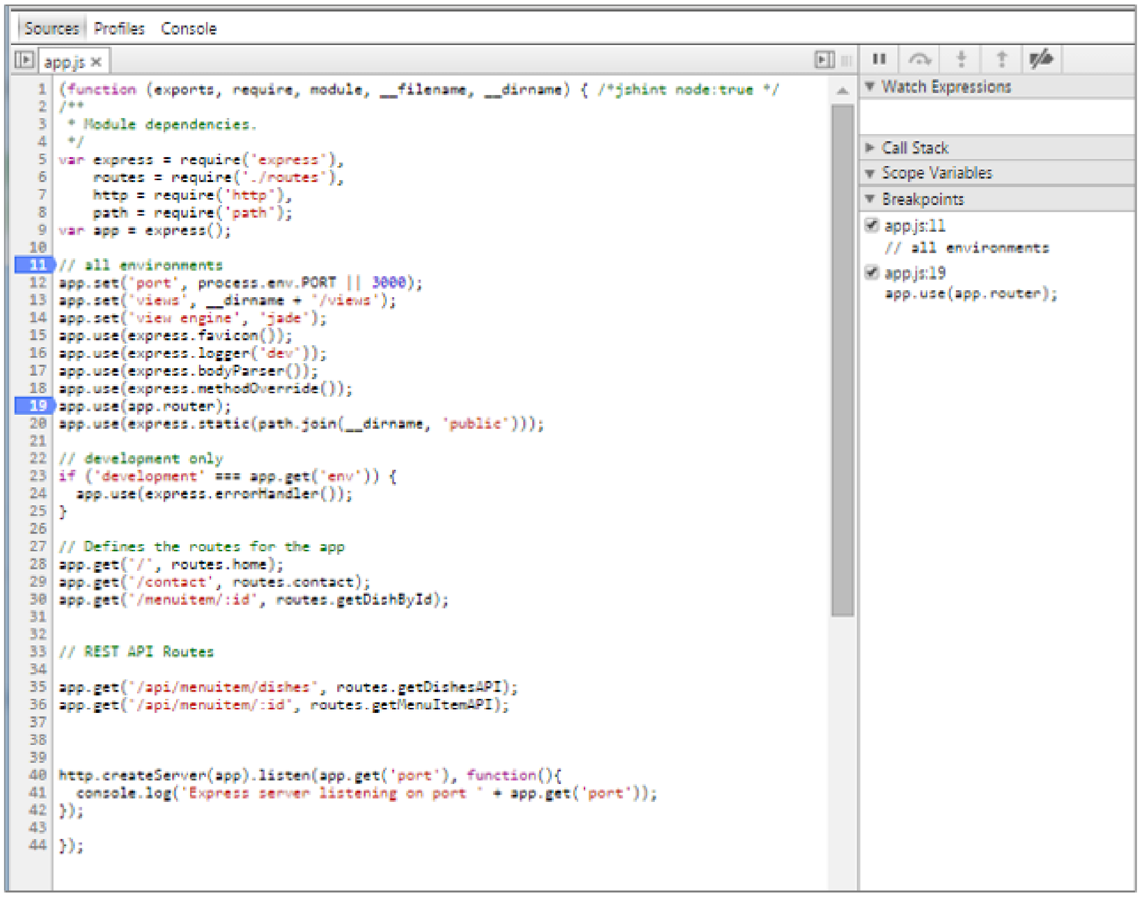
You are now managing the runtime (Node.js process) of your app. To m

YummyRation

Route: YummyRation.mybluemix.net

Your app is running.

- You now have access to an environment to set breakpoints, or inspect the call stack and variables.



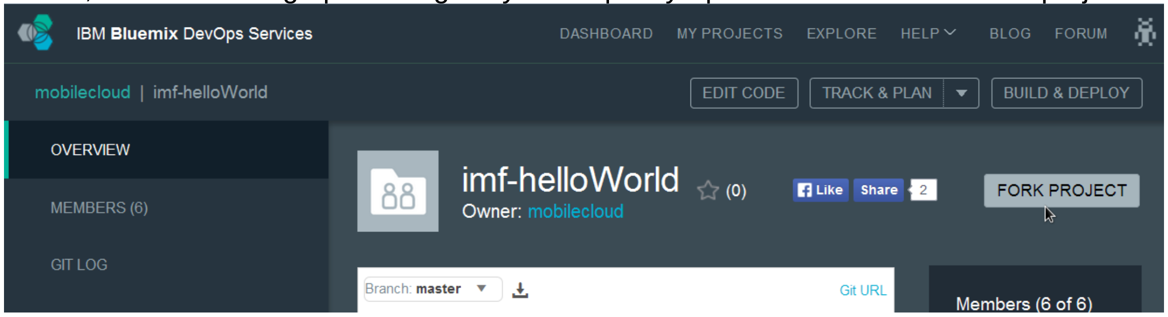
```
(function (exports, require, module, __filename, __dirname) { /*jshint node:true */
/**
 * Module dependencies.
 */
var express = require('express'),
    routes = require('./routes'),
    http = require('http'),
    path = require('path');
var app = express();

// all environments
app.set('port', process.env.PORT || 3000);
app.set('views', __dirname + '/views');
app.set('view engine', 'jade');
app.use(express.favicon());
app.use(express.logger('dev'));
app.use(express.bodyParser());
app.use(express.methodOverride());
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));

// development only
if ('development' === app.get('env')) {
  app.use(express.errorHandler());
}

// Defines the routes for the app
app.get('/', routes.home);
app.get('/contact', routes.contact);
app.get('/menuitem/:id', routes.getDishById);


// REST API Routes

app.get('/api/menuitem/dishes', routes.getDishesAPI);
app.get('/api/menuitem/:id', routes.getMenuItemAPI);



http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port ' + app.get('port'));
});

});
```

```
app.js:11
  // all environments
app.js:19
  app.use(app.router);
```

d. Understand capabilities of IBM Bluemix DevOps services source code management for projects

1. Using the fork option to copy an existing DevOps project into a new project for enhancement

   You may create a dedicated copy of a project in DevOps by selecting the Fork Project button, which will bring up a dialog for you to specify options and name the new project:



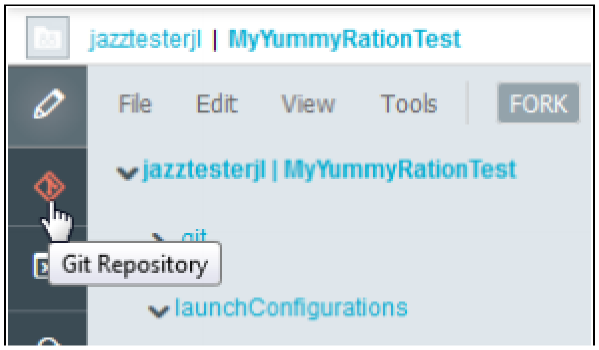
2. Understand the difference between a Commit and a Push and review and manage code pushes to the repository by project members

   IBM Bluemix DevOps Services supports parallel development so that teams can efficiently collaborate on the source code of a cloud application. Developers work in isolation (workspace) until they decide to share their code with the rest of the team. In order for source code modified or added to a cloud application to be available to the entire team, it must be delivered to the project repository.

   A set of hands-on steps are provided to show some key concepts of source code management.

   a. Commit workspace changes to your repository

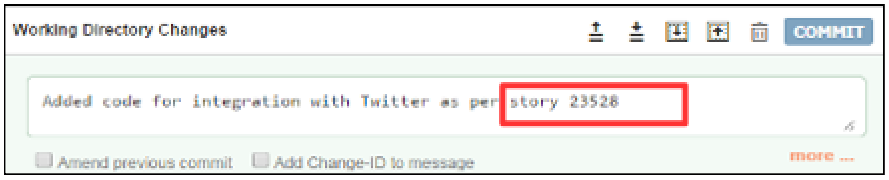
   - In IBM Bluemix DevOps Services, click **EDIT CODE**, then on the left pane, click the **Git Repository** icon.


jazztesterjl | MyYummyRationTest
File
Edit
View
Tools
FORK
jazztesterjl | MyYummyRationTest
git
Git Repository
launchConfigurations

   - In the Git Repository page, review the list of files in the **Working Directory**

Changes section. This assumes that you have modified some source code in your application before this step

- In the **Working Directory Changes** section, enter the following comment:
    - I added code for integration with Twitter
- Optionally, you can add a story number in your comment: "I added code for integration with Twitter as per story ####" and replace #### with **a real story number** from your project. Adding a story number in the comment will create a link between the change set and the story, permitting lifecycle traceability between code changes and work items.


Working Directory Changes

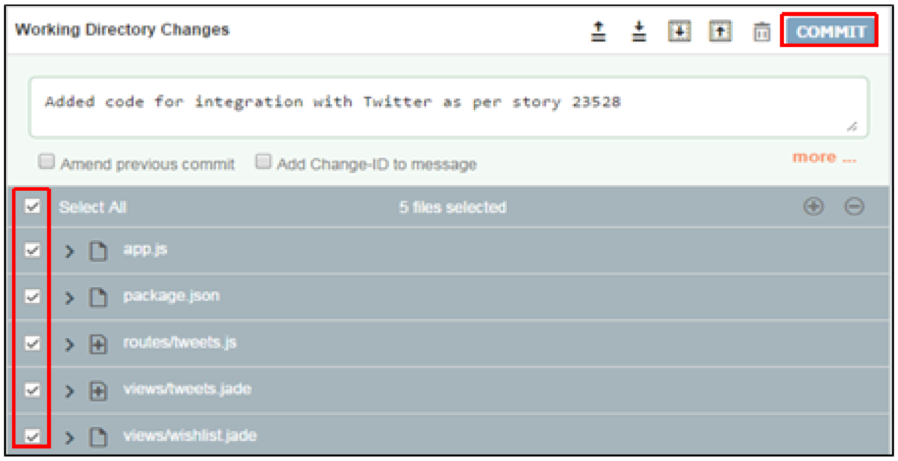
COMMIT

Added code for integration with Twitter as per story 23528

Amend previous commit Add Change-ID to message

more ...

- Select all the files and click **COMMIT**.


Working Directory Changes

COMMIT

Added code for integration with Twitter as per story 23528

Amend previous commit Add Change-ID to message

more ...

Select All 5 files selected

app.js

package.json

routes/tweets.js

views/tweets.jade

views/wishlist.jade

At this point, the files that you have changed in your workspace are saved into the Git repository but they are not visible to other team members.

b. Push your changes to the remote repository

To be visible to the whole team, changes from one developer must be pushed to the remote repository, the repository that all developers share.

- In the **OUTGOING** section of the Git Repository page, click **PUSH** to deliver your changes to the main branch of the repository.
- Your changes are merged with the code in the shared repository.

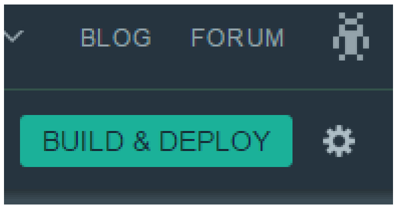
3. Verify the integrity of code delivered to the repository with a build

To support continuous integration, the code delivered to the shared repository must be verified with a build. The success of a build ensures the integrity of the source code.

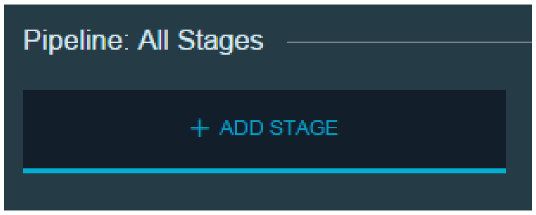
a. Configure a build stage

IBM Bluemix DevOps Services supports manual build and automated build through the Delivery Pipeline. In a pipeline, developers can add a stage to complete build jobs.

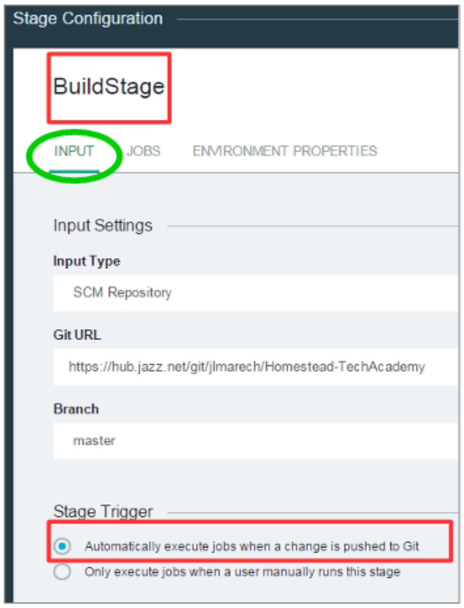
- On the upper right corner of the IBM DevOps Service project page, click the **BUILD & DEPLOY** button.



- When the pipeline page opens, click **ADD STAGE** to create a new stage.



- On the **INPUT** tab, give the stage a name (BuildStage) and make sure that the "*Automatically execute jobs...*" option is checked.
- On the **JOBS** tab, click **ADD JOB,** and select **Build** for the job type and **Simple** for the builder type.


Stage Configuration

BuildStage

INPUT JOBS ENVIRONMENT PROPERTIES

Input Settings

Input Type

SCM Repository

Git URL

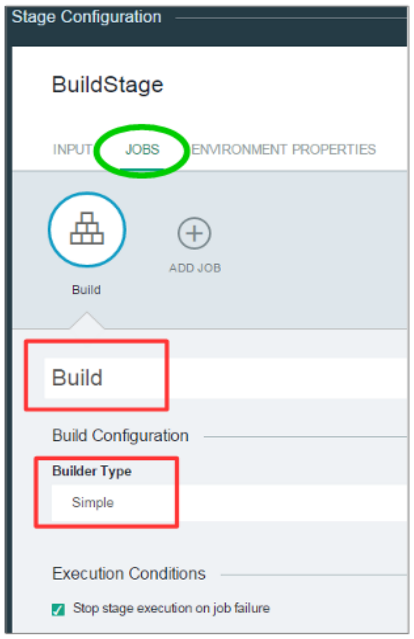
https://hub.jazz.net/git/jlmarech/Homestead-TechAcademy

Branch

master

Stage Trigger

- [✓] Automatically execute jobs when a change is pushed to Git
- [ ] Only execute jobs when a user manually runs this stage


Stage Configuration

BuildStage

INPUT JOBS ENVIRONMENT PROPERTIES

Build

Build Configuration

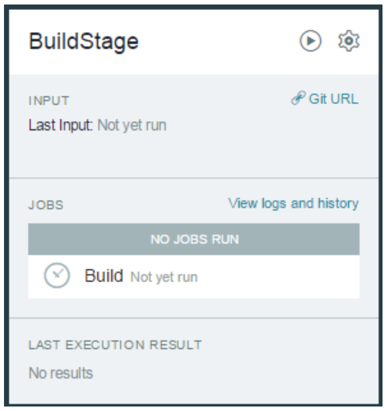
Builder Type

Simple

Execution Conditions

- [✓] Stop stage execution on job failure

- Click **SAVE.** Automated build is now enabled for your project, but the stage can also be used to start a build manually.


BuildStage

INPUT Git URL

Last Input: Not yet run

JOBS View logs and history

NO JOBS RUN

Build Not yet run

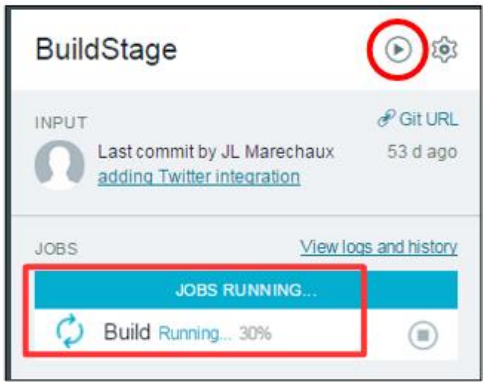
LAST EXECUTION RESULT

No results

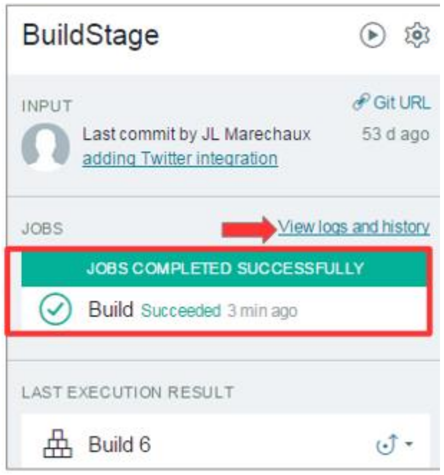
b. Build the cloud application

A stage associated to a build job is useful to verify the integrity of the source code. It can be used to build the application on demand, or to automatically build the application each time a change is delivered to the shared repository.

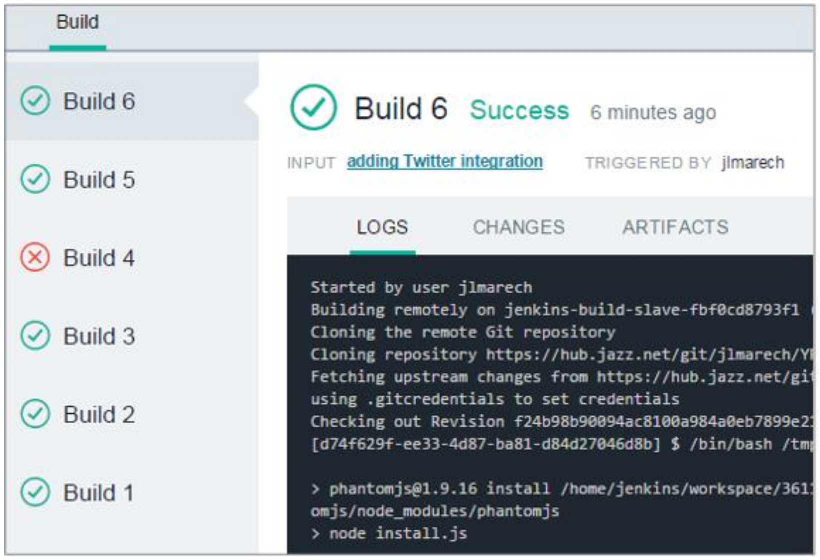
- To start a build of the application manually, click the Run stage button on the upper right corner of the stage. The build starts and the stage indicates progress.



- When the build completes successfully, the status is updated. The **View logs and history** link gives access to more details about past builds.


BuildStage
INPUT
Git URL
Last commit by JL Marechaux
adding Twitter integration
53 d ago
JOBS
View logs and history
JOBS COMPLETED SUCCESSFULLY
Build Succeeded 3 min ago
LAST EXECUTION RESULT
Build 6

- Click the **View logs and history** link on the build stage. You have access to a lot of information about past builds in your project, such as: build history, builds status (success or failure), build duration, build logs, and artifacts included in a build.



```
Started by user jlmarech
Building remotely on jenkins-build-slave-fbf0cd8793f1
Cloning the remote Git repository
Cloning repository https://hub.jazz.net/git/jlmarech/Y
Fetching upstream changes from https://hub.jazz.net/gi
using .gitcredentials to set credentials
Checking out Revision f24b98b90094ac8100a984a0eb7899e2
[d74f629f-ee33-4d87-ba81-d84d27046d8b] $ /bin/bash /tm

> phantomjs@1.9.16 install /home/jenkins/workspace/361
omjs/node_modules/phantomjs
> node install.js
```

e. Describe how use the to Build & Deploy option to manage continuous integration and continuous delivery

1. Understand the Delivery Pipeline service

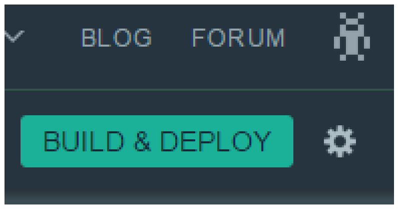
   IBM Bluemix DevOps Services supports continuous delivery through the Delivery Pipeline. In a pipeline, developers can add a stage to deploy their application more often and more quickly.

2. Role of Stages in the Delivery Pipeline, different Stage types and options for Stage Trigger

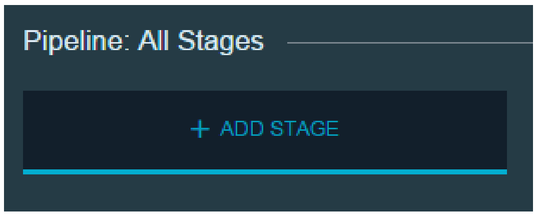
   Creating a Deployment Stage: IBM Bluemix DevOps Services supports continuous delivery through the Delivery Pipeline feature. In a pipeline, developers can add a stage to deploy their application more often and more quickly.

   A set of hands-on steps are provided to show some key concepts for Stages and Delivery Pipeline.

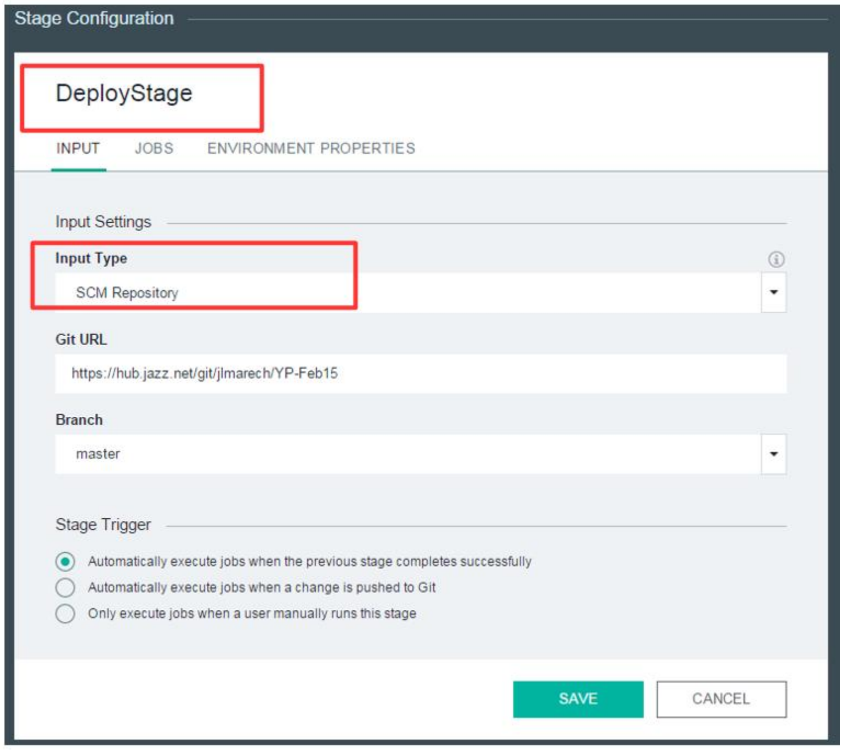
   - On the upper right corner of the IBM DevOps Service project page, click the **BUILD & DEPLOY** button, on the upper-right corner.



   - When the pipeline page opens, click **ADD STAGE** to create a new stage.



   - On the Pipeline page, click **ADD STAGE**.
   - On the **INPUT** tab, specify a stage name (DeployStage), and select **SCM Repository** for the Input Type.


Stage Configuration

DeployStage

INPUT JOBS ENVIRONMENT PROPERTIES

Input Settings

Input Type: SCM Repository

Git URL: https://hub.jazz.net/git/jlmarech/YP-Feb15

Branch: master

Stage Trigger

- [✓] Automatically execute jobs when the previous stage completes successfully
- [ ] Automatically execute jobs when a change is pushed to Git
- [ ] Only execute jobs when a user manually runs this stage

SAVE CANCEL

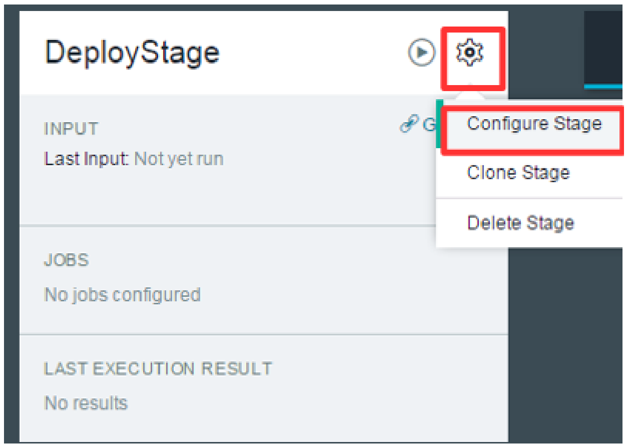
Note that for a deployment stage, it is usually recommended to use a build stage as the input instead of the source code repository (teams want to deploy only successful builds). The SCM repository is used in this example to simplify the scenario.

- Finally, click **SAVE**. The stage for deployment is now created. The next step is to define the jobs included in this stage.

3. Role of Jobs within a Stage and continuation options when a Job fails

Pipeline stages can be configured to include different jobs. Tests jobs are useful to verify that certain conditions are met in an application, for instance before deploying it.

- On the new stage (DeployStage), click the **Stage Configuration** button.

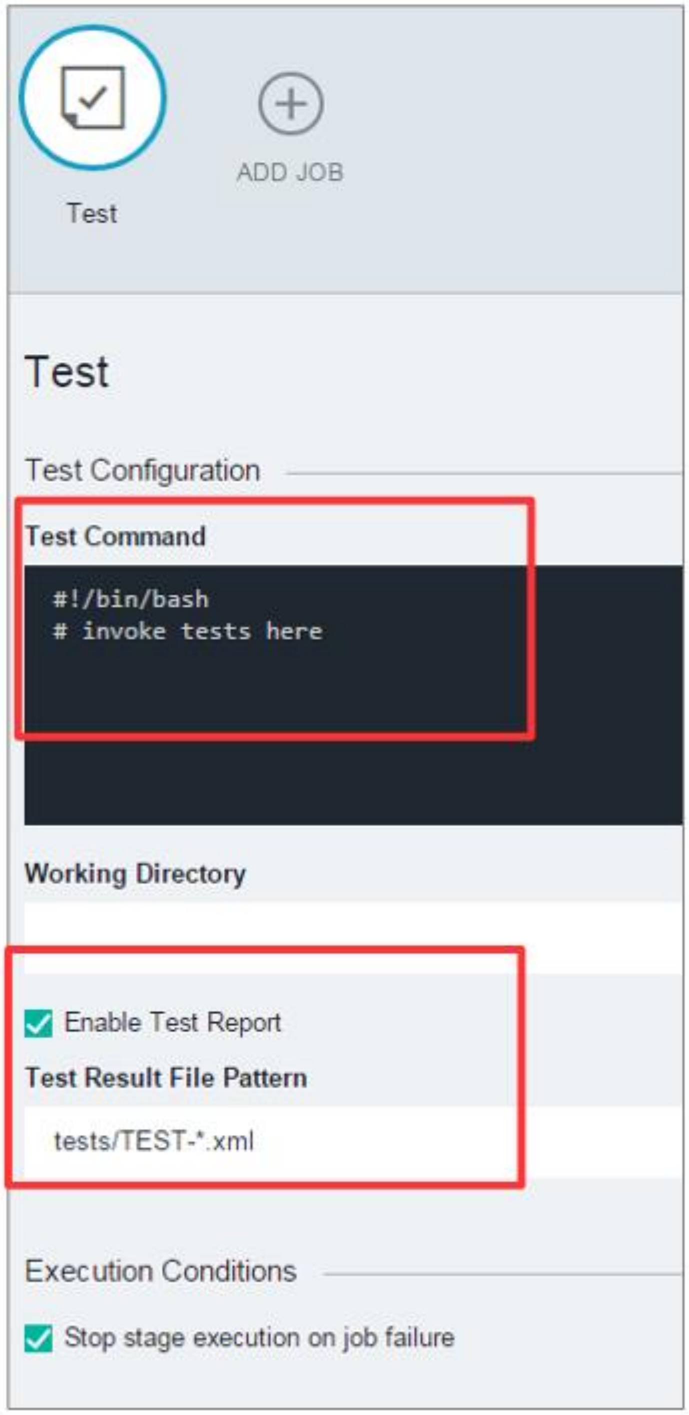

DeployStage

INPUT
Last Input: Not yet run

JOBS
No jobs configured

LAST EXECUTION RESULT
No results

- On the **JOBS** tab, click **ADD JOB** and select **Test** for the job type.
- Check the option to enable test reports.
- Note that the Test Command can be modified to invoke some test scripts, but this feature is out of the scope of this example.



Test

Test Configuration

Test Command

```
#!/bin/bash
# invoke tests here
```

Working Directory

✓ Enable Test Report

Test Result File Pattern

tests/TEST-*.xml

Execution Conditions

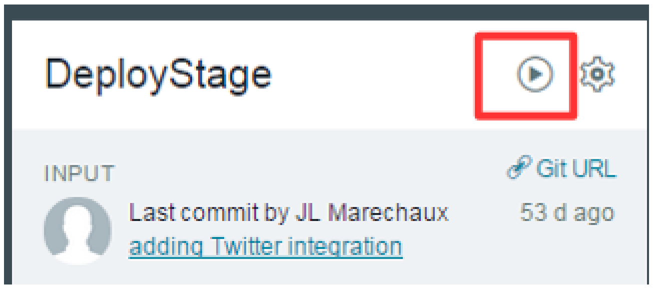
✓ Stop stage execution on job failure

Note that if the **Stop stage execution on job failure** option is selected, the stage will halt if the job fails. Otherwise, the stage will continue to execute the next job.

4. Understand relationships between changes to the source code repository and pipeline stages and automatically or manually run jobs in a stage

   IBM Bluemix DevOps Services supports manual and automated deployments through the Delivery Pipeline. In a pipeline, developers can manually run the jobs included in a stage or configure the stage for automated deployments.

   - To manually start the jobs in BuildStage, click the Run Stage button.



DeployStage

INPUT

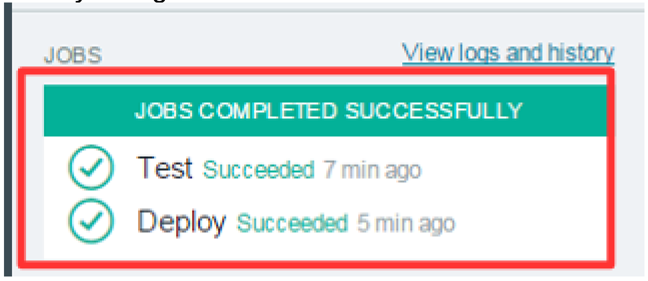
Git URL

Last commit by JL Marechaux

53 d ago

adding Twitter integration

   - The first job (Test) starts, and when it completes successfully, the second job (Deploy) is triggered.
   - When jobs are completed, the status of the stage is updated. The View logs and history link gives access to detailed information for each job.



JOBS

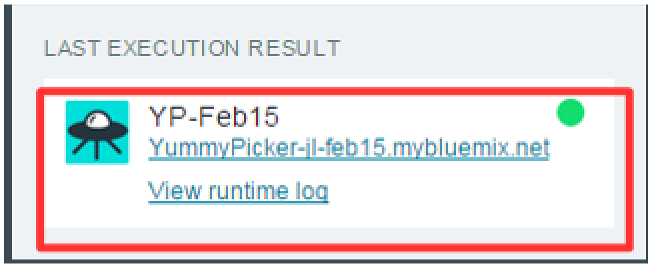
View logs and history

JOBS COMPLETED SUCCESSFULLY

Test Succeeded 7 min ago

Deploy Succeeded 5 min ago

   - If the deploy jobs completed successfully, the stage contains a link to the deployed application on Bluemix.



LAST EXECUTION RESULT

YP-Feb15

YummyPicker-jl-feb15.mybluemix.net

View runtime log

   - The developer can then access the deployed application using the link.

Section 6: Using Data Services

a. Describe the different types of data services available in IBM Bluemix PaaS

1. Key characteristics of a NoSQL database service
    a. NoSQL databases are managed data stores that do not follow the traditional relational database model. NoSQL databases store data using various models including column, graph, key-value, and documents. Examples of NoSQL document databases in IBM Bluemix PaaS are Cloudant (an extension of Apache CouchDB) and MongoDB.
    b. NoSQL document databases offer flexibility as there is no enforced schema for documents and document structure can be changed at any time without having to modifying existing data.
    c. NoSQL databases like Apache CouchDB and Cloudant can scale horizontally on commodity hardware to provide low cost, scalable performance as data volumes grow. In order to support this type of scalability Apache CouchDB and Cloudant were designed to prioritize database availability over consistency between distributed nodes. This is referred to as "eventual consistency".
2. Key characteristics of a SQL database service
    a. A SQL database service provides applications access to relational databases that are queried using SQL.
    b. The SQL Database service in Bluemix is an example of an SQL data service that is powered by IBM DB2 and has a variety of features including high availability, automated backups and data privacy.
3. Key characteristics of a in-memory columnar database service
    a. An in memory columnar database service provides optimized database access for analytic workloads.
    b. The dashDB service in IBM Bluemix is an example of an in memory, columnar databases service. dashDB is based on DB2 BLU and its columnar organizations favors analytic queries that access a large number of values from a subset of the columns and make heavy use of aggregations and joins. dashDB also leverages compression to allow larger amounts of data to reside in memory.
4. Key characteristics of key value pair data service
    a. Key value pair data services allow the efficient storage and retrieval of key value pair data.
    b. Examples of key value pair data services in IBM Bluemix include Redis, the IBM Data Cache for Bluemix and the IBM Session Cache for Bluemix. These services are particularly useful in web applications that need to maintain state between requests because they offer fast, efficient access to stateful data that is accessible by multiple instances
5. Key characteristics of a time series database service
    a. A time series database enables the efficient storage and retrieval of time series data.
    b. The IBM Time Series Database for Bluemix service is an example of a time series database service. The IBM Time Series Database for Bluemix can store time series data in up to 1/3 the space of other databases and provides SQL extensions specific to time series data.

References:

http://guide.couchdb.org/draft/consistency.html#consistency
http://www.ng.bluemix.net/docs/#services/Cloudant/index.html#Cloudant
http://www.ng.bluemix.net/docs/#services/SQLDB/index.html#SQLDB
http://www.ng.bluemix.net/docs/#services/dashDB/index.html#dashDB
http://www.ng.bluemix.net/docs/#services/TimeSeries/index.html#timeseriesdatabase
http://www.ng.bluemix.net/docs/#services/SessionCache/index.html#session_cache

b. Describe the unique features of IBM Bluemix PaaS data services

1. Understand the unique features of Cloudant NoSQL Database
    a. IBM Cloudant NoSQL DB for Bluemix is a NoSQL database as a service (DBaaS) that scales globally, runs non-stop, and handles data in JSON format and supports full text queries and geospatial queries. Cloudant NoSQL DB is an operational data store optimized to handle concurrent reads and writes, and provide high availability and data durability.

    b. Cloudant provides a RESTful API to create, read, update and delete documents

       Reading a document:

       Cloudant's RESTful API makes every document in your Cloudant database accessible as JSON via a URL; this is one of the features that make Cloudant so powerful for web applications.

       Each document in Cloudant has a unique _id field which can be used to retrieve it.

       To retrieve it via the API, simply append the document's id to the URL of the database. For a document of _id foo in the mydb database the GET URL to retrieve the document would look like this

       ```
       GET https://[username].cloudant.com/mydb/foo
       ```

       Insert a document:

       Documents can be inserted into Cloudant individually or in bulk.

       To insert documents you have the option to provide a unique value for the _id field. If the document to be inserted doesn't define a _id value, one gets assigned on insert.

       If you define a _id for your document up front you need to make sure that the _id isn't already in use. If it is, the insert will fail.

       Code examples

       Insert via CURL

       Command
       ```
       curl -d '{"season": "summer", "weather": "usually warm and
       sunny"}' -X POST https://[username].cloudant.com/crud/ -H
       "Content-Type:application/json"
       ```
       Response
       ```
       {"ok":true,"id":"590e2bca76c09882e37dea534b000be3","rev":"1-
       0af5e64fe24d262db237b9f14046f490"}
       ```

       If you want to set the _id when you insert, you can do it in two ways: POST and PUT.

       Set the _id via POST

POST the document with the _id in the document body:

Command

```
curl -d '{"season": "summer", "weather": "usually warm and sunny",
"_id":"foo"}' -X POST https://[username].cloudant.com/crud/ -H
"Content-Type:application/json"
```

Response

```
{"ok":true,"id":"foo","rev":"1-0af5e64fe24d262db237b9f14046f490"}
```

Set the _id via PUT

Or PUT the document, specifying the _id in the URL:

Command

```
curl -d '{"season": "summer", "weather": "usually warm and
sunny"}' -X PUT https://[username].cloudant.com/crud/bar -H
"Content-Type:application/json"
```

Response

```
{"ok":true,"id":"bar","rev":"1-0af5e64fe24d262db237b9f14046f490"}
```

Update and delete documents:

The _rev field gets added to your documents by the server when you insert or modify them, and is included in the server response when you make changes or read a document. The _rev is built from a crude counter and a hash of the document and is used to determine what needs to be replicated between servers, and if a client is trying to modify the latest version of a document. For this reason updates need to send the _rev token to be able to modify a document.

It is important to note that _rev **should not** be used to build a version control system, it's an internal value used by the server and older revisions are transient, and removed regularly.

The code or command line to update a document is the same as to insert, just **be sure to include the _rev in the document body**.

As you might expect deletions are done by using the DELETE HTTP method. There are some cases where firing a DELETE might not be possible so you can also delete a document by adding _deleted to the document and update it. This is especially useful for bulk operations, where many documents may be created, updated or deleted in a single HTTP operation. As you'll be removing the document you can delete the rest of its contents, apart from the _id, _rev and _deleted fields. If you leave other fields they will be in the documents "tombstone", this can be useful when replicating or validating document edits.

Code examples

To delete a document you need its _id and _rev, the easiest way to get the _rev for a known document _id is to issue a HEAD request against the document:

Get the _rev

Command
```
curl -i -X HEAD https://[username].cloudant.com/crud/[doc_id]
```
Response
```
HTTP/1.1 200 OK
X-Couch-Request-ID: 89d9d456
Server: CouchDB/1.0.2 (Erlang OTP/R14B)
ETag: "2-e4b98cc0fc2a181e4eb26f8ad41fa5fe"
Date: Mon, 04 Jun 2012 14:47:15 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 113
Cache-Control: must-revalidate
```

Delete the document

CouchDB sets the ETag to be the document _rev (which is handy for caching purposes). Now we have the _id and _rev we can delete the document with:

Command
```
curl -X DELETE
https://[username].cloudant.com/crud/[doc_id]\?rev\=[doc_rev]
```
Response
```
{"ok":true,"id":"[doc_id]","rev":"[doc_rev]"}
```

Verify deletion
Command
```
curl  https://[username].cloudant.com/crud/[doc_id]
```
Response
```
{"error":"not_found","reason":"deleted"}
```

Delete via PUT

You can also delete a document via a PUT by adding the _deleted attribute to the document:

Command
```
curl -d '{"_rev":"[doc_rev]", "_deleted":true}' -X PUT
https://[username].cloudant.com/crud/[doc_id]
```
Response
```
{"ok":true,"id":"[doc_id]","rev":"[doc_rev]"}
```

Verify deletion
Command
```
curl  https://[username].cloudant.com/crud/[doc_id]
```
Response
```
{"error":"not_found","reason":"deleted"}
```

Reference: https://cloudant.com/for-developers/crud/

c. Cloudant allows the creation of indexes via the use of MapReduce

Secondary indexes, or views, are defined in a map function, which pulls out data from your documents and an optional reduce function that aggregates the data emitted by the map.

These functions are written in JavaScript and held in "design documents"; special documents that the database knows contain these - and other - functions. Design documents are special documents that define secondary indexes.

A sample design document with MapReduce functions

```
{
  "_id": "_design/name",
  "views": {
    "view1": {
      "map":"function(doc){emit(doc.field, 1)}",
      "reduce": "function(key, value, rereduce){return sum(values)}"
    }
  }
}
```

The naming convention for design documents is such that the name follows _design/ in the _id. This code defines view1 for the design document name. Design documents can contain multiple views; each is added to the views object.

Reference: https://cloudant.com/for-developers/views/

d. Cloudant Query provides a declarative way to define and query indexes

Cloudant Query is a declarative JSON querying syntax for Cloudant databases. Cloudant Query wraps several index types, starting with the Primary Index out-of-the-box. Cloudant Query indexes can also be built using MapReduce Views (where the index type is json), and Search Indexes (where the index type is text).

If you know exactly what data you want to look for, or you want to keep storage and processing requirements to a minimum, you can specify how the index is created, by making it of type json.

For maximum possible flexibility when looking for data, you would typically create an index of type text. Indexes of type text have a simple mechanism for automatically indexing all the fields in the documents.

A sample json document to create a query index:

```
    {
          "index": {
             "fields": [“make"]
          },
          “name": “make-index“,
          “type": “json“
    }
```

Once the query index is created, the index may be searched using a selector which is also written as a json document. Selectors are conceptually similar to a SQL statement's WHERE clause.

An example selector to return documents with the make VW with model year 2000 or later:

```
{
    “selector": {
        “make": “VW“,
        “year“: {“$ge“: 2000}
    }
}
```

Cloudant Query provides two api endpoints, _index and _find that are used to manage indexes and perform queries against them using selectors.

Reference: https://docs.cloudant.com/cloudant_query.html

e. Cloudant Sync simplifies large-scale mobile development

   Cloudant Sync enables you to push database access to the farthest edge of the network —mobile devices, remote facilities, sensors, and internet-enabled goods, so that you can:

   - Scale bigger
   - Enable client apps to continue running off-line

   Cloudant Sync allows mobile and distributed apps to scale by replicating and synching data between multiple readable, writeable copies of the data in other data centers, and even on mobile iOS and Android devices. This is much easier and more cost efficient than growing a single, central database to handle all data collection.

   Cloudant Sync allows you to create a single database for every user; you simply replicate and sync the copy of this database in Cloudant with a local copy on their phone or tablet (or vehicle, sensor, appliance, etc.). This can reduce round-trip database requests with the server. If there's no network connection, the app runs off the database on the device; when the network connection is restored, Cloudant re-syncs the device and server. Some of Cloudant's largest mobile developers have scaled into the millions of databases.

2. Understand the unique features of dashDB
   a. dashDB is a data warehousing service that stores relational data, including special types such as geospatial data. Stored data can be analyzed with SQL or advanced built-in analytics like predictive analytics and data mining, analytics with R, and geospatial analytics. dashDB provides in-memory database technology to support both columnar and row-based tables.

   b.

3. Understand the unique features of the IBM Time Series Database for Bluemix

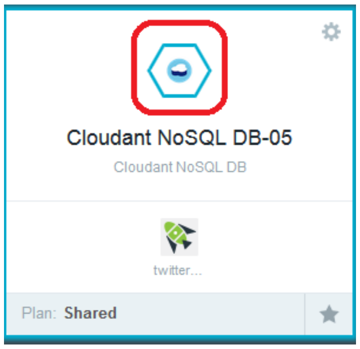
a. IBM Time Series Database for Bluemix service is a managed data store for Internet of Things device data and time series analysis of the data. The Time Series database service supports multiple methods for applications to access it to store, update, and query the data in a Time Series Database including the MongoDB APIs, a REST API, the IoT REST API, and the IBM Informix JDBC API.

c. Manage instances of IBM Bluemix PaaS data services: Cloudant NoSQL Database, and dashDB.

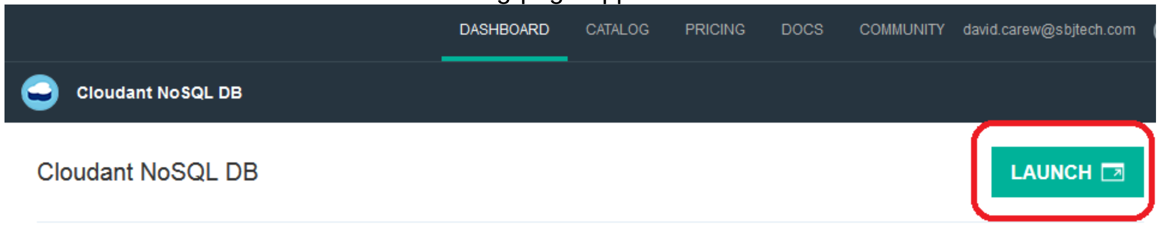
1. Manage instances of the Cloudant NoSQL DB service
Once an instance of the Cloudant NoSQL database service has been created in IBM Bluemix you can use the provided web interface to administer the database.

A set of hands-on steps are provided to show some key concepts of managing the Cloudant NoSQL DB service.

Launch the Cloudant NoSQL administration tool
Click on the hexagon for your instance of Cloudant NoSQL database service by clicking on its hexagon from the Bluemix Dashboard.



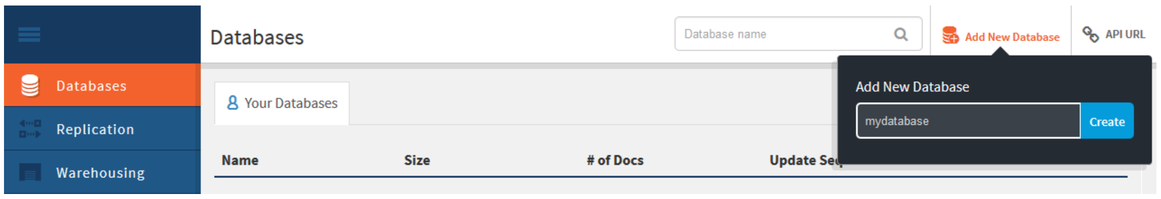
Click on Launch when the services Landing page appears:



a. Create a database

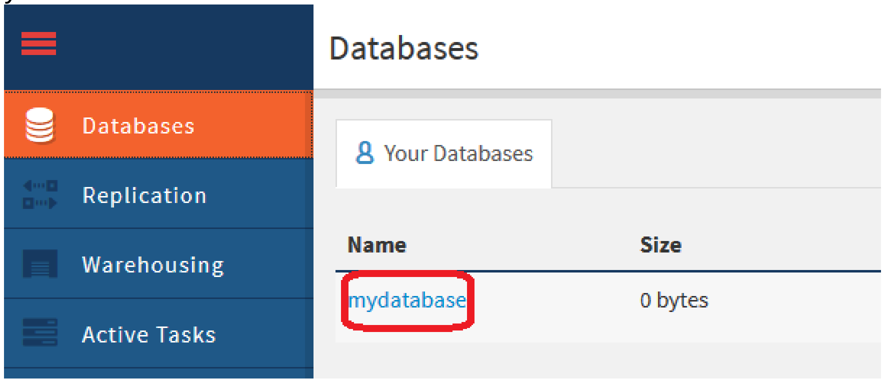
From the Cloudant Administration Tool click on Add New Database, enter the name that you want for the new database and click Create.

Note that the **must** start with a lowercase letter and contain only the following characters:
- Lowercase characters (a-z)
- Digits (0-9)
- Any of the characters _, $, (,), +, -, and /

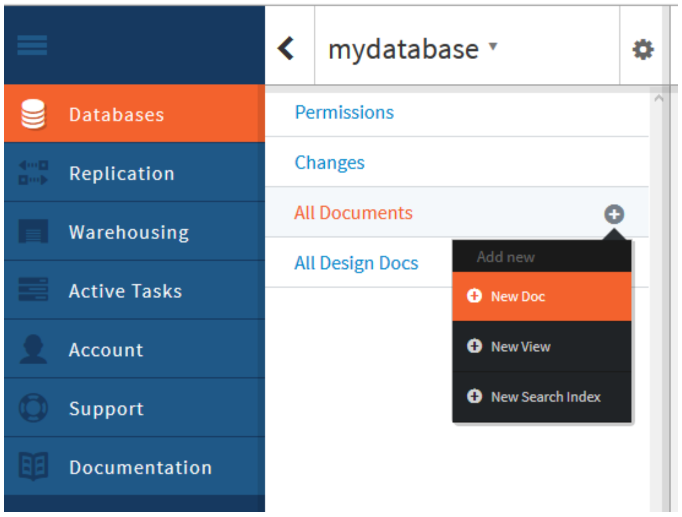


You'll be taken to the administration screen for the new database.

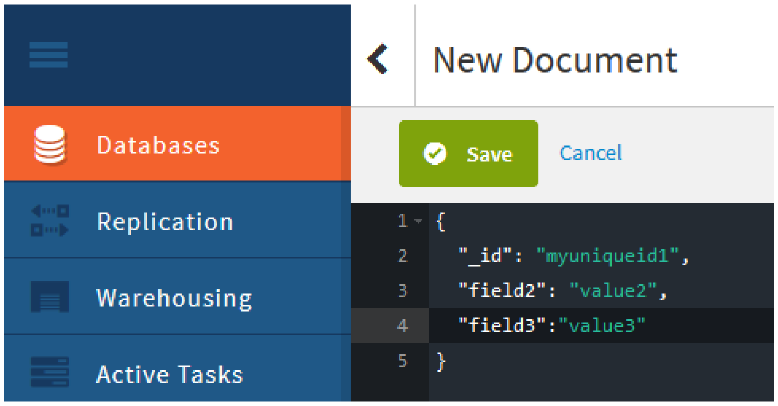
b. Add data to an existing database
From the home page of the Cloudant Administration tool click on the link for the database that you want to administer:



Click on the + icon next to All Documents and select New Doc from the context menu:



A new JSON document appears with a single attribute name _id. This is the unique identifier for your new document. You can accept the generated value or put it your own. Add the additional fields to the document.

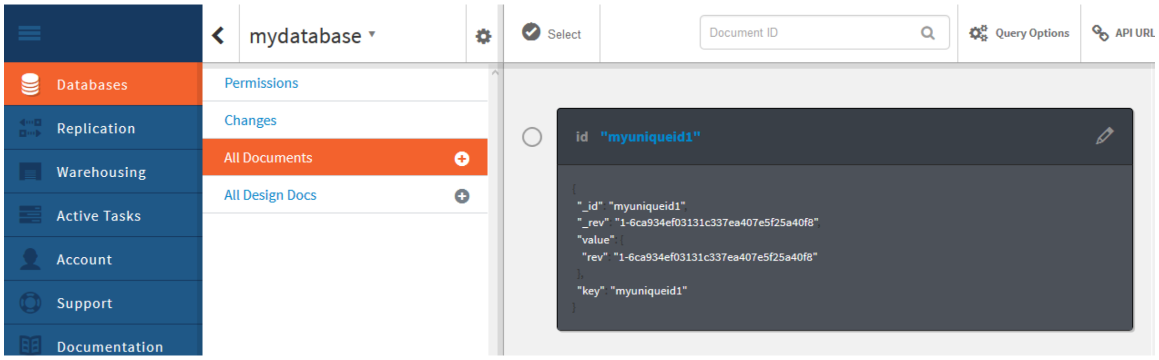


```
{
  "_id": "myuniqueid1",
  "field2": "value2",
  "field3":"value3"
}
```

When you're done click Save to save the changes.

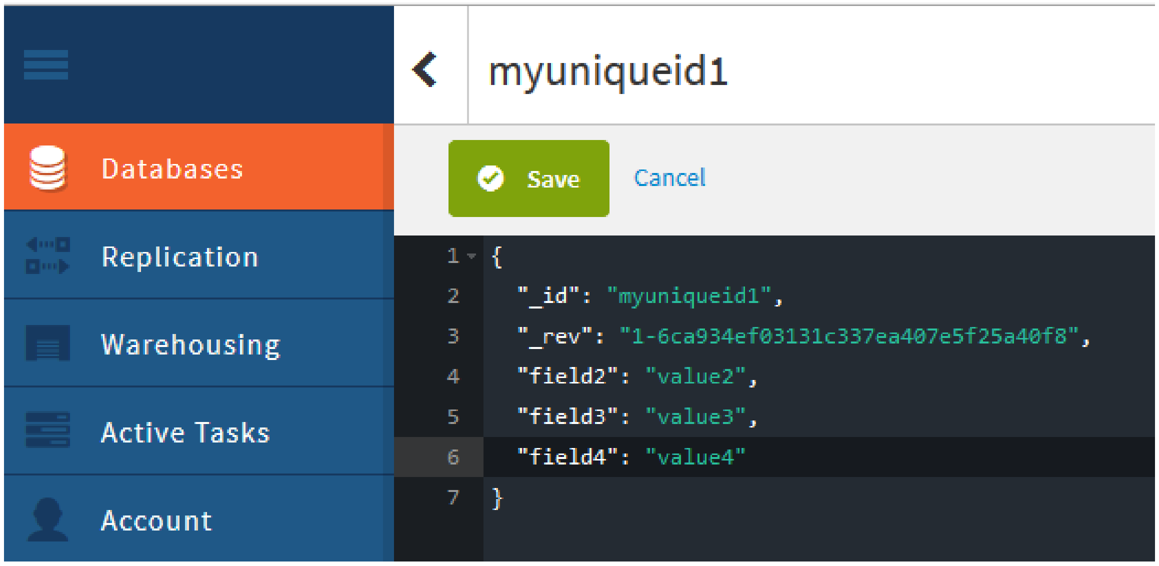
c. Edit documents in an existing database
From the home page of the Cloudant Administration tool click on the link for the database that you want to administer.

Click on All Documents, a summary of the documents in the database appears on the right:



```
{
 "_id": "myuniqueid1",
 "_rev": "1-6ca934ef03131c337ea407e5f25a40f8",
 "value": {
  "rev": "1-6ca934ef03131c337ea407e5f25a40f8"
 },
 "key": "myuniqueid1"
}
```

Click on the pencil icon to edit the documents and then make your required changes.

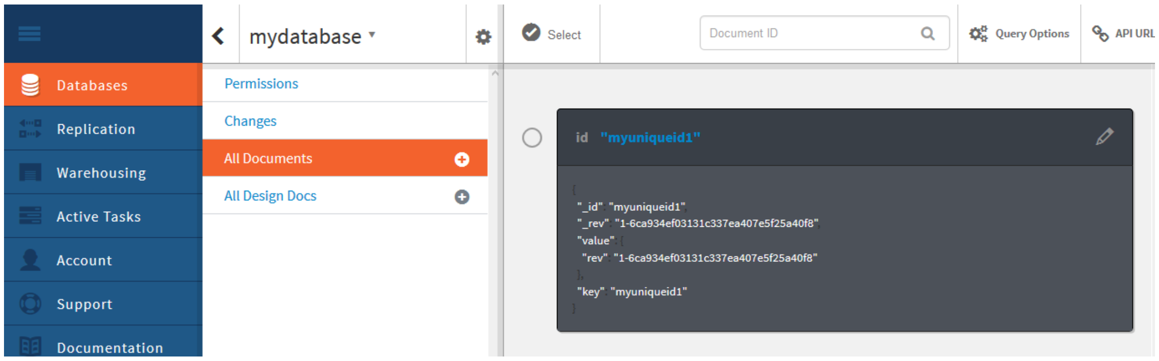


```
{
  "_id": "myuniqueid1",
  "_rev": "1-6ca934ef03131c337ea407e5f25a40f8",
  "field2": "value2",
  "field3": "value3",
  "field4": "value4"
}
```

Click Save to save your changes.

d. Clone existing documents
From the home page of the Cloudant Administration tool click on the link for the database that you want to administer.

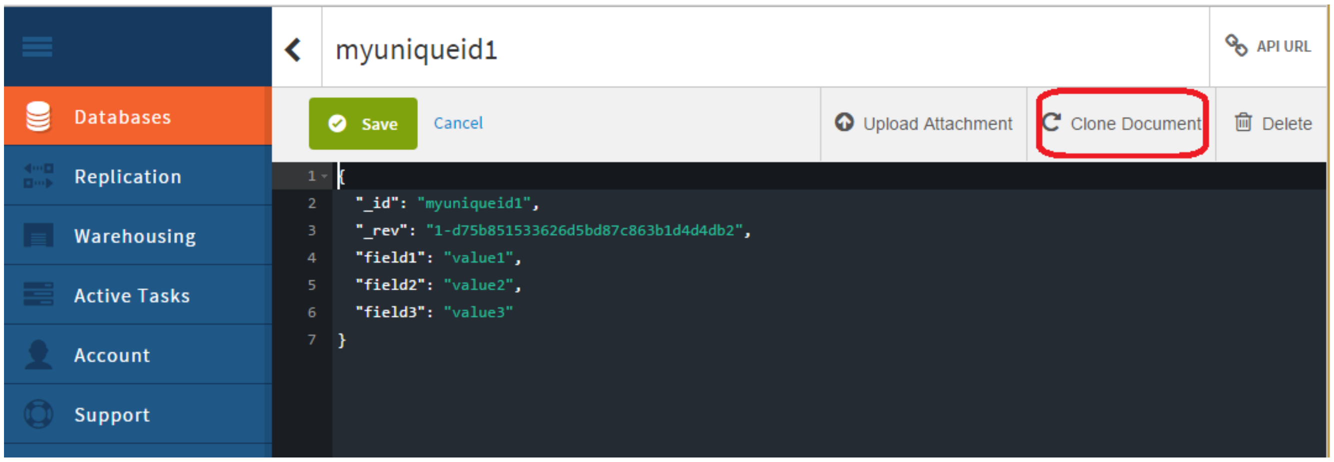
Click on All Documents, a summary of the documents in the database appears on the right:



```
{
 "_id": "myuniqueid1",
 "_rev": "1-6ca934ef03131c337ea407e5f25a40f8",
 "value": {
  "rev": "1-6ca934ef03131c337ea407e5f25a40f8"
 },
 "key": "myuniqueid1"
}
```

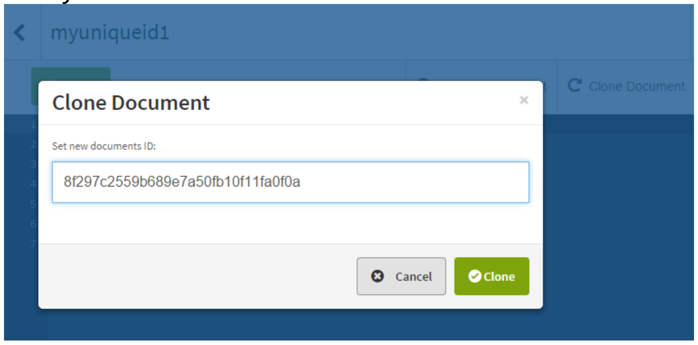
Click on the pencil icon of the document you want to clone.

Click on Clone Document in the document editor:



```
{
  "_id": "myuniqueid1",
  "_rev": "1-d75b851533626d5bd87c863b1d4d4db2",
  "field1": "value1",
  "field2": "value2",
  "field3": "value3"
}
```

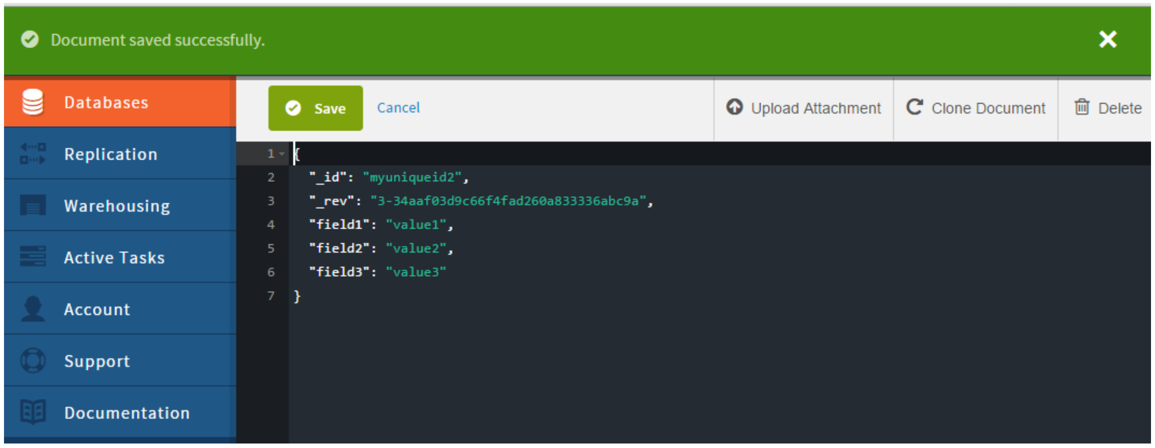
You'll be prompted accept a system generated unique id for the new clone or to provide a unique id of your choice:



Set new documents ID:

8f297c2559b689e7a50fb10f11fa0f0a

Change the ID to something unique (or accept the generated one) and click Clone. You'll be taken to a document editor with the clone of the original document which is identical to the original (except for the _id and _rev fields):

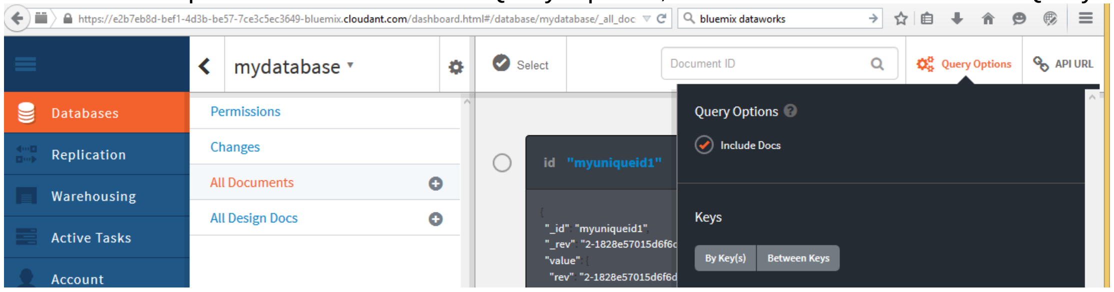


```
{
  "_id": "myuniqueid2",
  "_rev": "3-34aaf03d9c66f4fad260a833336abc9a",
  "field1": "value1",
  "field2": "value2",
  "field3": "value3"
}
```

e. Simple query of all documents in an existing database
From the home page of the Cloudant Administration tool click on the link for the database that you want to administer.

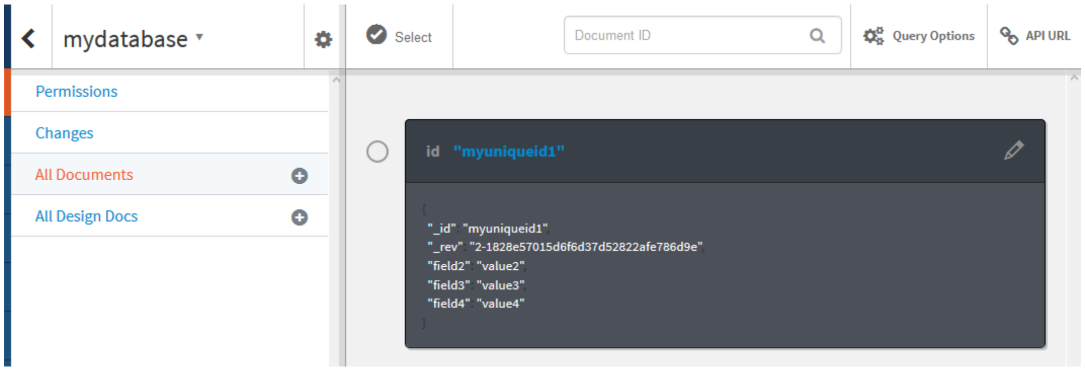
Click on All Documents, a summary of the documents in the database appears on the right.

To see the complete documents click on Query Options, select Include Docs and click Query:



```
{
 "_id": "myuniqueid1",
 "_rev": "2-1828e57015d6f6c
 "value": {
  "rev": "2-1828e57015d6f6d
```

All fields in all your documents will appear.



```
{
 "_id": "myuniqueid1",
 "_rev": "2-1828e57015d6f6d37d52822afe786d9e",
 "field2": "value2",
 "field3": "value3",
 "field4": "value4"
}
```

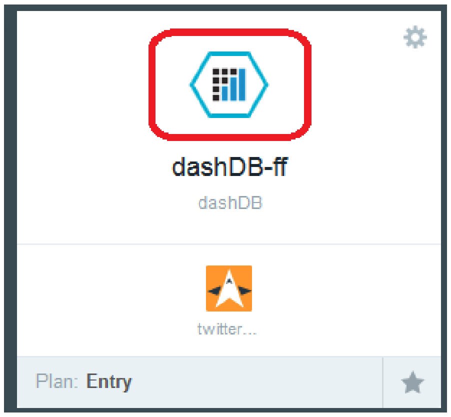
2. Manage instances of dashDB service

Once an instance of the dashDB service has been created in IBM Bluemix you can use the provided web interface to administer it. There are many features in the tool including the capability to load CSV data, load Geospatial data and to sync from an existing Cloudant database.

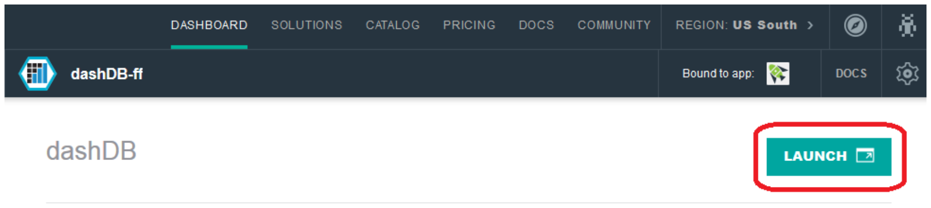
A set of hands-on steps are provided to show some key concepts of managing the dashDB service.

Launch the dashDB administration tool

Click on the hexagon for your instance of the dashDB service by clicking on its hexagon from the Bluemix Dashboard:

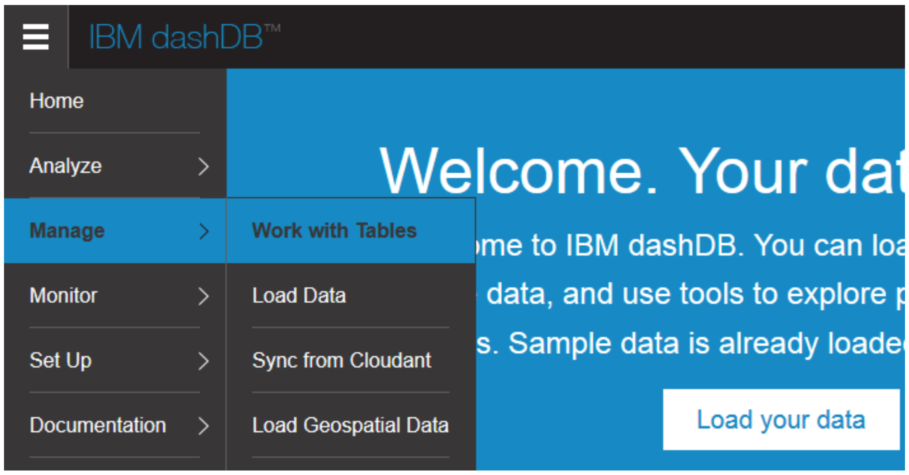


Click on Launch when the service's landing page appears:

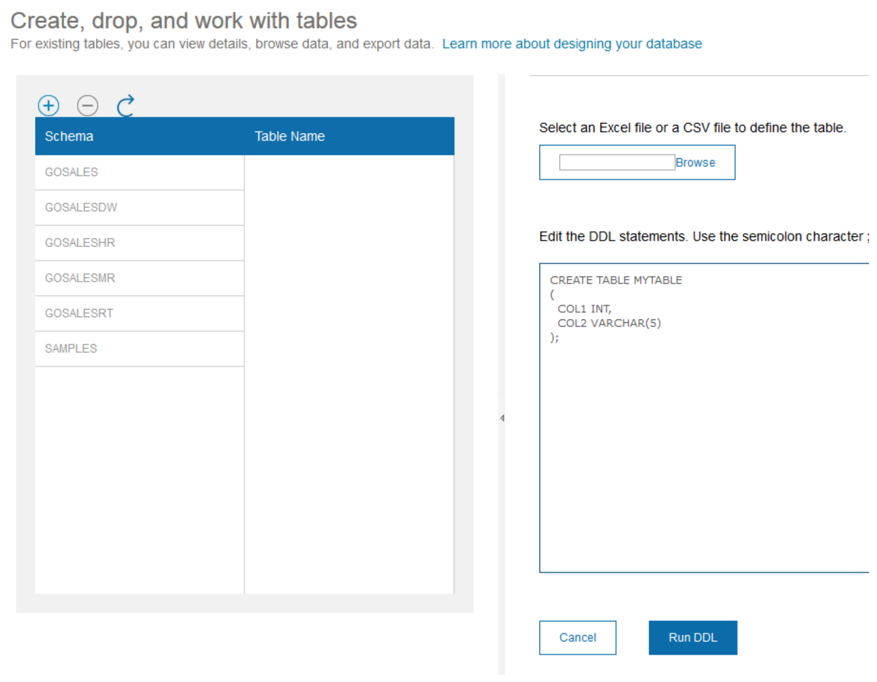


a. Create a new table in dashDB

From the dashDB Administration tool landing page select Manage->Work with tables:



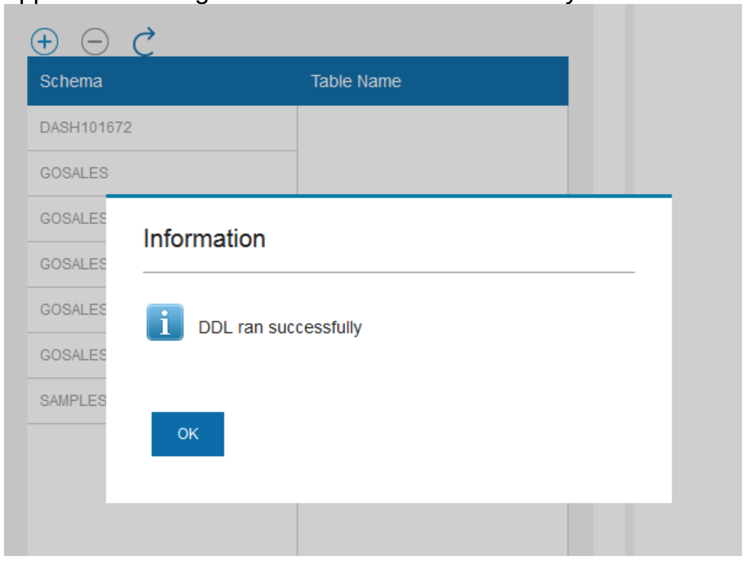
Click on the + icon to enter the SQL DDL to create a new table. (Note dashDB uses DB2 10.5 SQL syntax). Don’t specify a new schema name as the dashDB user that has been created for you does not have CREATE SCHEMA authority. Leaving the schema name out altogether will create the table with the default schema.



Create, drop, and work with tables
For existing tables, you can view details, browse data, and export data. Learn more about designing your database
Schema
Table Name
GOSALES
GOSALESDW
GOSALESHR
GOSALESMR
GOSALESRT
SAMPLES
Select an Excel file or a CSV file to define the table.
Browse
Edit the DDL statements. Use the semicolon character ;
Cancel
Run DDL

```
CREATE TABLE MYTABLE
(
  COL1 INT,
  COL2 VARCHAR(5)
);
```

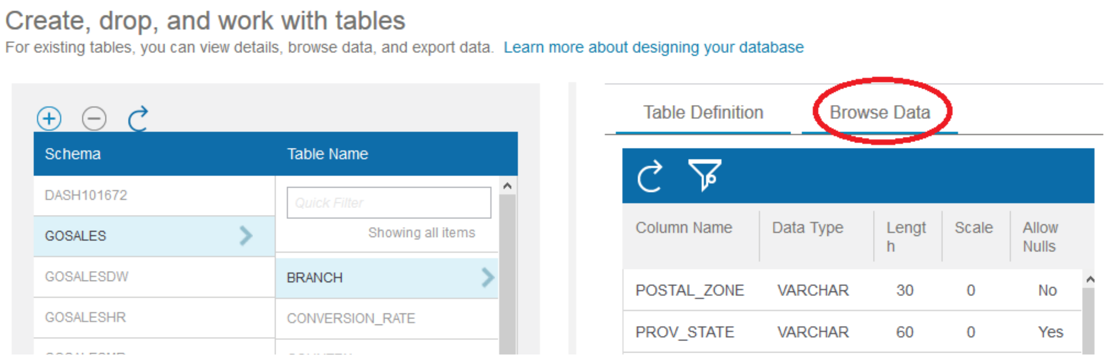
After you entered the DDL to create the table, click Run DDL to create the table. A dialog will appear indicating that the table was successfully created:



DDL ran successfully

b. Browse the contents of an existing table in dashDB
From the dashDB Administration tool landing page select Manage->Work with tables.

Several sample tables have already been created for you. Select GOSALES as the schema and BRANCH as the table. Click Browse Data:



Create, drop, and work with tables

For existing tables, you can view details, browse data, and export data. Learn more about designing your database

Schema	Table Name
DASH101672	
GOSALES	BRANCH
GOSALESDW	CONVERSION_RATE
GOSALESHR	

Table Definition | Browse Data

Column Name	Data Type	Length	Scale	Allow Nulls
POSTAL_ZONE	VARCHAR	30	0	No
PROV_STATE	VARCHAR	60	0	Yes

The first 1000 rows of data will be returned.

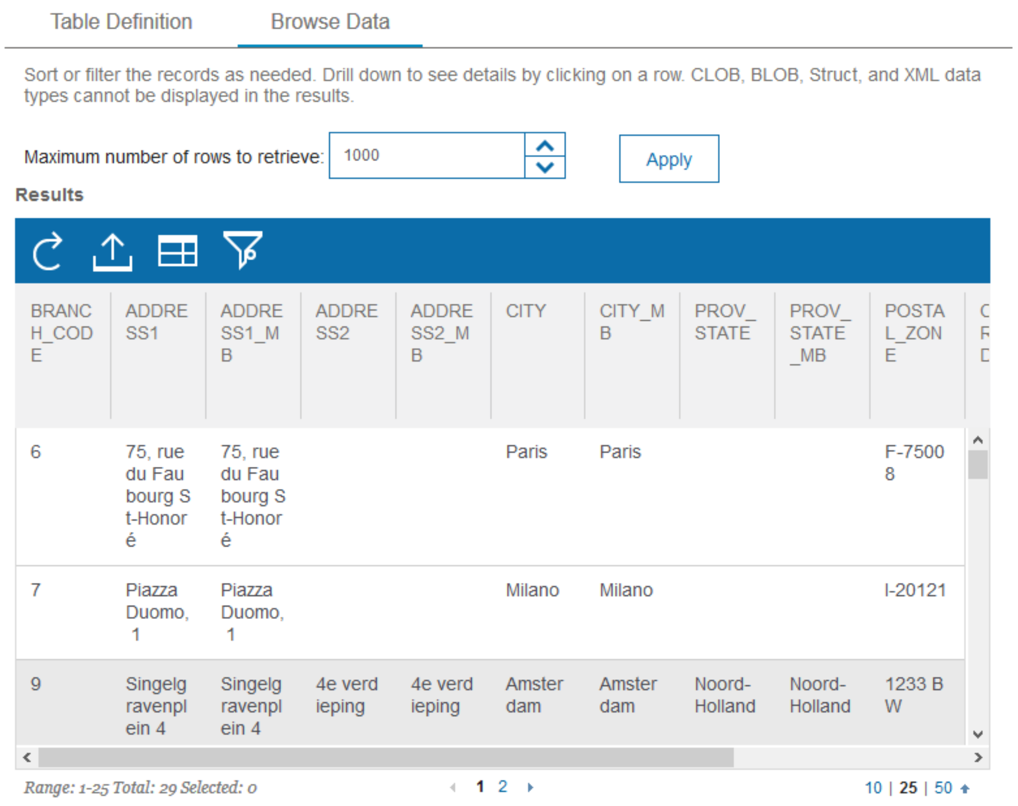


Table Definition | Browse Data

Sort or filter the records as needed. Drill down to see details by clicking on a row. CLOB, BLOB, Struct, and XML data types cannot be displayed in the results.

Maximum number of rows to retrieve: 1000 Apply

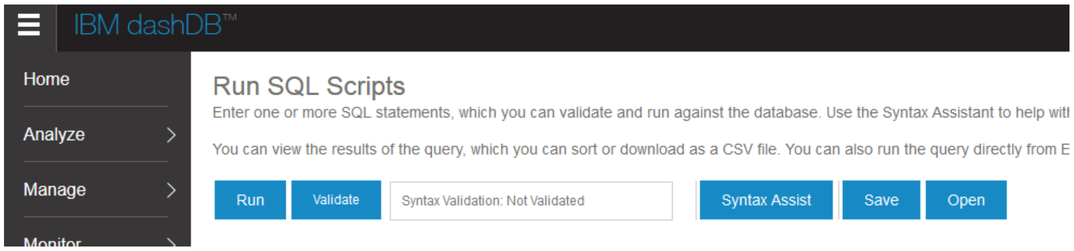
Results

BRANCH_CODE	ADDRESS1	ADDRESS1_MB	ADDRESS2	ADDRESS2_MB	CITY	CITY_MB	PROV_STATE	PROV_STATE_MB	POSTAL_ZONE
6	75, rue du Faubourg St-Honoré	75, rue du Faubourg St-Honoré			Paris	Paris			F-75008
7	Piazza Duomo, 1	Piazza Duomo, 1			Milano	Milano			I-20121
9	Singelgravenplein 4	Singelgravenplein 4	4e verdieping	4e verdieping	Amsterdam	Amsterdam	Noord-Holland	Noord-Holland	1233 BW

Range: 1-25 Total: 29 Selected: 0 1 2 10 | 25 | 50

c. Run SQL scripts in dashDB

You can run SQL script in dashDB with the Administration tool. The tool provides a Validate button to verify that your script is valid SQL, there is also a Syntax Assist button will guide you through the creation of common SQL query types and a Run button that will run the scripts, Scripts can be named and saved for future use.

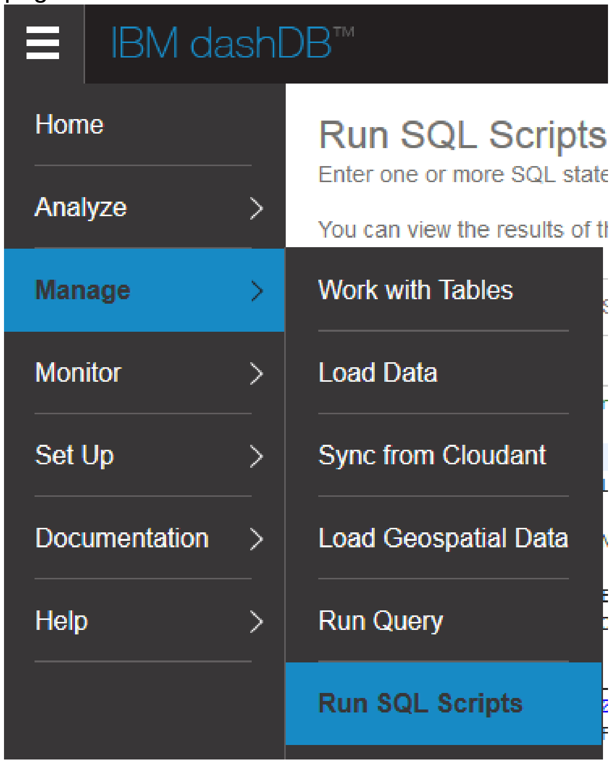


Run SQL Scripts

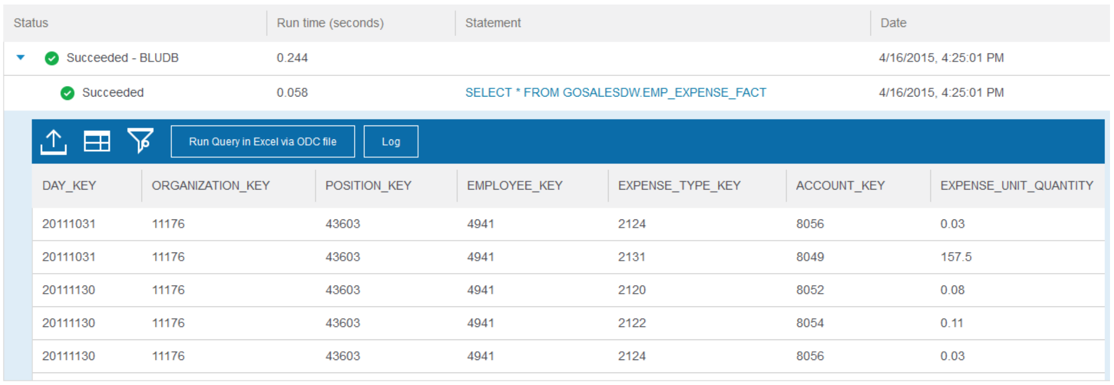
Enter one or more SQL statements, which you can validate and run against the database. Use the Syntax Assistant to help wit

You can view the results of the query, which you can sort or download as a CSV file. You can also run the query directly from E

To run SQL script, select Manage->Run SQL Scripts from the dashDB Administration tool landing page:



An SQL script querying some of the sample data will be preloaded for you. Click on Run to see the results of running the script.



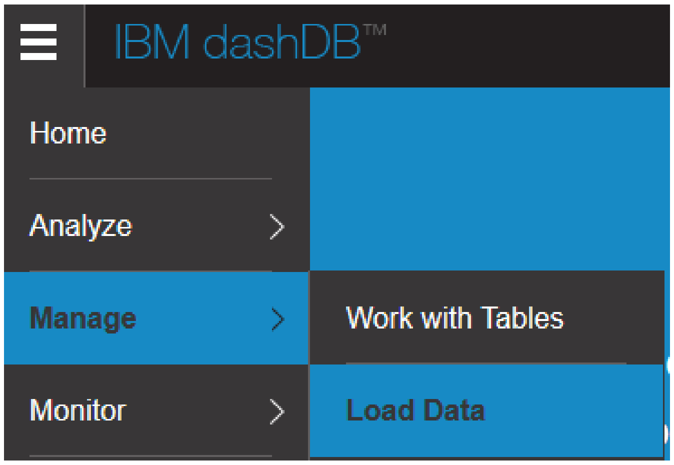
Status	Run time (seconds)	Statement	Date
Succeeded - BLUDB	0.244		4/16/2015, 4:25:01 PM
Succeeded	0.058	SELECT * FROM GOSALESDW.EMP_EXPENSE_FACT	4/16/2015, 4:25:01 PM

DAY_KEY	ORGANIZATION_KEY	POSITION_KEY	EMPLOYEE_KEY	EXPENSE_TYPE_KEY	ACCOUNT_KEY	EXPENSE_UNIT_QUANTITY
20111031	11176	43603	4941	2124	8056	0.03
20111031	11176	43603	4941	2131	8049	157.5
20111130	11176	43603	4941	2120	8052	0.08
20111130	11176	43603	4941	2122	8054	0.11
20111130	11176	43603	4941	2124	8056	0.03

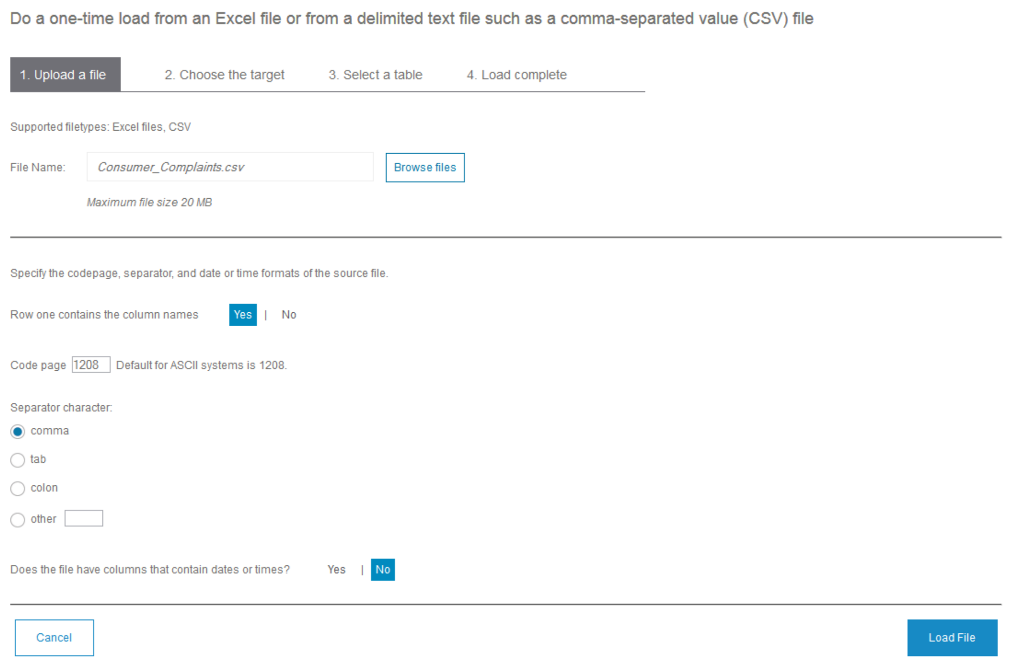
d. Import CSV data into dashDB
You can import CSV data into dashDB via the administration console.

The website http://data.gov maintains a CSV file containing all the current complaints filed by citizens to the Consumer Financial Protection Bureau. Download the file https://data.consumerfinance.gov/api/views/s6ew-h6mp/rows.csv?accessType=DOWNLOAD to your local machine. The default name is ***Consumer_Complaints.csv***

From the dashDB Administration tool landing page select Manage->Load Data.



Click on Browse to select the ***Consumer_Complaints.csv*** **file** and then click on Load File:


Do a one-time load from an Excel file or from a delimited text file such as a comma-separated value (CSV) file

Supported filetypes: Excel files, CSV

File Name: *Consumer_Complaints.csv* Browse files

Maximum file size 20 MB

Specify the codepage, separator, and date or time formats of the source file.

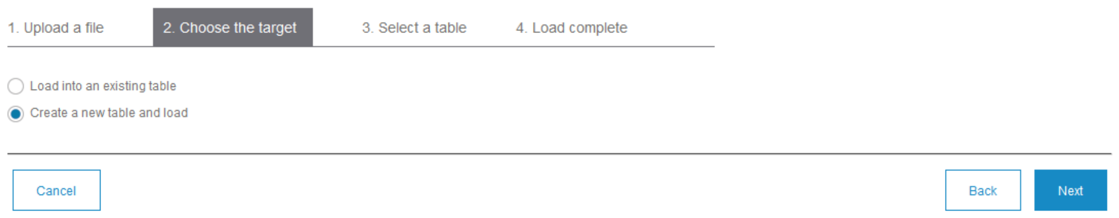
Row one contains the column names [✓ Yes] | No

Code page 1208 Default for ASCII systems is 1208.

Separator character:
● comma
○ tab
○ colon
○ other

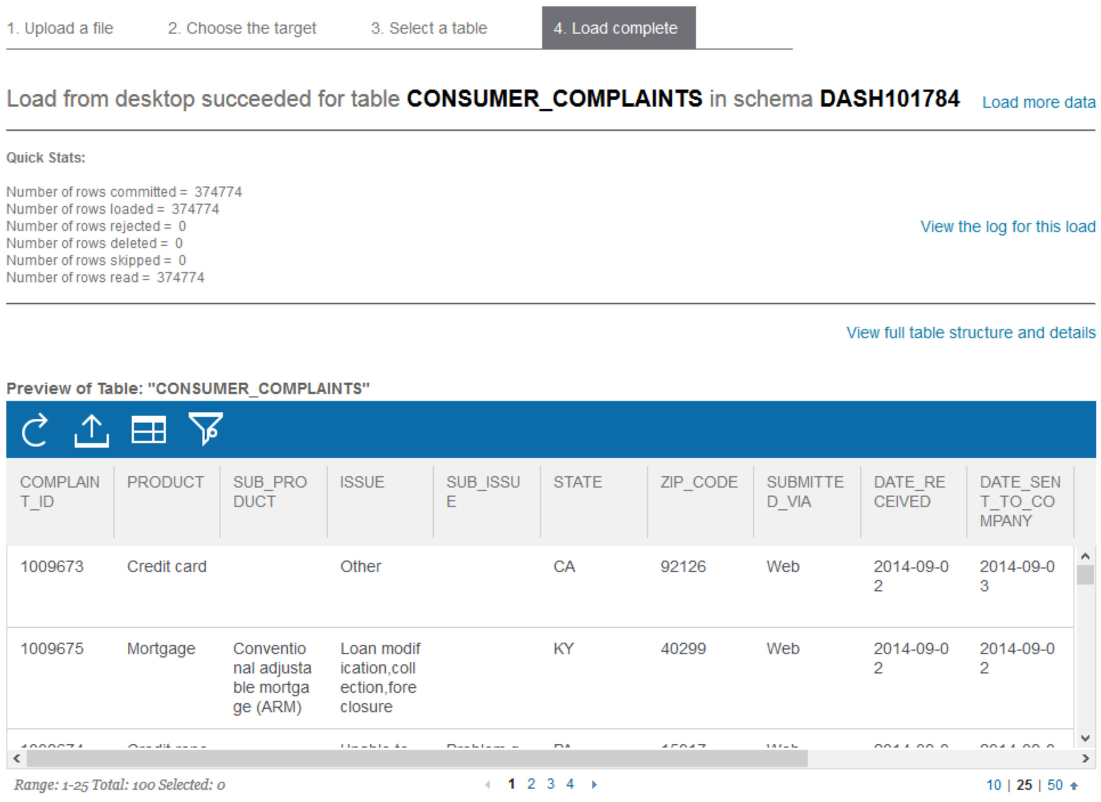
Does the file have columns that contain dates or times? Yes | [✓ No]

Click Next
Select Create a new table and load and click Next:



( ) Load into an existing table
(●) Create a new table and load

Click Finish to load the data. A preview of the data in the new table will appear as well as the number of rows imported.



Load from desktop succeeded for table **CONSUMER_COMPLAINTS** in schema **DASH101784**

Quick Stats:

Number of rows committed = 374774
Number of rows loaded = 374774
Number of rows rejected = 0
Number of rows deleted = 0
Number of rows skipped = 0
Number of rows read = 374774

View the log for this load
View full table structure and details

Preview of Table: "CONSUMER_COMPLAINTS"

COMPLAINT_ID	PRODUCT	SUB_PRODUCT	ISSUE	SUB_ISSUE	STATE	ZIP_CODE	SUBMITTED_VIA	DATE_RECEIVED	DATE_SENT_TO_COMPANY
1009673	Credit card		Other		CA	92126	Web	2014-09-02	2014-09-03
1009675	Mortgage	Conventional adjustable mortgage (ARM)	Loan modification,collection,foreclosure		KY	40299	Web	2014-09-02	2014-09-02

Range: 1-25 Total: 100 Selected: 0

d. Describe the IBM DataWorks service for Bluemix

1. Summarize capabilities of IBM DataWorks

The IBM DataWorks service in Bluemix allows you to identify relevant data, transform the data to suit your needs, and load it to a system for use.

IBM DataWorks is available from the DataWorks service Dashboard. In IBM DataWorks, you begin by finding the data that you want to work with from data sources like IBM SQL Database and IBM dashDB. You use metrics to better understand your data quality and identify areas to improve.

To improve the data quality, you work with a sample of the data and apply shaping actions such as sorting, filtering, and joining. You can apply the actions to the full data set and load the data to destinations such as Cloudant NoSQL DB. Actions are defined and processed through Activities.

Once an Activity is created, it may be used to set up recurring shaping and copy actions and the Activity control panel includes a display of the current run status.

IBM DataWorks supports the source and target combinations in the following table. All of the supported targets are compatible with each source.

Sources	**Targets**
Amazon Redshift	IBM Cloudant NoSQL DB
Apache Hive	IBM dashDB
Cloudera Impala	IBM Pure Data for Analytics
IBM Cloudant NoSQL DB	IBM Watson™ Analytics
IBM dashDB	IBM SQL Database
IBM DB2®	
IBM Informix®	
IBM Pure Data for Analytics	
IBM SQL Database	
Microsoft Azure	
Microsoft SQL Server	
MySQL	
Oracle	
Pivotal Greenplum	
PostgreSQL	
Salesforce.com	
Sybase	
Sybase IQ	

2. Perform common tasks using DataWorks
   a. Data Loading – Filter and move data

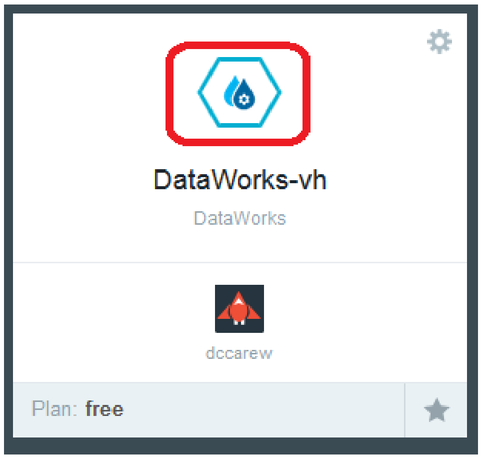
A set of steps providing a walk through the process of using DataWorks to move a subset of data in dashDB instance to an instance of SQL Database are provided to show key concepts of the DataWorks service.

1. Create an instance of dashDB

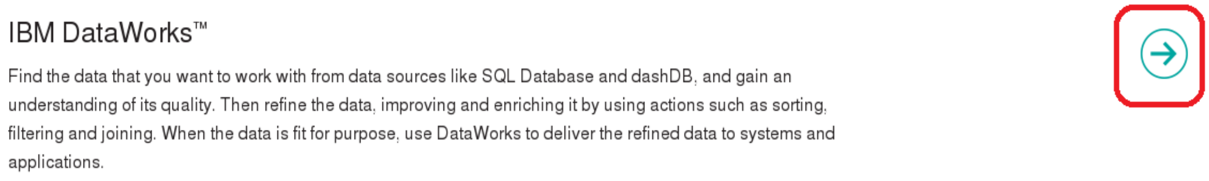
2. Create an instance of Cloudant NoSQL DB

3. Create an instance of IBM DataWorks

4. Once an instance of the DataWorks service has been created in IBM Bluemix you can start using DataWorks. Launch DataWorks by clicking the hexagon for your instance of the DataWorks by clicking on its hexagon from the Bluemix Dashboard:



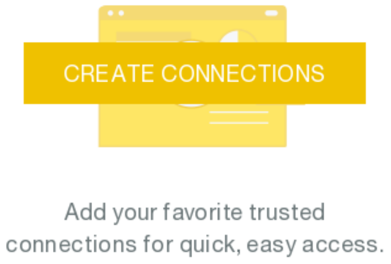
5. Click on the arrow when the service's landing page appears:



IBM DataWorks™

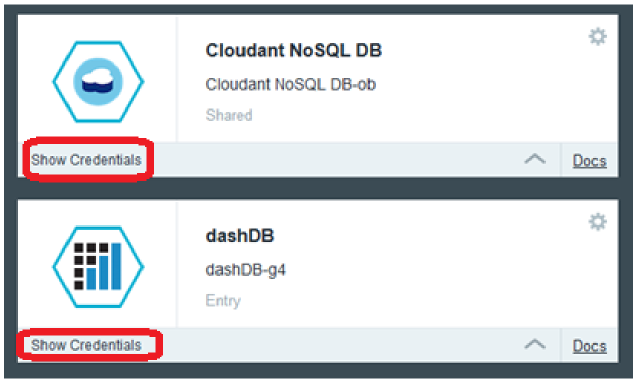
Find the data that you want to work with from data sources like SQL Database and dashDB, and gain an understanding of its quality. Then refine the data, improving and enriching it by using actions such as sorting, filtering and joining. When the data is fit for purpose, use DataWorks to deliver the refined data to systems and applications.

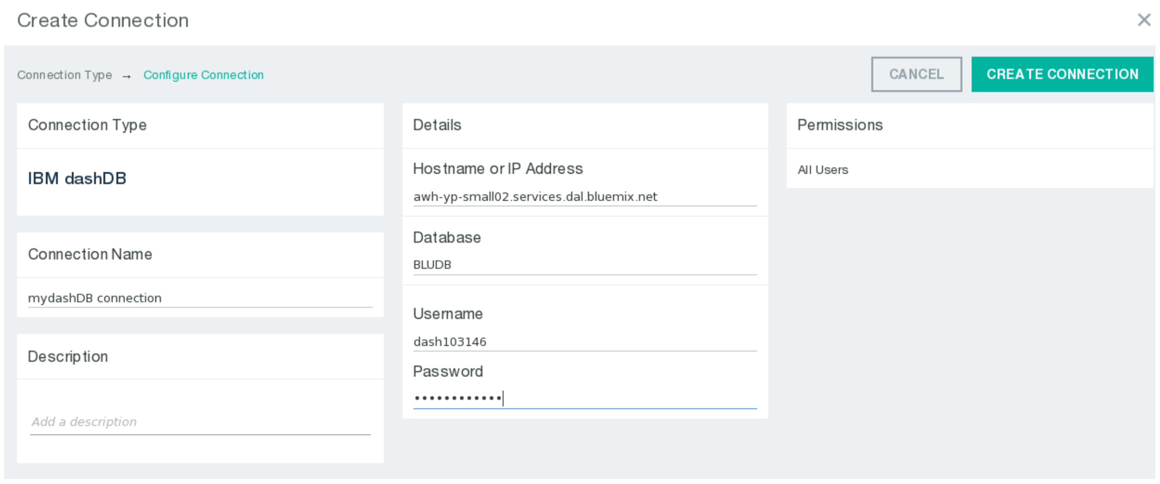
6. Click on Create Connections


CREATE CONNECTIONS
Add your favorite trusted connections for quick, easy access.

7. Click on the dashDB icon and then click on Add Connection.

8. Give the connection a name and description and use the credential information from your dashDB's instance credentials (note you can get this by clicking on the Show credentials link for the instance).




Create Connection

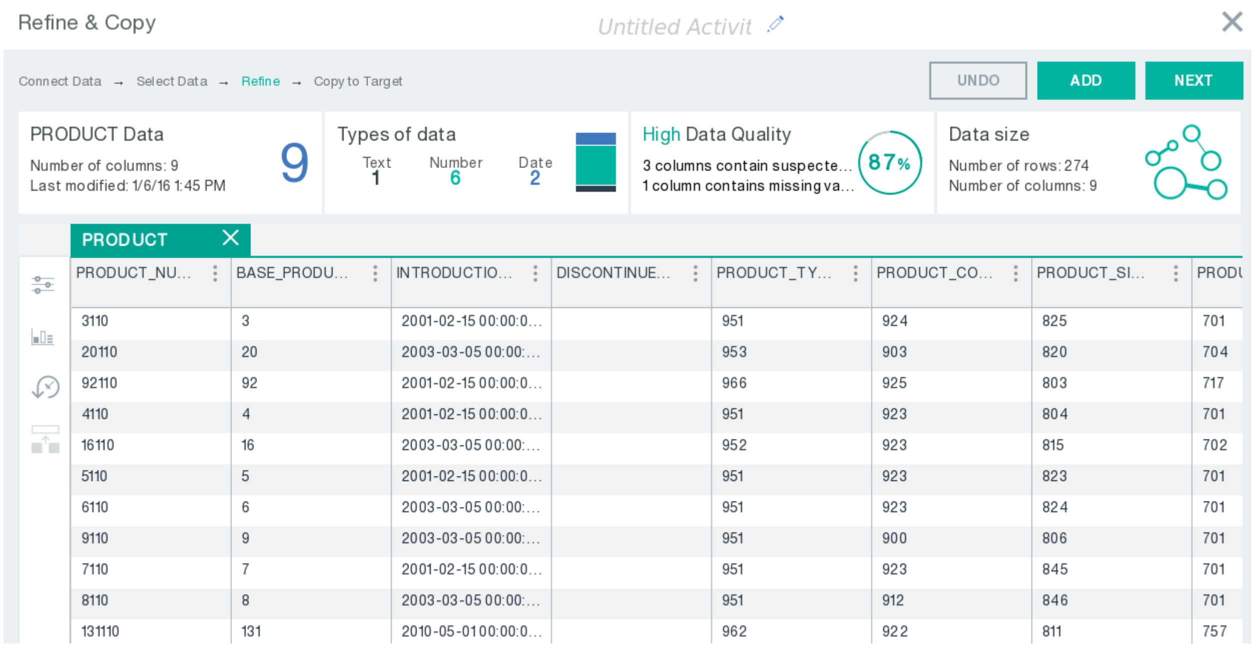
Connection Type → Configure Connection

CANCEL | CREATE CONNECTION

Connection Type	IBM dashDB
Connection Name	mydashDB connection
Description	Add a description
Hostname or IP Address	awh-yp-small02.services.dal.bluemix.net
Database	BLUDB
Username	dash103146
Password	••••••••••••
Permissions	All Users

When you’re done the dialog should like something like this:

9. Click on Create Connection
10. Repeat steps 7-8 to Add another connection to Cloudant NoSQL DB
11. Select Refine & Copy from the tasks list
12. Select the dashDB connection as the source
13. Select GOSALES from the Schemas column and Product from the Tables and Views column.
14. Select Product_type_code and then click Refine.


Refine & Copy

Untitled Activit

Connect Data → Select Data → Refine → Copy to Target

UNDO | ADD | NEXT

PRODUCT Data
Number of columns: 9
Last modified: 1/6/16 1:45 PM
9

Types of data
Text 1
Number 6
Date 2

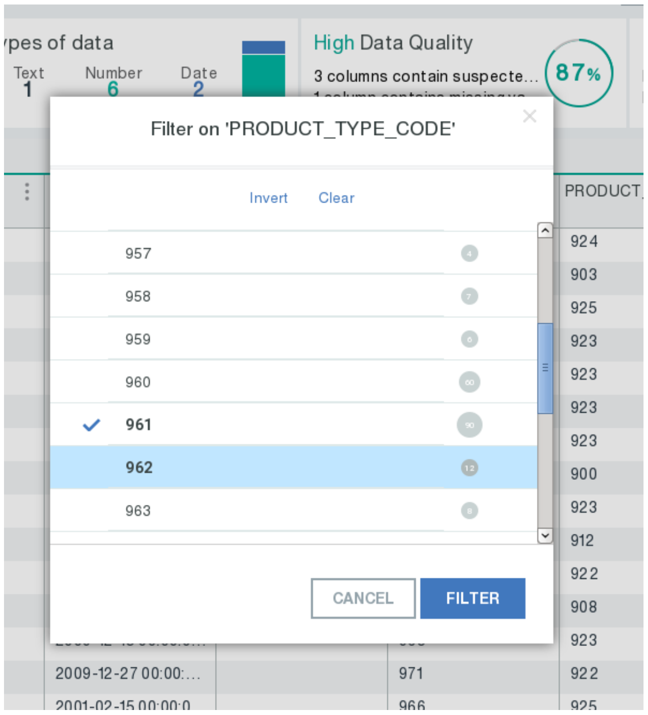
High Data Quality
3 columns contain suspecte…
1 column contains missing va…
87%

Data size
Number of rows: 274
Number of columns: 9

PRODUCT

PRODUCT_NU…	BASE_PRODU…	INTRODUCTIO…	DISCONTINUE…	PRODUCT_TY…	PRODUCT_CO…	PRODUCT_SI…	PRODU
3110	3	2001-02-15 00:00:0…		951	924	825	701
20110	20	2003-03-05 00:00:…		953	903	820	704
92110	92	2001-02-15 00:00:0…		966	925	803	717
4110	4	2001-02-15 00:00:0…		951	923	804	701
16110	16	2003-03-05 00:00:…		952	923	815	702
5110	5	2001-02-15 00:00:0…		951	923	823	701
6110	6	2003-03-05 00:00:…		951	923	824	701
9110	9	2003-03-05 00:00:…		951	900	806	701
7110	7	2001-02-15 00:00:0…		951	923	845	701
8110	8	2003-03-05 00:00:…		951	912	846	701
131110	131	2010-05-01 00:00:0…		962	922	811	757

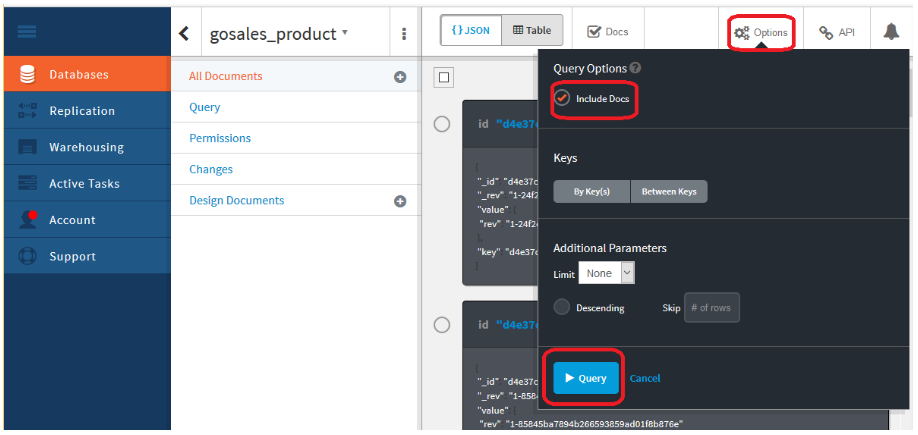
15. Click the dots on the Product_type_code column header and select Filter.
16. Scroll down and select the row with 961 from the list



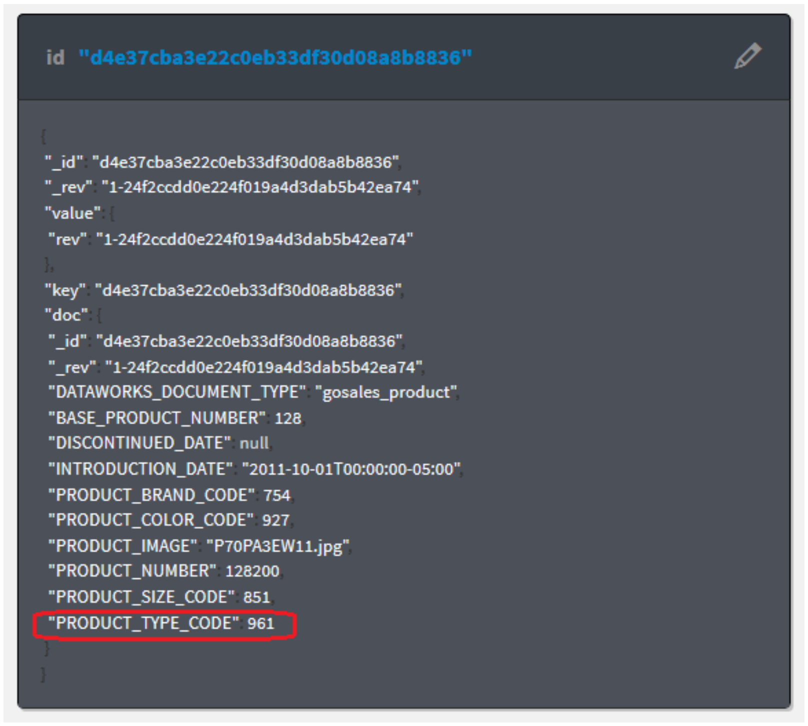
17. Click on Filter and then click Next.

18. Select the Cloudant NoSQL DB connection as the target
19. Take the default to append data
20. Assign a name to the Activity at the top and click Run.

21. Verify the data is in Cloudant NoSQL DB by launching the service console
22. After opening the console, click on the gosales_product database and then select Options and toggle to Include Docs



Verify that the PRODUCT_TYPE_CODE value is 961 for the documents (the value that you specified in the filter).



id "d4e37cba3e22c0eb33df30d08a8b8836"

```
{
"_id": "d4e37cba3e22c0eb33df30d08a8b8836",
"_rev": "1-24f2ccdd0e224f019a4d3dab5b42ea74",
"value": {
 "rev": "1-24f2ccdd0e224f019a4d3dab5b42ea74"
},
"key": "d4e37cba3e22c0eb33df30d08a8b8836",
"doc": {
 "_id": "d4e37cba3e22c0eb33df30d08a8b8836",
 "_rev": "1-24f2ccdd0e224f019a4d3dab5b42ea74",
 "DATAWORKS_DOCUMENT_TYPE": "gosales_product",
 "BASE_PRODUCT_NUMBER": 128,
 "DISCONTINUED_DATE": null,
 "INTRODUCTION_DATE": "2011-10-01T00:00:00-05:00",
 "PRODUCT_BRAND_CODE": 754,
 "PRODUCT_COLOR_CODE": 927,
 "PRODUCT_IMAGE": "P70PA3EW11.jpg",
 "PRODUCT_NUMBER": 128200,
 "PRODUCT_SIZE_CODE": 851,
 "PRODUCT_TYPE_CODE": 961
 }
}
```

Next Steps

1. Take the IBM Cloud Platform Application Development V1 assessment test using the promotion code *a285w15* for $15 ($15 USD savings).

2. If you pass the assessment exam, visit pearsonvue.com/ibm to schedule your test delivery session. Use the promotion code *c285w20* to receive 20% off.

3. If you failed the assessment exam, review how you did by section. Focus attention on the sections where you need improvement. Keep in mind that you can take the assessment exam as many times as you would like ($15 per exam), however, you will still receive the same questions only in a different order.