

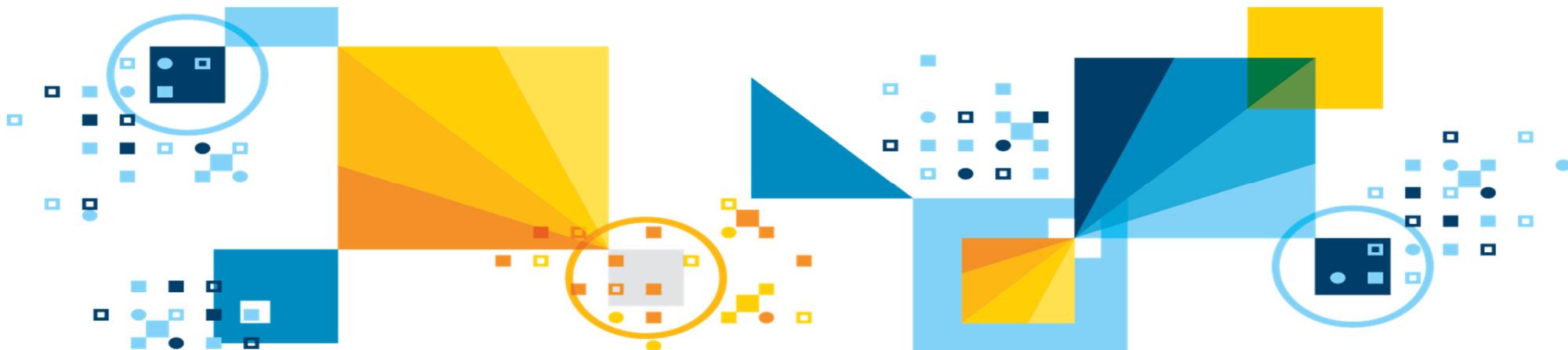
# Watson Studio Local Training

**Feb. 20, 2019**

# Agenda

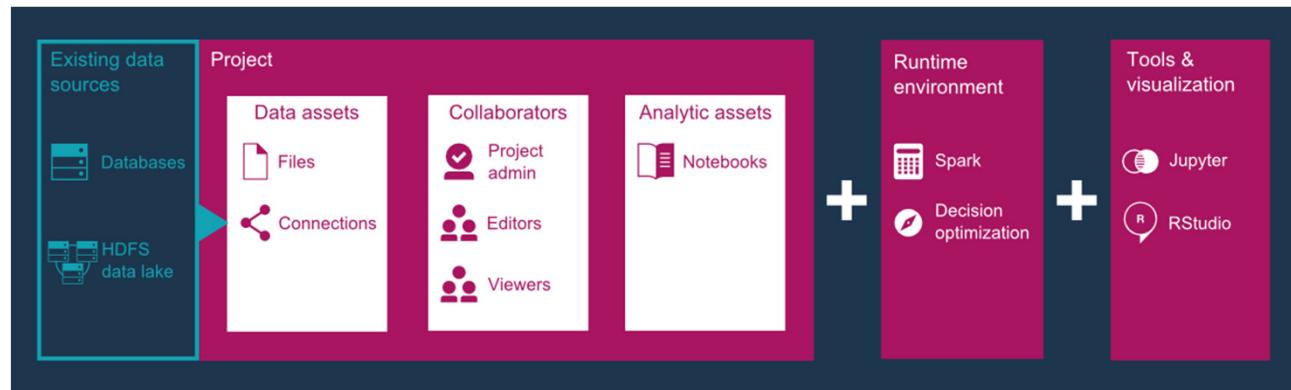
- **Day 1:**
  - Overview
  - Architecture
  - Organize Assets
  - Load and Access Data
  - Analyze Data
  - Model Lifecycle
- **Day 2:**
  - Model Management & Deployment
  - Kubernetes 101
  - WSL Security
  - User Management
  - Nodes Management
  - Services Management
  - Pods Management
- **Day 3:**
  - Monitor Nodes
  - Alerts
  - Package Management
  - Hadoop Integration
  - Image Management
  - Best Practice
  - Health Check
  - Troubleshoot

# Watson Studio Local (WSL) Training Overview



## Overview

- IBM Watson Studio Local (WSL) is an out-of-the-box on premises enterprise solution for data scientists and data engineers.
- It offers a suite of data science tools, such as RStudio, Spark, Jupyter, and Zeppelin notebooks, that are integrated with proprietary IBM technologies.

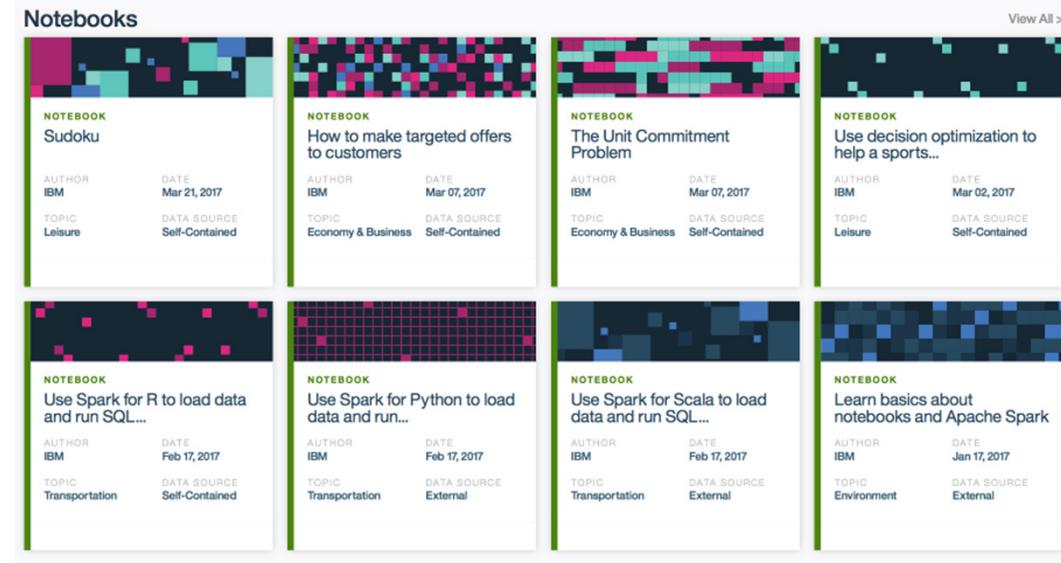


## Overview: Community

- Jump start your analysis with a sample notebook in R, Python, or Scala, copy content into a Jupyter notebook, or work in a fully embedded RStudio environment

### Community

**Notebooks**

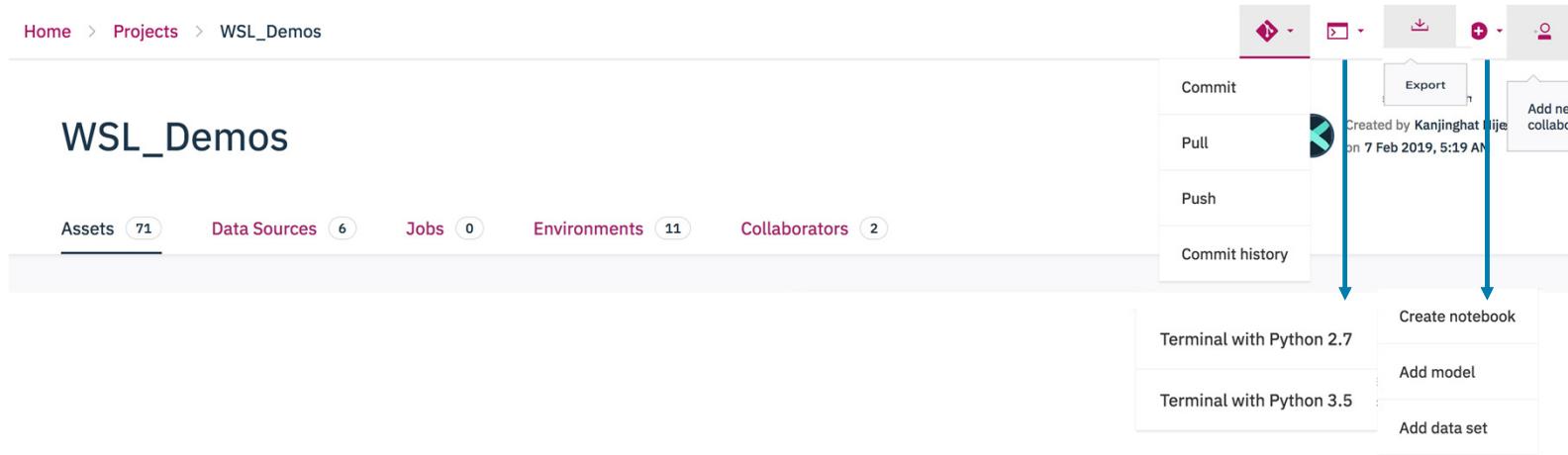


NOTEBOOK	AUTHOR	DATE	TOPIC	DATA SOURCE
Sudoku	IBM	Mar 21, 2017	Leisure	Self-Contained
How to make targeted offers to customers	IBM	Mar 07, 2017	Economy & Business	Self-Contained
The Unit Commitment Problem	IBM	Mar 07, 2017	Economy & Business	Self-Contained
Use decision optimization to help a sports...	IBM	Mar 02, 2017	Leisure	Self-Contained
Use Spark for R to load data and run SQL...	IBM	Feb 17, 2017	Transportation	Self-Contained
Use Spark for Python to load data and run...	IBM	Feb 17, 2017	Transportation	External
Use Spark for Scala to load data and run SQL...	IBM	Feb 17, 2017	Transportation	External
Learn basics about notebooks and Apache Spark	IBM	Jan 17, 2017	Environment	External

[View All >](#)

## Overview: Projects

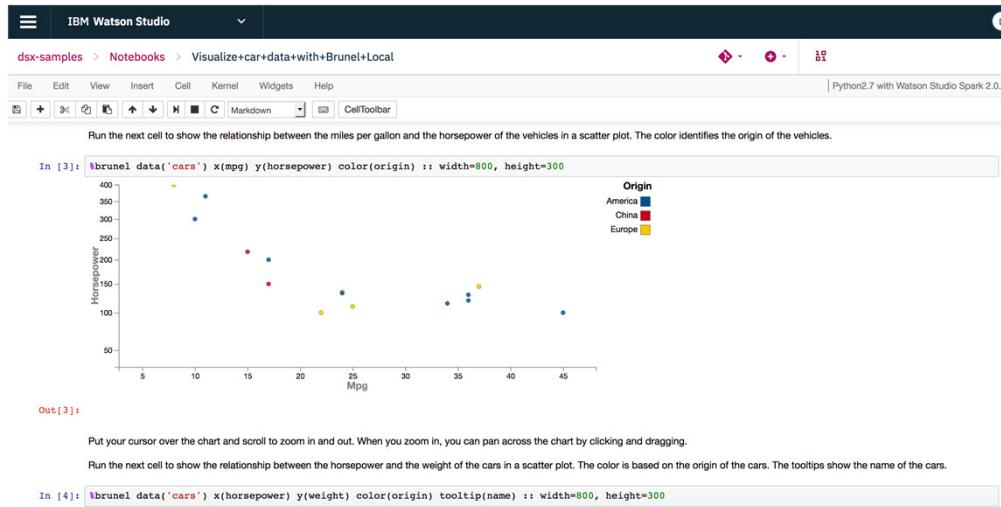
- Projects are your home base for collaboration.
- You can use projects to collect assets, such as notebooks, models, data sources, and remote and local data sets in one place
- Projects are a great way to work with a team: share a set of assets and then build new models and analysis together



The screenshot shows the 'WSL\_Demos' project page. At the top, there's a navigation bar with 'Home > Projects > WSL\_Demos'. Below the navigation is a header with the project name 'WSL\_Demos' and tabs for 'Assets' (71), 'Data Sources' (6), 'Jobs' (0), 'Environments' (11), and 'Collaborators' (2). On the right side, there's a toolbar with icons for Commit, Pull, Push, and Commit history, along with an 'Export' button and an 'Add new collaborator' button. A tooltip for the 'Commit' icon indicates it was created by Kanjinghat Nijje on 7 Feb 2019, 5:19 AM. Below the toolbar, there are sections for 'Create notebook', 'Terminal with Python 2.7', 'Add model', and 'Terminal with Python 3.5', each with an 'Add data set' link.

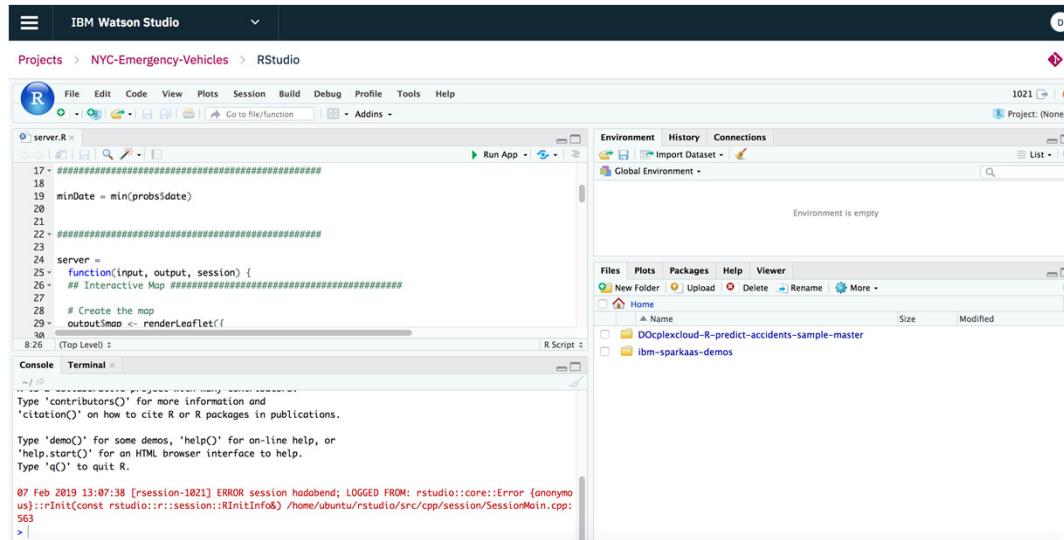
## Overview: Notebooks

- You can create a Jupyter notebook in R, Python, or Scala, or a Zeppelin notebook with a combination of R, Python, or Scala
- Start from scratch, import an existing notebook, or use one of the samples from the Community.
- When you open a new or existing notebook you can view and use connected data sources, see the comments and versions, and share your notebook with others.



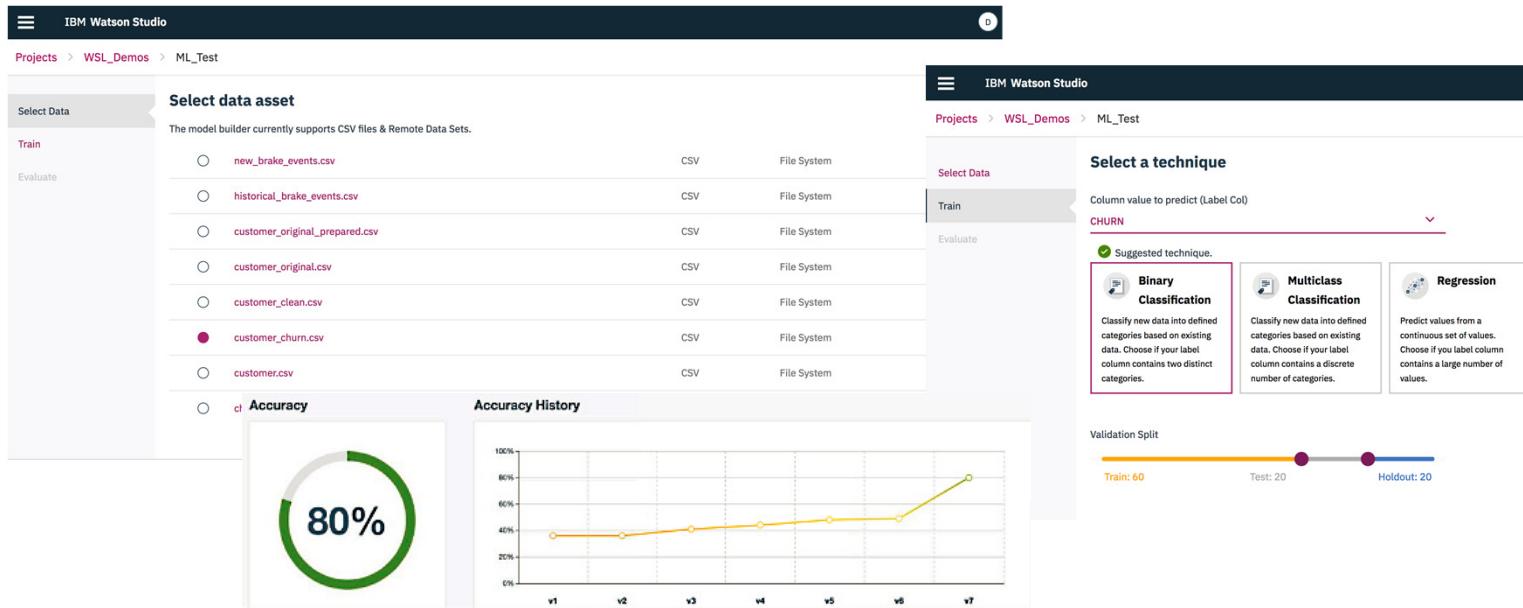
## Overview: R Studio

- R is a popular statistical analysis and machine-learning package that enables data management and includes tests, models, analyses, and graphics, and enables data management.
- RStudio, included in Watson Studio Local, provides an IDE for working with R.
- An RStudio session created in WSL includes 2 GB of storage and 5 GB of memory available for your use.



## Overview: Model Builder

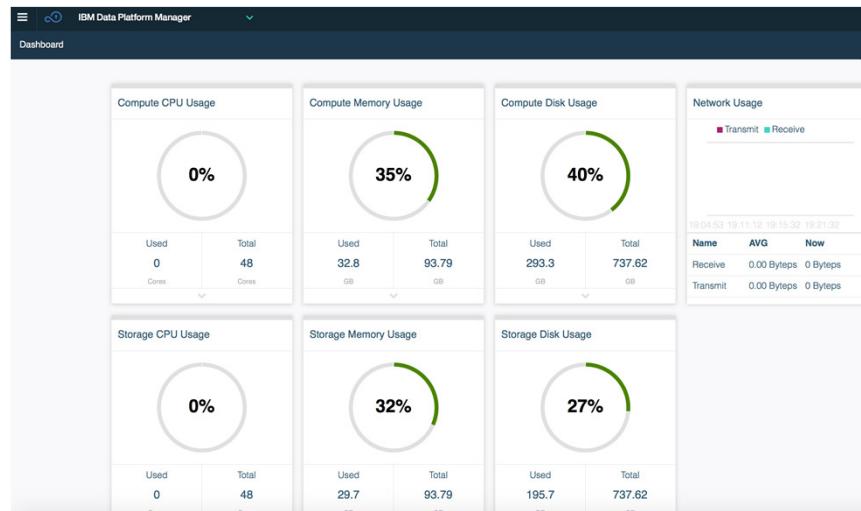
- The WSL client provides tools to help you create and train machine learning models that can analyze data assets and extract value from them.
- The Model Builder provides “clickers” a way to create machine learning models by guiding them through the process of preparing data, selecting an appropriate algorithm, and scoring the model



The screenshot shows the IBM Watson Studio Model Builder interface. On the left, under 'Select Data', a list of CSV files is shown, with 'customer\_churn.csv' selected. A large green circular progress bar at the bottom indicates an accuracy of 80%. On the right, under 'Select a technique', three options are listed: 'Binary Classification' (selected), 'Multiclass Classification', and 'Regression'. Below this, a 'Validation Split' slider shows 'Train: 60', 'Test: 20', and 'Holdout: 20'.

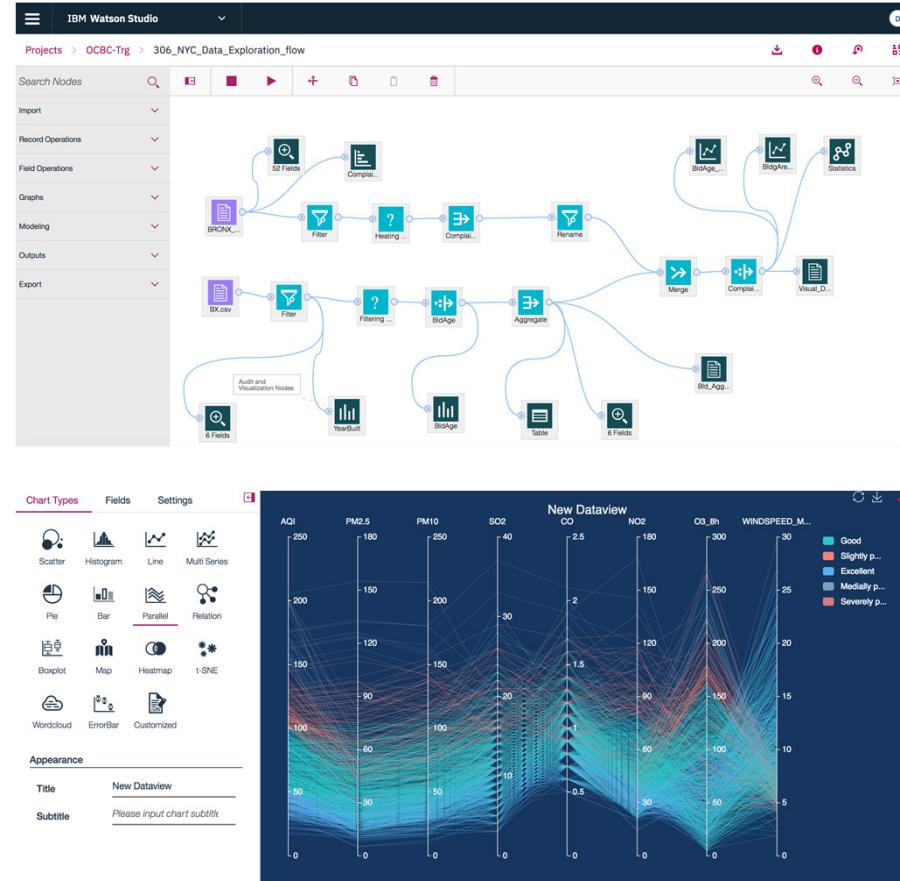
## Overview: Admin Console

- WSL manages your resources, and provides automated scaling and recovery options
- The administration console helps you manage and monitor your hardware, users, and services
- Security is built into WSL from data encryption at rest and in motion to user management.



# Overview: SPSS Modeler Add On

- Visual productivity tool for “clickers” instead of “coders”
  - Allows you to build machine learning models as a flow to conveniently prepare data, train the model, and evaluate it
  - Completely new interactive visualizations



# Overview: Cognos Dashboard

Simplifying **Data Exploration** to initiate analysis and share results

## Visualize

Represent analytic results as compelling interactive graphics

- Immediately view effective visualizations fitting the data selected.
- Choose your favorite graph or chart type from palette of options
- Apply Filters to individual or multiple visualizations.
- Customize appearance with individual settings or dashboard themes.

## Share

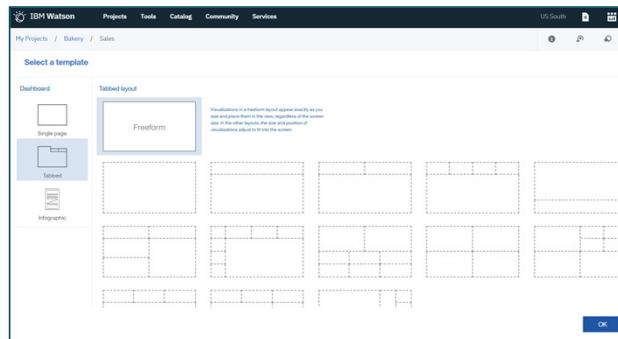
Deliver dashboards across the enterprise, providing insights where business decisions are made.

- Collaborate on dashboards with other project users.
- Share interactive dashboards through a public URL for use by others outside the platform.

## Discover

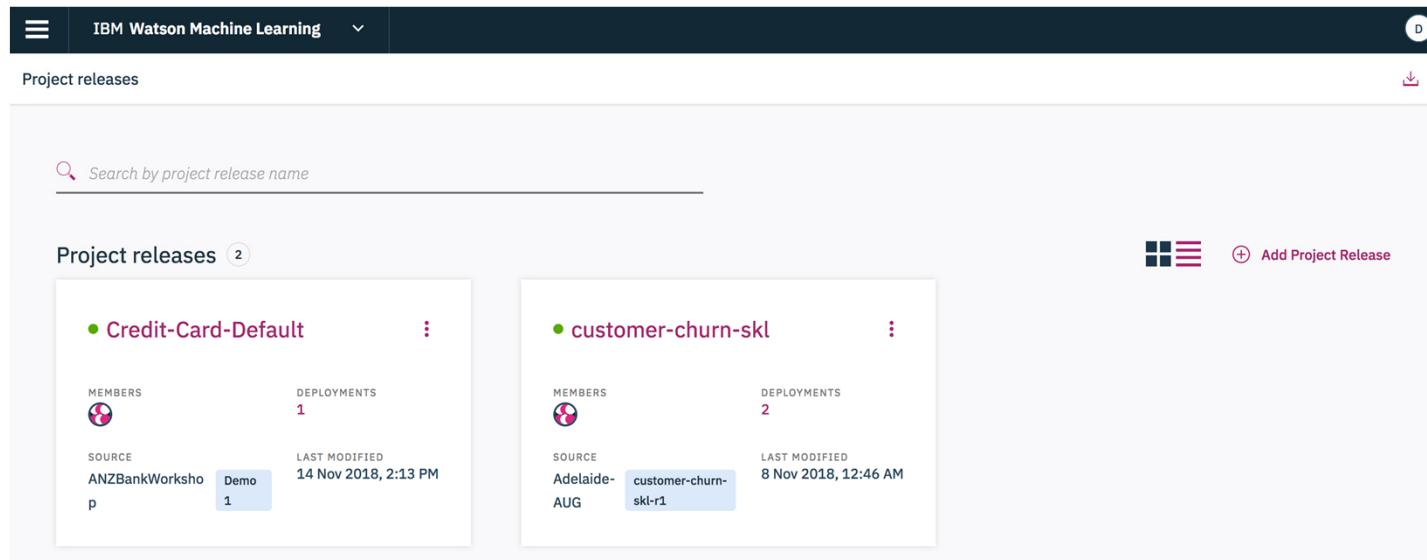
Utilize enterprise data assets with confidence through projects and the Watson Knowledge Catalog.

- Access data assets associated with your project
- Data sources may be local uploaded files or connections to relational sources.
- Search for data through the data catalog, then explore using dashboards.



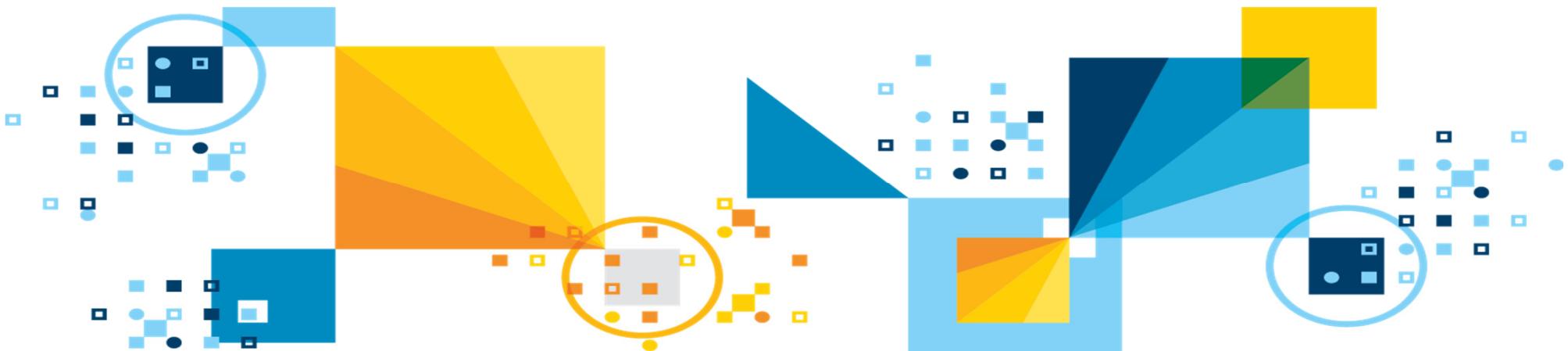
## Overview: Watson Machine Learning (WML)

- To expose a checkpoint of assets to outside users, a WSL administrator can create a project release and deploy the assets within it.
- A project release represents a project tag that can be launched as a production environment within WSL. The WSL administrator can monitor releases and deployments from WML.



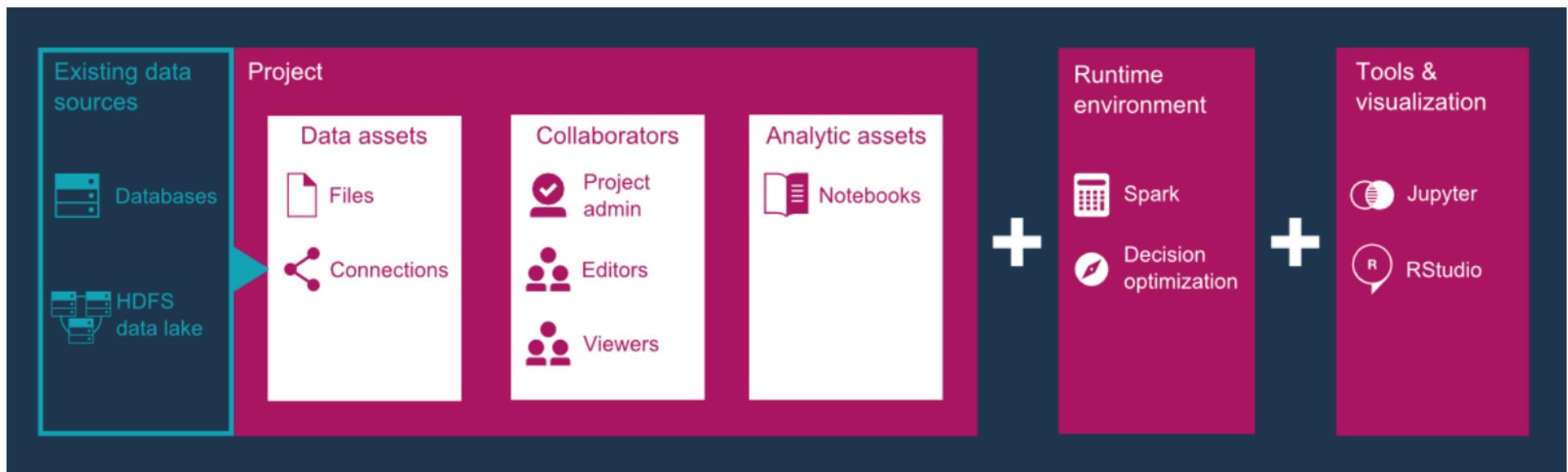
The screenshot shows the IBM Watson Machine Learning interface. At the top, there's a dark header bar with the IBM Analytics logo on the left and the WML logo on the right. Below the header is a navigation bar with a menu icon, the text "IBM Watson Machine Learning", and a dropdown arrow. On the far right of the navigation bar is a circular icon with a letter "D". The main content area has a light gray background. At the top left of this area, there's a search bar with a magnifying glass icon and the placeholder text "Search by project release name". To the right of the search bar is a small red download icon. Below the search bar, the text "Project releases" is displayed, followed by a count of "2". Underneath this, there are two cards, each representing a project release. The first card is for "Credit-Card-Default" and the second for "customer-churn-skl". Each card has a green circular icon with a white dot in the center. The "Credit-Card-Default" card shows "1 MEMBER" with a profile picture, "1 DEPLOYMENTS" with a blue button labeled "Demo", and "LAST MODIFIED 14 Nov 2018, 2:13 PM". The "customer-churn-skl" card shows "2 MEMBERS" with a profile picture, "2 DEPLOYMENTS" with a blue button labeled "customer-churn-skl-r1", and "LAST MODIFIED 8 Nov 2018, 12:46 AM". At the bottom right of the main content area, there are three icons: a blue square with four smaller squares inside, a red square with three horizontal lines inside, and a green square with a plus sign inside. Next to these icons is the text "+ Add Project Release".

# Watson Studio Local (WSL) Training Architecture

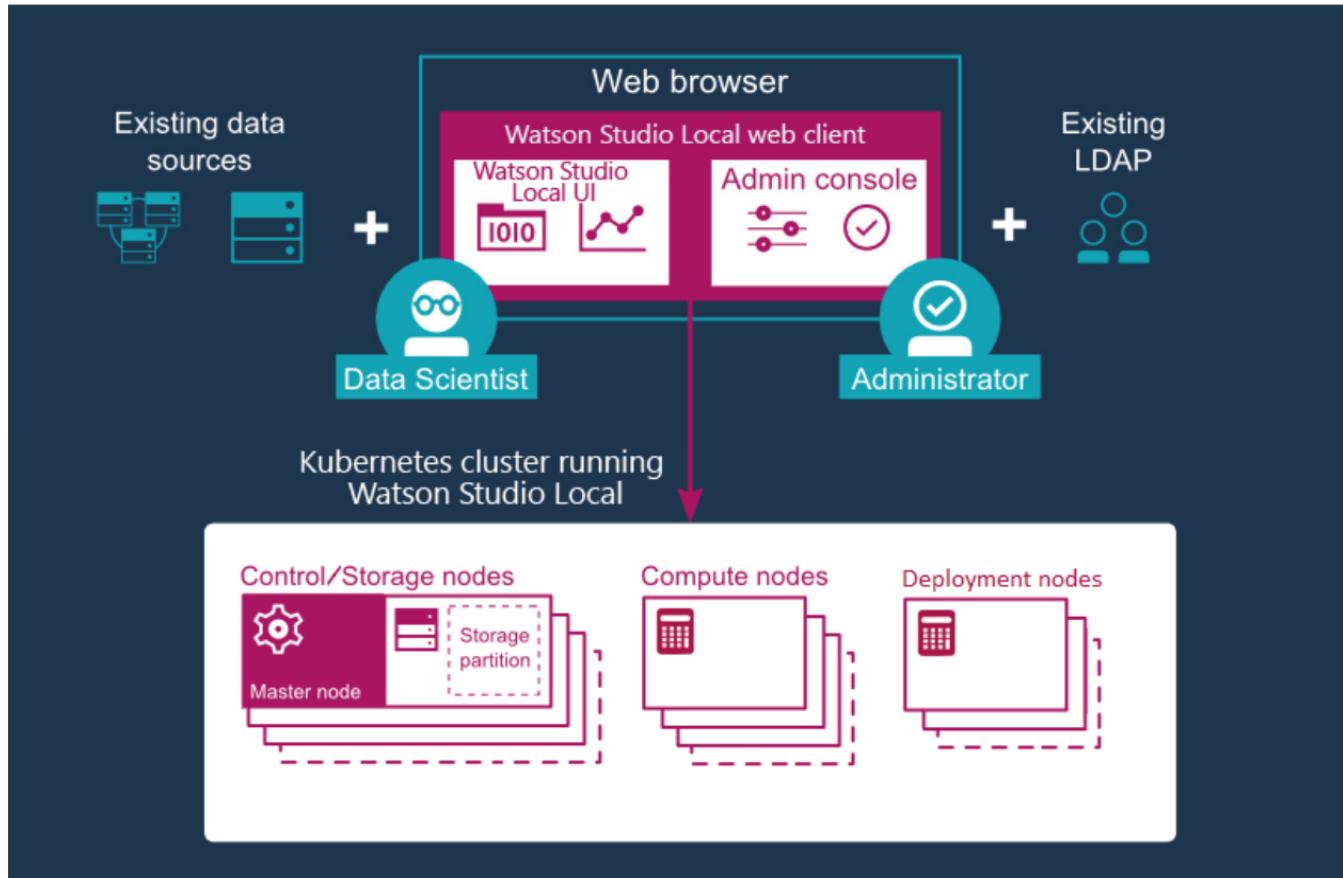


## IBM Watson Studio Local (WSL) – Overview

- IBM Watson Studio Local is an out-of-the-box enterprise solution for data scientists and data engineers.
- It offers a suite of data science tools, such as RStudio, Spark, Jupyter, and Zeppelin notebooks, that are integrated with proprietary IBM technologies.

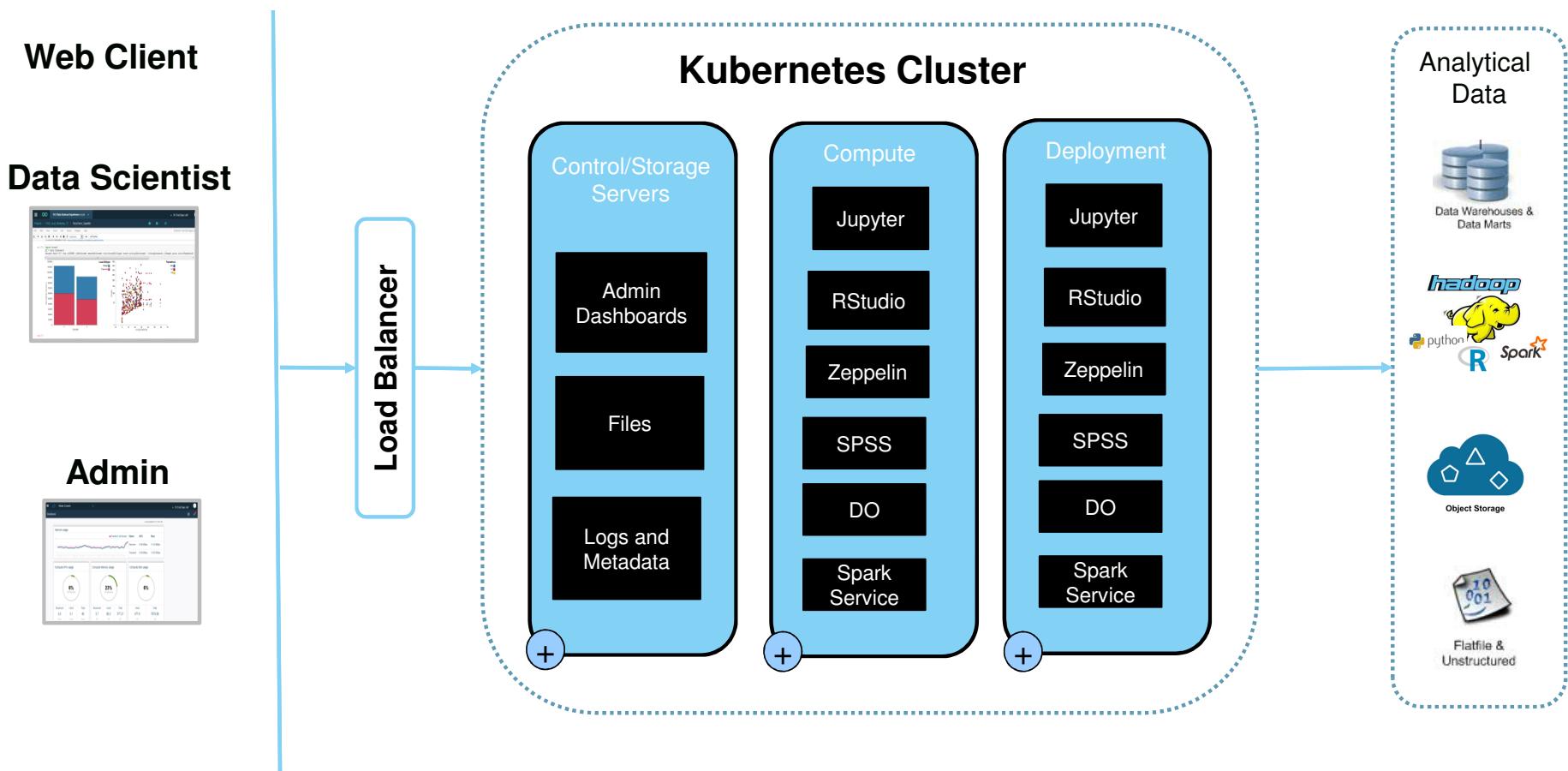


# IBM Watson Studio Local (WSL) – Architecture



1. HA is available by default. The deployments cannot be made HA for a single deployment node.
2. Vertical and horizontal scalability. Control/Compute and Deployment nodes can be added at any time.
3. LDAP authentication supported. Native authentication available.
4. Connect to data using JDBC, API or built-in connectors.
5. Supports Remote execution of Spark, Python, in R in a Hadoop cluster

# IBM Watson Studio Local (WSL) – Architecture



## IBM Watson Studio Local (WSL) – Cluster Configuration

- WSL cluster has several types of nodes (systems)
  - Compute: run development workload (Notebooks, RStudio, etc.)
  - Control/Storage: manage the system and store data/metadata
  - Deployment: dedicated to running production workload
- Cluster Configurations

Cluster Type	# Node breakdown	Notes
3 nodes	3 shared control / compute	Unable to deploy assets.
4 nodes	3 shared control / compute + 1 deploy	
5 nodes	3 shared control / compute + 2 deploy	
7 nodes	3 shared control + 3 compute + 1 deploy	
8 nodes	3 shared control + 3 compute + 2 deploy	
11 nodes	3 shared control + 6 compute + 2 deploy	

## IBM Watson Studio Local (WSL) – Kubernetes

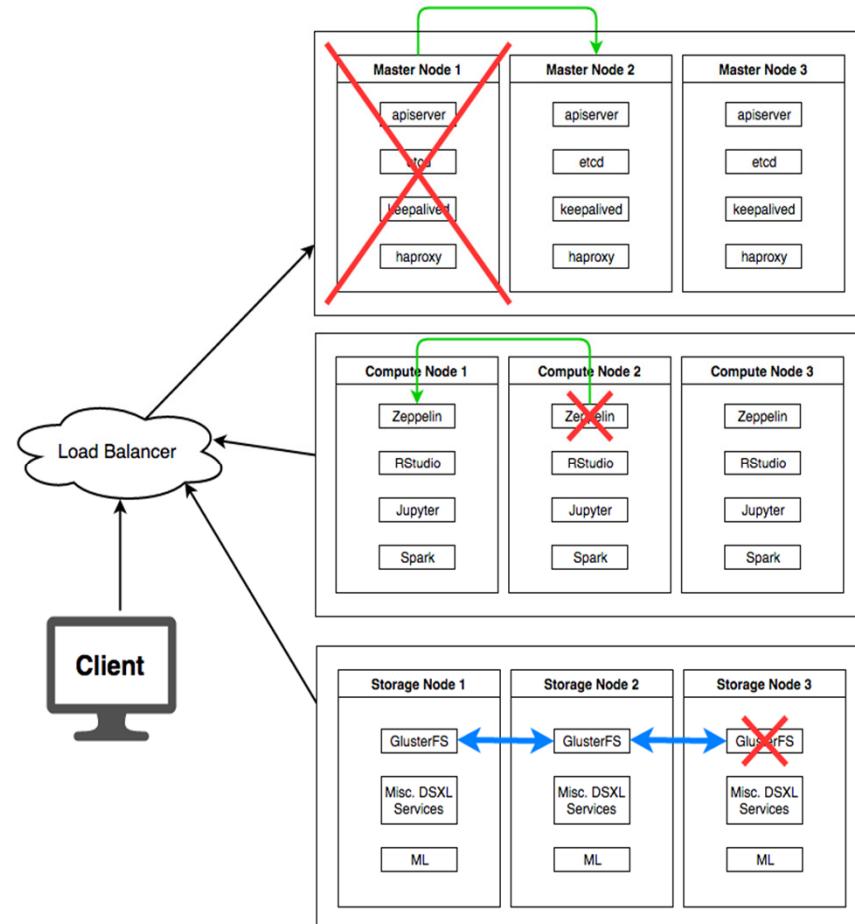
- WSL Local deploys into its own Kubernetes cluster or ICP (IBM Cloud Private)
  - Deployment into other clusters is not supported
- Kubernetes provides
  - High availability for services
  - Scalability
  - Monitor and display system status

## IBM Watson Studio Local (WSL) – High Availability

- Platform HA
  - By default, the development nodes (Control/Compute/Storage) are configured for HA
    - 3 nodes in 4-node configuration
    - 6 nodes in a 7-node configuration
  - For Deployment nodes, the customer can configure active/active or active/passive HA
    - Active/active HA: at least 2 Deployment nodes
    - Active/passive HA: 1 Deployment node and 1 cold standby

## IBM Watson Studio Local (WSL) – High Availability

- Platform HA
  - WSL can tolerate failure of
    - 1 node in a 3-node configuration
    - 3 nodes in a 6-node configuration



## IBM Watson Studio Local (WSL) – High Availability

### ■ Service HA

- For services that are deployed as pods, Kubernetes will monitor and redeploy services when they are down
- No session failover, which means that some services may be down for a few minutes
  - ✓ Any type of assets that requires a “session” (for example, batch scoring), will need to be restarted
- For online scoring, it’s possible to do deployment to multiple environments
  - ✓ Ensures uninterrupted online scoring

## IBM Watson Studio Local (WSL) – Kubernetes and Docker

- Kubernetes and Docker are open source technologies
  - Used for deploying applications with microservices architecture
- Docker provides a container for services
  - Once the service is “dockerized”, it can run in any environment managed by Docker
- Kubernetes is a platform for managing *containers* (including Docker)
  - Deployment
  - Scalability
  - Environment status

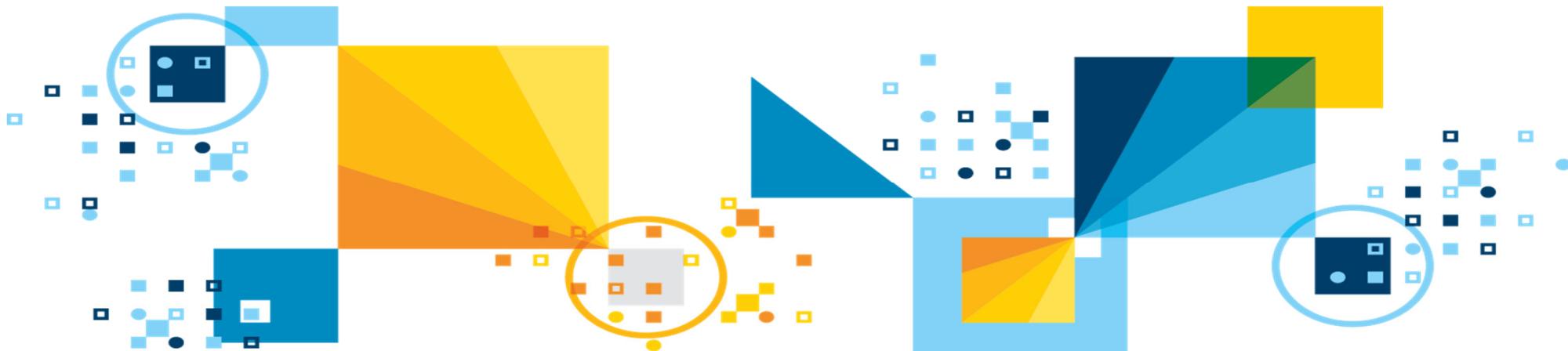


## Architecture - Overview

- WSL runs on a Kubernetes cluster of servers with the following components:
  - Control Nodes (Master)
    - Cluster management and HA, on 3 master nodes
    - Uses etcd as key value store that persists the cluster state and stores metadata about cluster service deployment and health
    - Uses Prometheus for monitoring and Elk for logging
  - Storage Nodes
    - Data stores and storage management
    - Uses GlusterFS for cluster storage and Cloudant DB as service meta database
    - Uses Redis as in-memory database and Elasticsearch DB for logs
  - Compute Nodes
    - Runs data science related services

# Watson Studio Local (WSL) Training

## Organize Assets



## Organize Assets in a Project

- A **Project** is a collection of assets that you use to achieve a particular data analysis goal.
- Your project assets can include:
  - Notebooks
  - RStudio files
  - Models
  - Data sets
  - Scripts
- **Restriction:** A project name, asset name, data source name, and remote data set name cannot contain any special characters.

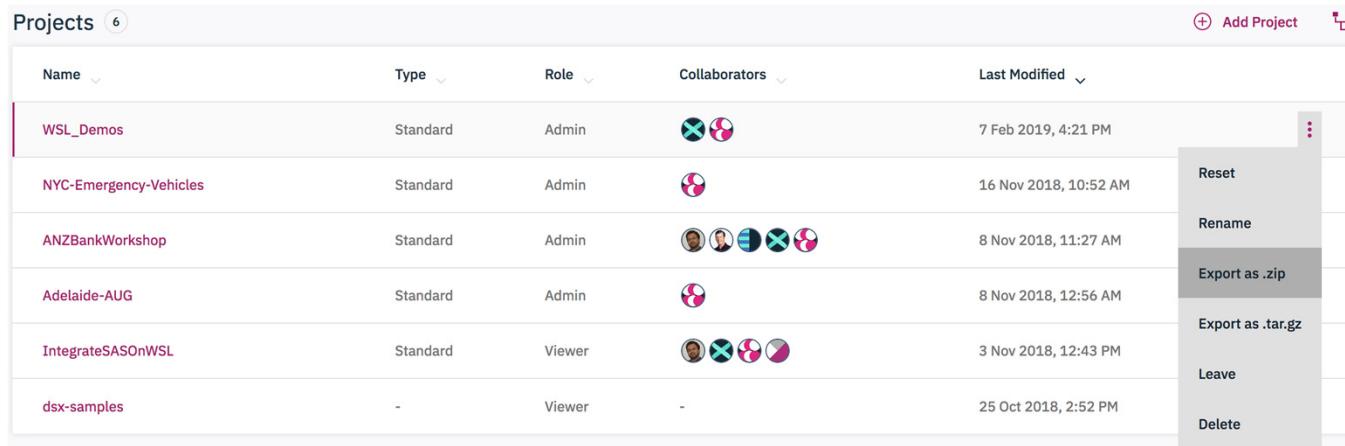
## Organize Assets in a Project - Create a Project

- To create a project, go to the Projects list and click **Add Project**.
- For a new blank project, click the **New** tab
  - To import a preexisting project from your local device, click the **From file tab** and upload the ZIP or TAR.GZ.
- Click **Create**
- Your new project opens and you can start adding collaborators and assets to it

Project type	Collaboration privileges	Master repository	Repository copy
Standard	Managed in DSX	Master repository exists in the DSX cluster file system	Each collaborator gets a copy
GitHub	Managed outside of DSX	Master repository exists in GitHub	Each user gets a copy when the project is imported from GitHub

## Organize Assets in a Project – Export a Project

- You can download a project as a ZIP or TAR.GZ file by clicking **Export as** button
  - Note that the environments in the project do not get exported

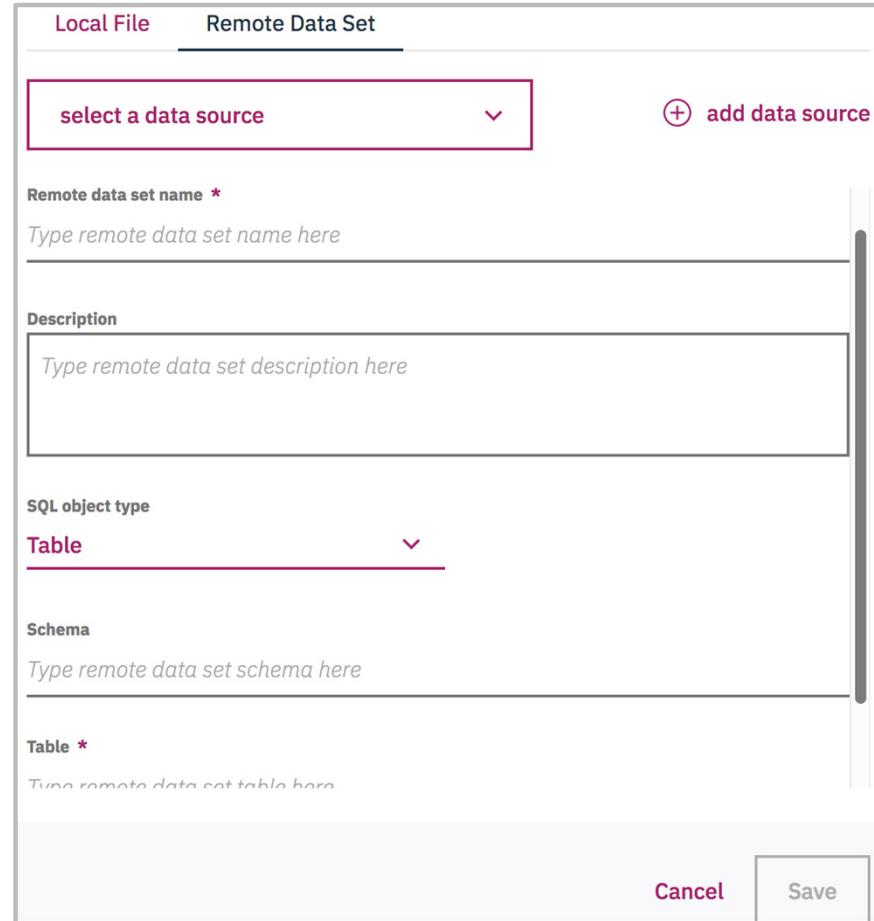


Name	Type	Role	Collaborators	Last Modified	Actions
WSL_Demos	Standard	Admin		7 Feb 2019, 4:21 PM	<ul style="list-style-type: none"><li>Reset</li><li>Rename</li><li>Export as .zip</li><li>Export as .tar.gz</li><li>Leave</li><li>Delete</li></ul>
NYC-Emergency-Vehicles	Standard	Admin		16 Nov 2018, 10:52 AM	
ANZBankWorkshop	Standard	Admin		8 Nov 2018, 11:27 AM	
Adelaide-AUG	Standard	Admin		8 Nov 2018, 12:56 AM	
IntegrateSASOnWSL	Standard	Viewer		3 Nov 2018, 12:43 PM	
dsx-samples	-	Viewer	-	25 Oct 2018, 2:52 PM	

- Restriction:** When you import this project, data source credentials will not be imported. For any data sources with credentials, you will need to open the imported project and specify the credentials for the data source again

## Organize Assets in a Project – Add Data Sources

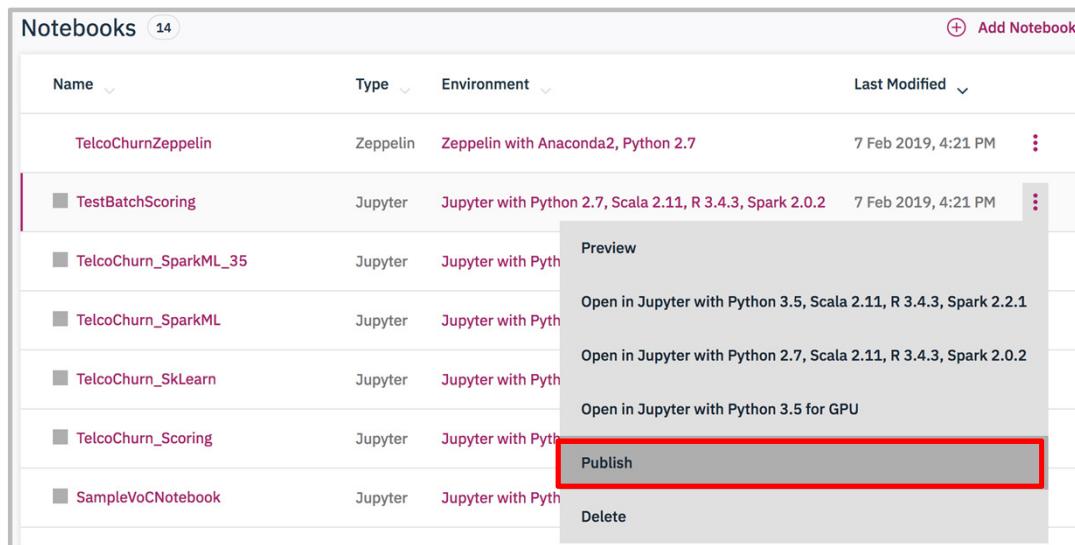
- A **Remote Data Source** provides a way for you to connect your data source directly to your project
  - For example, a database
- A **Remote Data Source** allows you to securely store information about your database and credentials
- To add a data source go to the **Data Source** page in your project
- You can create multiple **Remote Data Sets** within a Remote Data Source



The screenshot shows a dialog box for adding a data source. At the top, there are tabs for "Local File" and "Remote Data Set", with "Remote Data Set" selected. Below the tabs is a dropdown menu labeled "select a data source". To the right of the dropdown is a button with a plus sign and the text "add data source". The main area contains fields for "Remote data set name \*", "Description", "SQL object type", "Schema", and "Table \*". Each field has a placeholder text: "Type remote data set name here", "Type remote data set description here", "Table", "Type remote data set schema here", and "Type remote data set table here". At the bottom right of the dialog are "Cancel" and "Save" buttons.

## Organize Assets in a Project – Publish Assets

- A **Published** asset is a read-only copy of the asset
- To create one, go to the Assets page and click **Publish** next to the file
- The publish action creates a read-only snapshot of the current version of the asset, copies it to a published content directory in the user-home file system (if the file already exists, then it is versioned), and automatically generates a URL where the asset can be viewed



The screenshot shows a list of notebooks in the 'Notebooks' section. One notebook, 'TestBatchScoring', has a context menu open. The menu includes options like 'Preview', 'Open in Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1', 'Open in Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2', 'Open in Jupyter with Python 3.5 for GPU', and 'Publish'. The 'Publish' button is highlighted with a red box.

Name	Type	Environment	Last Modified
TelcoChurnZeppelin	Zeppelin	Zeppelin with Anaconda2, Python 2.7	7 Feb 2019, 4:21 PM
TestBatchScoring	Jupyter	Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2	7 Feb 2019, 4:21 PM
TelcoChurn_SparkML_35	Jupyter	Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1	
TelcoChurn_SparkML	Jupyter	Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2	
TelcoChurn_SkLearn	Jupyter	Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1	
TelcoChurn_Scoring	Jupyter	Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1	
SampleVoCNotebook	Jupyter	Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1	

## Organize Assets in a Project – Publish Assets

- The following assets can be published:
  - Jupyter notebook content
  - Local data sets
  - R Shiny web apps
- If you publish a Jupyter notebook, then the published copy is automatically converted to HTML:
  - You can either rerun the entire notebook (might take awhile) or publish it as-is
  - You can either include code cells in the published copy, or hide the code cells so that only the output appears.
- You can set the following content visibility permissions for the published asset:
  - All users with the URL (anyone outside of WSL can view it)
  - Any authenticated user (only signed in WSL users can view it)
  - Restricted to members in the selected project (only collaborators in the selected project can view it)

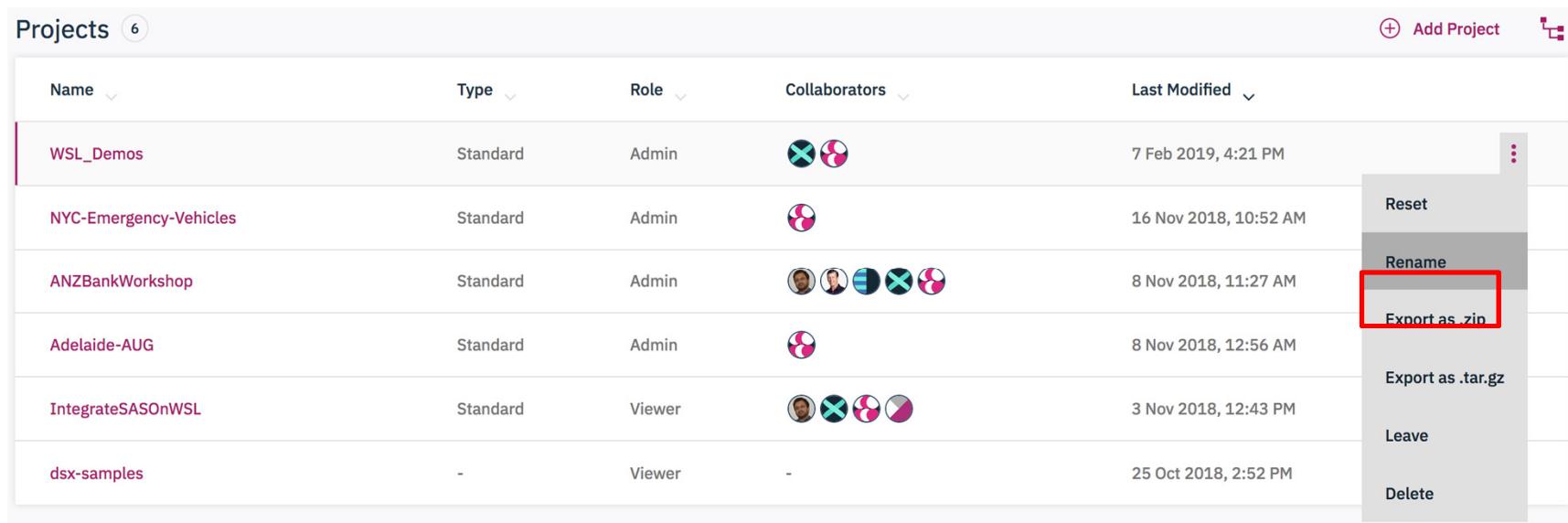
## Organize Assets in a Project – Publish Assets

- You can only publish to projects that you have Admin access to, and you cannot publish an asset to a project that was imported from GitHub (because these are not WSL managed projects)
- WSL then automatically generates a permalink URL to the published asset that you can copy
- Alternatively, WSL users can view the published asset in the Published Assets page
  - **Note:** The Published Assets page only shows assets that the signed in WSL user has permissions to
- To unpublish a file, you can go to the Published Assets page and click Unpublish next to it

Published Assets					
NAME	TYPE	DATE PUBLISHED	PUBLISHER	PUBLISHED FROM	VISIBLE TO
TelcoChurn-SKL-Pub	HTML	2-9-2019	Derek Chan	WSL_Demos	Anyone with the link <a href="#">Unpublish</a> 

## Organize Assets in a Project – Rename a Project

- If you have Admin permissions on a project, you can rename it by clicking Rename next to it.
- This renames the project for all of the collaborators, and automatically stops the Admin's runtimes active for that project. When the renaming completes, any access to Notebooks or RStudio will automatically start up the runtimes inside the context of the new project



Name	Type	Role	Collaborators	Last Modified	Actions
WSL_Demos	Standard	Admin		7 Feb 2019, 4:21 PM	<ul style="list-style-type: none"><li>Reset</li><li><b>Rename</b></li><li><b>Export as .zip</b></li><li>Export as .tar.gz</li><li>Leave</li><li>Delete</li></ul>
NYC-Emergency-Vehicles	Standard	Admin		16 Nov 2018, 10:52 AM	
ANZBankWorkshop	Standard	Admin		8 Nov 2018, 11:27 AM	
Adelaide-AUG	Standard	Admin		8 Nov 2018, 12:56 AM	
IntegrateSASOnWSL	Standard	Viewer		3 Nov 2018, 12:43 PM	
dsx-samples	-	Viewer	-	25 Oct 2018, 2:52 PM	

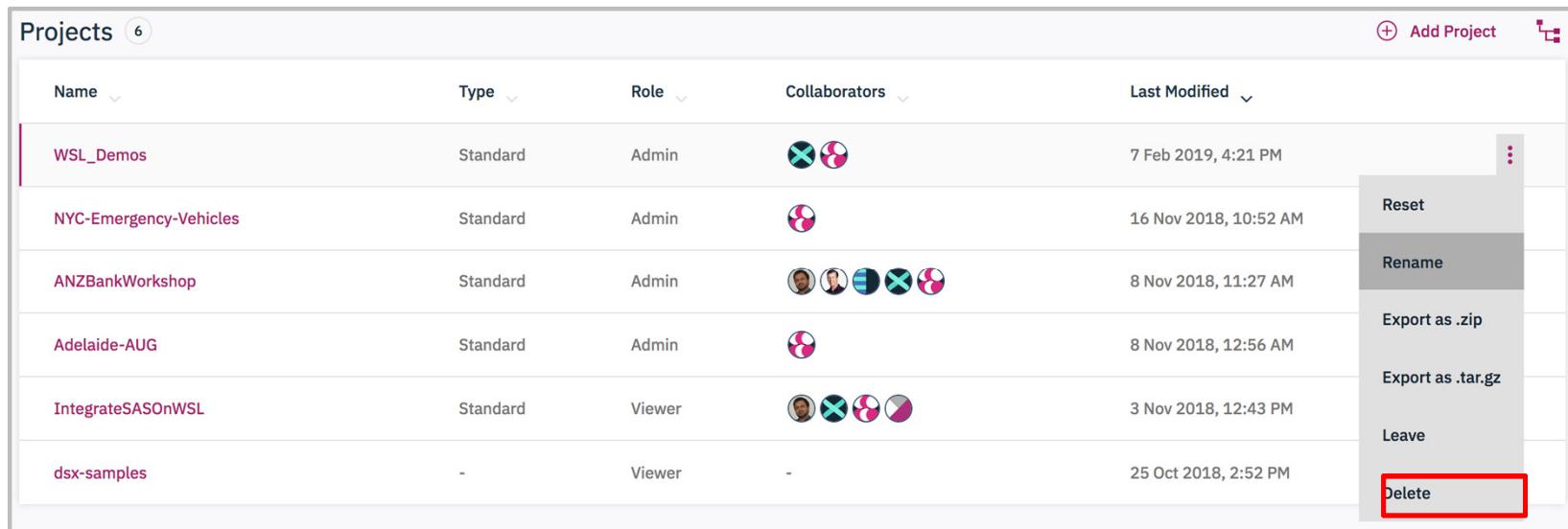
## Organize Assets in a Project – Rename a Project

- The Admin can also choose to manually start each in the Environments page. Because the containers are not stopped for the collaborators, each collaborator must stop the runtimes associated with the old project name in the All Active Environments page
- Any subsequent access to Notebooks and RStudio would automatically bring up the runtimes with the correct project name context, or the collaborator can go to “**All Active Environments**” and manually start the runtimes
- Also, collaborators should verify that assets like notebooks and scripts do not directly specify the project name in any of the paths

Runtimes 9									
Name	Runtime Type	User	Project	Job Name (Run Id)	Date Started	Requested Cpu (Cores)	Requested Gpus	Requested Memory (Gb)	Status
All Projects ▾ All Runtime Types ▾									
Watson Explorer	Environment	Derek Chan	Adelaide-AUG	—	8 Nov 2018, 9:03 AM	—	—	—	<span>●</span> <span>⋮</span>
Decision Optimization	Environment	Sumeet Parashar	dsx-samples	—	30 Jan 2019, 5:09 AM	2.0	—	1.0	<span>●</span>
Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2	Environment	Gianmaria Leo	dsx-samples	—	1 Feb 2019, 10:10 PM	—	—	—	<span>●</span>
Data Refinery	Environment	Kanjinghat Nijesh	WSL_Demos	—	7 Feb 2019, 4:27 PM	—	—	—	<span>●</span> <span>⋮</span>
Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1	Environment	Derek Chan	ANZBankWorkshop	—	7 Feb 2019, 8:57 PM	—	—	—	<span>●</span> <span>⋮</span>

## Organize Assets in a Project – Delete a Project

- If you have Admin permissions on a standard project, you can delete it by clicking Delete next to it
- This deletes the project for all of the collaborators, and deletes all assets (and the storage directories) associated with the project
- If necessary, a WSL Admin can manually recover deleted projects from the WSL system's recycle bin directory



Name	Type	Role	Collaborators	Last Modified	Actions
WSL_Demos	Standard	Admin		7 Feb 2019, 4:21 PM	<ul style="list-style-type: none"><li>⋮</li><li>Reset</li><li>Rename</li><li>Export as .zip</li><li>Export as .tar.gz</li><li>Leave</li><li><b>Delete</b></li></ul>
NYC-Emergency-Vehicles	Standard	Admin		16 Nov 2018, 10:52 AM	
ANZBankWorkshop	Standard	Admin		8 Nov 2018, 11:27 AM	
Adelaide-AUG	Standard	Admin		8 Nov 2018, 12:56 AM	
IntegrateSASOnWSL	Standard	Viewer		3 Nov 2018, 12:43 PM	
dsx-samples	-	Viewer	-	25 Oct 2018, 2:52 PM	

## Organize Assets in a Project – Delete a Project

- If an Admin deletes a GitHub project, then only the WSL copy of the project will be deleted (not the remote repository on GitHub).
- **Recommendation:** Before deleting a project, ensure the project's collaborators have stopped all running containers, saved their work, and exported a copy of the project if they need a backup of it, and stopped all runtimes for that project

## Manage Collaborators

- If you have Admin permissions for a project, you can add collaborators, change collaborator permissions, or remove collaborators from that project on its Collaborators page.
- The collaborator permissions are:
  - **Viewer:** Can view the project, accept changes, and commit changes to their own local copy of the project.
  - **Editor:** Can control project assets. Can accept, commit, and push changes.
  - **Admin:** Can control project assets, collaborators, and settings. Can accept, commit, and push changes.

Collaborators <span style="font-size: small;">2</span>					<span style="color: red;">+</span> Add Collaborator
Name	Role	Email	Date Added		
Derek Chan	Admin	derek.chan@au1.ibm.com	7 Feb 2019, 5:21 AM	<span style="color: red;">⋮</span>	
Kanjinghat Nijesh	Admin	knijesh@sg.ibm.com	7 Feb 2019, 5:19 AM	<span style="color: red;">⋮</span>	
	Editor				
	Viewer				

## Manage Collaborators

- From the Git Actions icon ( ) in the project action bar, an Admin or Editor can push and commit changes by clicking “Push”, followed by “Commit”
- A Viewer can add an asset but cannot push changes
- An Admin, Editor, or Viewer can pull changes from the Master Repository by clicking “Pull,” or reset the project to what is currently in the Master Repository by clicking “Reset” next to the project

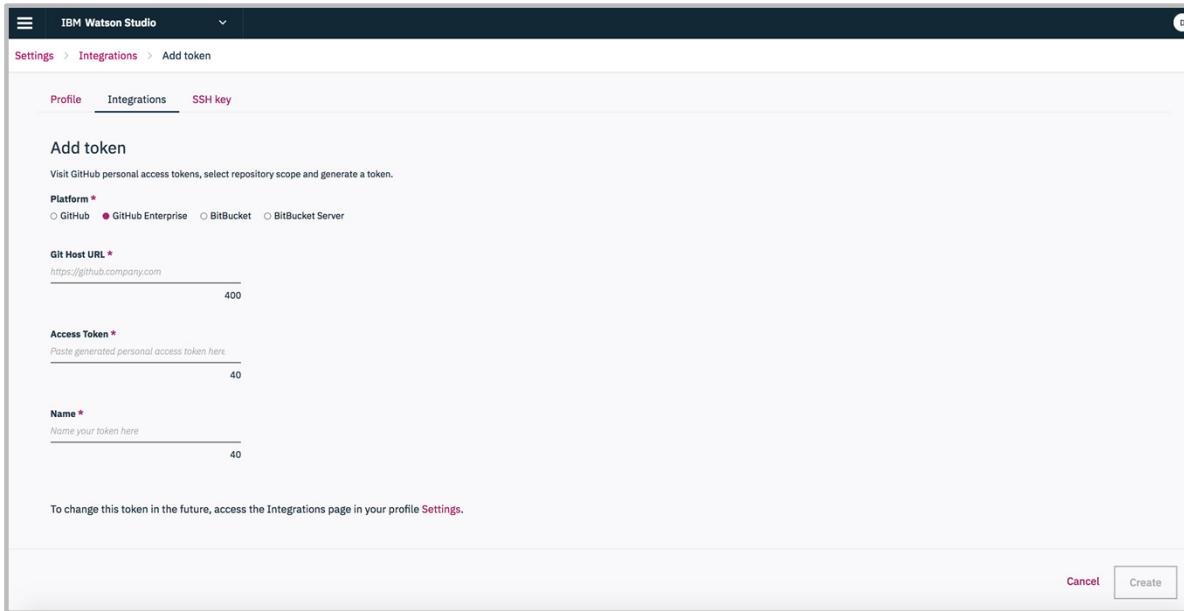
## Integration with Github

- To collaborate with external stakeholders and the data science community at large, you can import projects from a Git repository and push project assets and changes to a Git repository from WSL.
- Before importing and pushing assets on a Git repository, you must enable your WSL user account to access the Git repository. You enable access by creating a personal access token with the required access scope in the Git repository and linking the token to your WSL account.
- To create a personal access token:

GitHub Enterprise	<p>a.Go to your GitHub Enterprise deployment.</p> <p>b.Go to <b>Settings</b> and then select <b>Developer settings &gt; Personal access tokens</b></p> <p>c.Click <b>Generate a token</b>.</p> <p>d.On the <b>New personal access token</b> page, select the <b>repo</b> scope so that you can import assets from the repository and commit and push changes to the repository.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><table><tbody><tr><td><input checked="" type="checkbox"/> <b>repo</b></td><td>Full control of private repositories</td></tr><tr><td><input type="checkbox"/> <b>repo:status</b></td><td>Access commit status</td></tr><tr><td><input type="checkbox"/> <b>repo_deployment</b></td><td>Access deployment status</td></tr><tr><td><input type="checkbox"/> <b>public_repo</b></td><td>Access public repositories</td></tr><tr><td><input type="checkbox"/> <b>repo:invite</b></td><td>Access repository invitations</td></tr></tbody></table></div> <p>You can add additional scopes, but repo is the minimum required scope.</p> <p>e.Click <b>Generate token</b> and copy the token that is generated.</p>	<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories	<input type="checkbox"/> <b>repo:status</b>	Access commit status	<input type="checkbox"/> <b>repo_deployment</b>	Access deployment status	<input type="checkbox"/> <b>public_repo</b>	Access public repositories	<input type="checkbox"/> <b>repo:invite</b>	Access repository invitations
<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories										
<input type="checkbox"/> <b>repo:status</b>	Access commit status										
<input type="checkbox"/> <b>repo_deployment</b>	Access deployment status										
<input type="checkbox"/> <b>public_repo</b>	Access public repositories										
<input type="checkbox"/> <b>repo:invite</b>	Access repository invitations										

# Integration with Github

- To add your personal token:
  - Open your profile settings, select the Git Integrations tab, and click Add token.



The screenshot shows the 'Add token' dialog box within the IBM Watson Studio interface. The title bar reads 'IBM Watson Studio' with a dropdown arrow, followed by 'Settings > Integrations > Add token'. Below the title bar are three tabs: 'Profile' (selected), 'Integrations' (highlighted in red), and 'SSH key'. The main content area is titled 'Add token' with the sub-instruction: 'Visit GitHub personal access tokens, select repository scope and generate a token.' Under the 'Platform' section, 'GitHub' is selected (radio button is checked). The 'Git Host URL' field contains 'https://github.company.com'. The 'Access Token' field is empty and has the placeholder 'Paste generated personal access token here'. The 'Name' field is empty and has the placeholder 'Name your token here'. At the bottom of the dialog, a note says 'To change this token in the future, access the Integrations page in your profile [Settings](#)'. In the bottom right corner, there are 'Cancel' and 'Create' buttons.

- Fill out the fields as required, and be sure to paste the token that you copied into the Access token field and to give the token a name.
- Click Create.

## Manage Resources

- In WSL click the Menu icon ( ) and select **All Active Environments**
- This brings you to a screen where you can view all the current environments, workers, and services in the WSL cluster
- You can view and sort the users who created them, how much CPU and memory is reserved and whether they are currently running

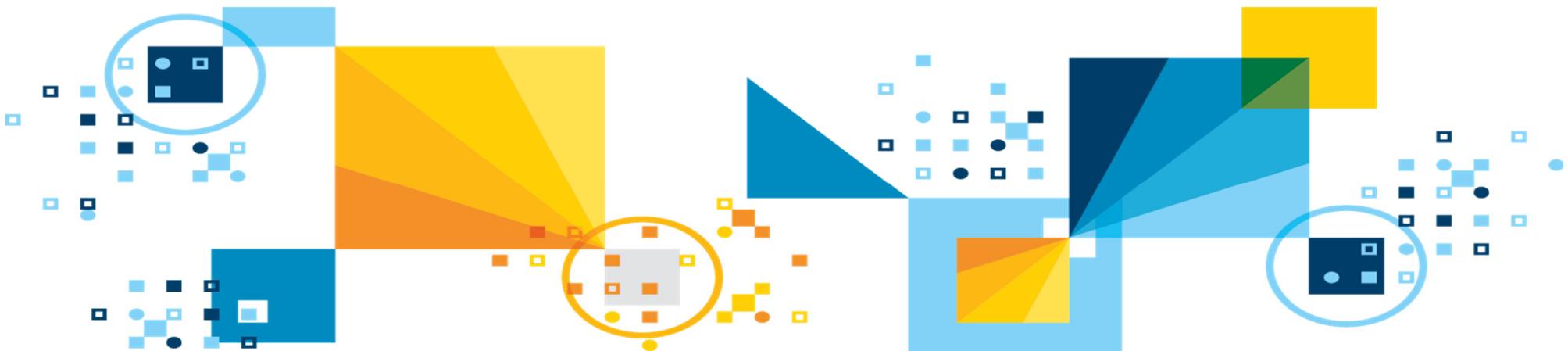
Runtimes 9										All Projects	All Runtime Types
Name	Runtime Type	User	Project	Job Name (Run Id)	Date Started	Requested Cpu (Cores)	Requested Gpus	Requested Memory (Gb)	Status		
Watson Explorer	Environment	Derek Chan	Adelaide-AUG	—	8 Nov 2018, 9:03 AM	—	—	—	<span style="color: green;">●</span>	<span style="color: red;">⋮</span>	
Decision Optimization	Environment	Sumeet Parashar	dsx-samples	—	30 Jan 2019, 5:09 AM	2.0	—	1.0	<span style="color: green;">●</span>		
Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2	Environment	Gianmaria Leo	dsx-samples	—	1 Feb 2019, 10:10 PM	—	—	—	<span style="color: green;">●</span>		
Data Refinery	Environment	Kanjinghat Nijesh	WSL_Demos	—	7 Feb 2019, 4:27 PM	—	—	—	<span style="color: green;">●</span>	<span style="color: red;">⋮</span>	
Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1	Environment	Derek Chan	ANZBankWorkshop	—	7 Feb 2019, 8:57 PM	—	—	—	<span style="color: green;">●</span>	<span style="color: red;">⋮</span>	

## Manage Resources

- To stop a running environment, worker, or service, click **Stop Now** next to it
- You will see the Status indicator change from green to red and then the environment should disappear from this page all together
  - You will have to refresh the page to see this
- That is the notification that your environment has stopped

Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2	Environment	Derek Chan	dsx-samples	—	8 Feb 2019, 12:00 AM	—	—	—	—	 Stop now 
RStudio with R 3.4.3	Environment	Derek Chan	NYC-Emergency-Vehicles	—	8 Feb 2019, 12:07 AM	—	—	—	—	 

# Watson Studio Local (WSL) Training Load and Access Data

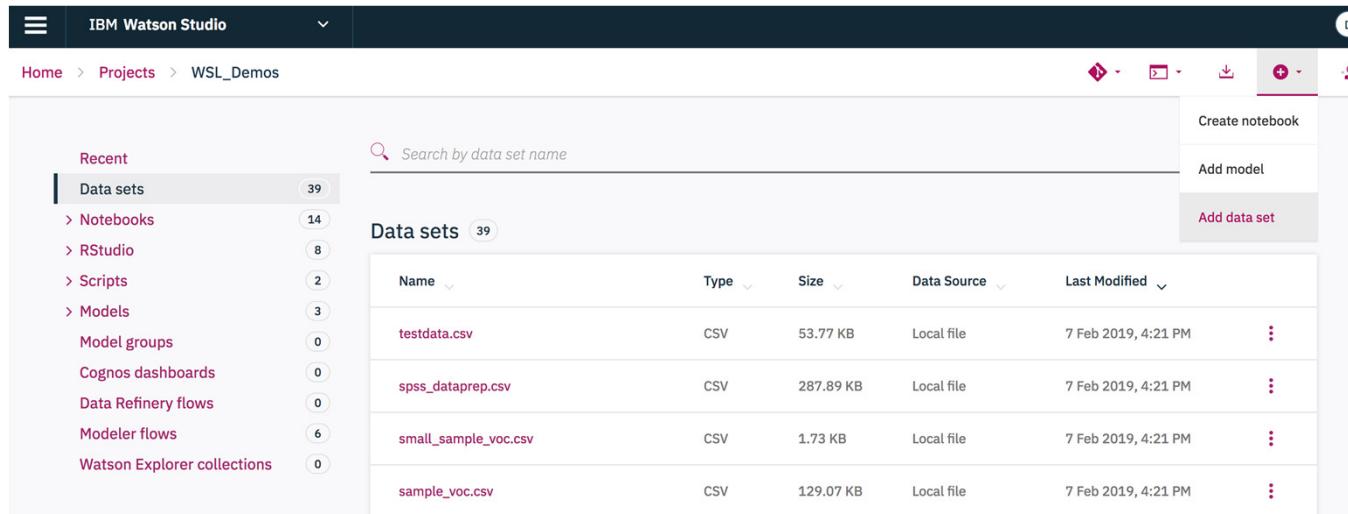


## Load and Access Data

- Your notebooks can load and access data from databases and services using the following methods:
  - Access data from local files
  - Access data in relational databases
  - Add data sources and remote data sets
  - Import a custom JDBC data source
  - Write to data sources
- We will walk through a demonstration of each in the following slides
- It is worth noting that more often than not the data provided to you might contain messy or inaccurate data, or might not be in a suitable structure for your use. You will need to prepare the data (clean and transform it) before you can use it for building models or performing other analytics
  - Cleaning data generally takes 80% of a data scientists time where as modeling accounts for the other 20%

## Add Data from Local Files

- You can upload and load local data assets such as CSV files into your WSL project. When you add a local data asset to a project, any collaborator in that project can load data from it.
- **To upload a local data asset:**
  1. In your project, go to your **Data Sets** page and click **Add data set**. Alternatively, you can click **Add data set** from the project pull-down menu.

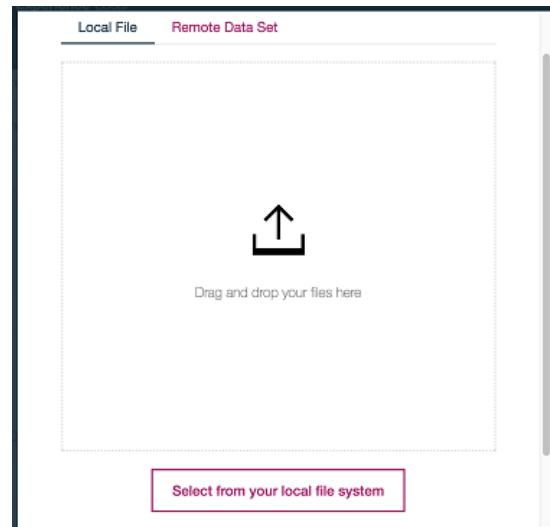


The screenshot shows the IBM Watson Studio interface. The top navigation bar includes 'IBM Watson Studio' and a search bar. Below the navigation is a breadcrumb trail: 'Home > Projects > WSL\_Demos'. On the left, a sidebar lists 'Recent' items: 'Data sets' (39), 'Notebooks' (14), 'RStudio' (8), 'Scripts' (2), 'Models' (3), 'Model groups' (0), 'Cognos dashboards' (0), 'Data Refinery flows' (0), 'Modeler flows' (6), and 'Watson Explorer collections' (0). A search bar at the top right contains 'Search by data set name'. The main content area is titled 'Data sets (39)' and displays a table with columns: Name, Type, Size, Data Source, and Last Modified. The table lists four CSV files: 'testdata.csv', 'spss\_dataprep.csv', 'small\_sample\_voc.csv', and 'sample\_voc.csv', all categorized as 'Local file' and modified on '7 Feb 2019, 4:21 PM'. To the right of the table is a vertical menu with options: 'Create notebook', 'Add model', and 'Add data set'.

Name	Type	Size	Data Source	Last Modified
testdata.csv	CSV	53.77 KB	Local file	7 Feb 2019, 4:21 PM
spss_dataprep.csv	CSV	287.89 KB	Local file	7 Feb 2019, 4:21 PM
small_sample_voc.csv	CSV	1.73 KB	Local file	7 Feb 2019, 4:21 PM
sample_voc.csv	CSV	129.07 KB	Local file	7 Feb 2019, 4:21 PM

## Add Data from Local Files

- 2. Click the Local File tab

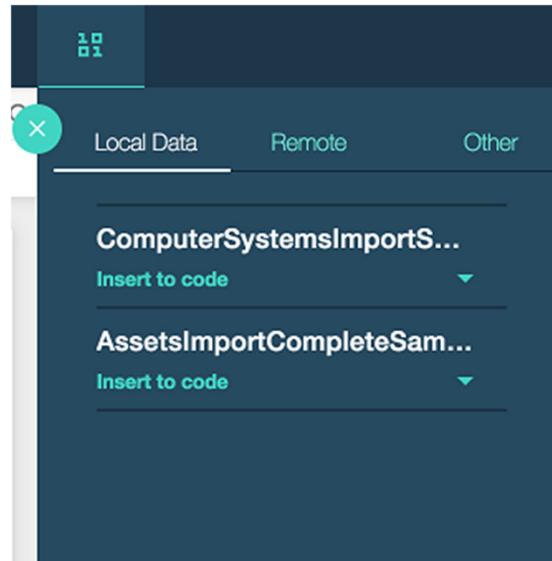


- 3. Drag or browse to your local file system to the palette. The file is now added to the project.

## Add Data from Local Files

- **To Load data from a local data asset:**

1. Open the notebook and click the **Find and Add Data** icon in the toolbar (  ). Only CSV or JSON files can be inserted into notebooks this way currently, and the first data row is always read as the header row
2. Click the **Local Data** tab and select the CSV file to insert



## Access Data in Relational Databases

- If your notebook needs to connect directly to a relational database with your own custom SQL queries, then you can use APIs from third-party modules. The WSL Local documentations recommends:
  - Scala: org.apache.spark
  - Python: jaydebeapi
  - R: rJDBC for Jupyter and Spark for Zeppelin
- In the following slides we will run through an example of how to use Python to connect to directly to a DB2 database

## Access Data in Relational Databases

- To use the jaydebeapi APIs to create JDBC connections to databases, import the following library in your notebook:

```
import jaydebeapi
```

- When the modules are imported, you can form the JDBC string and call the connect function along with a className to create the connection.
- The JDBC string is specific to each database and needs the driver className included when the connection function is called.

Python example for Db2 and Db2 Warehouse on Cloud (previously known as dashDB)

```
import jaydebeapi, sys

#Enter the values for your database connection
dsn_database = "BLUDB" # e.g. "BLUDB" Name of the database
dsn_hostname = "dashdb.services.dal.bluemix.net" # e.g.: "bluemix05.bluforcloud.com"
dsn_port = "50000" # e.g. "50000" Database port number
dsn_uid = "dashxxxx" # e.g. "dash104434" User id
dsn_pwd = "xxxxxxxx" # e.g. "7dBZ3jWt9xN6$o0JiX!m" User password for the database

connection_string='jdbc:db2://'+dsn_hostname+':'+dsn_port+'/'+dsn_database
if (sys.version_info >= (3,0)):
    conn = jaydebeapi.connect("com.ibm.db2.jcc.DB2Driver", connection_string, [dsn_uid, dsn_pwd])
else:
    conn = jaydebeapi.connect("com.ibm.db2.jcc.DB2Driver", [connection_string, dsn_uid, dsn_pwd])

curs = conn.cursor()
curs.execute("select * from db2inst1.sales")
curs.fetchall()
```

## Securing Database Credentials / Sensitive Information

- If your notebook includes code cells with sensitive data, such as credentials for data sources, you can hide those code cells from anyone you share your notebook with.
- When sharing a WSL Notebook only other Project collaborators can see hidden cells
- To hide code cells:
  - Open the notebook and select the code cell to hide.
  - Insert a comment with the hide tag on the first line of the code cell
    - For Python, R, and Scala, enter the following syntax **# @hidden\_cell**

```
# @hidden_cell
credentials_1 <-list(port = "xxxx",db = "BLUDB",username = "xxxxxxxx".
```

## Access Data in Relational Databases

- Although the previous example works it does not return the results in the most usable format
  - Output of that code snippet returns a list which you have to convert to a data frame and include all column names
- The preferred method to use is shown below:

```
import ibm_db
import ibm_db_dbi
import pandas as pd

#Define a function that takes the output from the database and stores it directly into a pandas dataframe
def SQLToDF(sql):
    con = ibm_db.connect(connection_str,'','')
    conn = ibm_db_dbi.Connection(con)
    return pd.read_sql(sql,conn)
```

- Where `connection_str` passes the database credentials in the format:

```
connection_str = 'database=xxxxxx;hostname=xxxxxx;port=50000;uid=xxxxxx;pwd=xxxxxx'
```

- This function outputs a Pandas data frame which is much easier to use

## Add Data Sources and Remote Data Sets

- A **data source** provides a secure mechanism to store and manage credentials for a database, as opposed to storing them in files managed by the notebook or RStudio tools.
- It also makes a password easier to change because it can be updated in one secure location, as opposed to every file where that database is referenced.
- **Remote data sets** can be added to each data source. Remote data sets connect directly to a schema and table name, and can be accessed directly from WSL APIs, machine learning models, and flows.
- When you collaborate on a project, you can opt to create remote data sets for the most relevant and useful tables from the data source.

## Add Data Sources and Remote Data Sets

- In WSL, you can create data sources and remote data sets for the following on-premises databases and services:
  - Big SQL
  - Db2 Warehouse on Cloud
  - Db2 for Linux, UNIX, and Windows
  - DB2 for z/OS
  - Hive for HDP
  - HDFS for HDP
  - Hive for Cloudera (CDH)
  - HDFS for Cloudera (CDH)
  - Informix
  - Netezza
  - Oracle

## Add Data Sources and Remote Data Sets

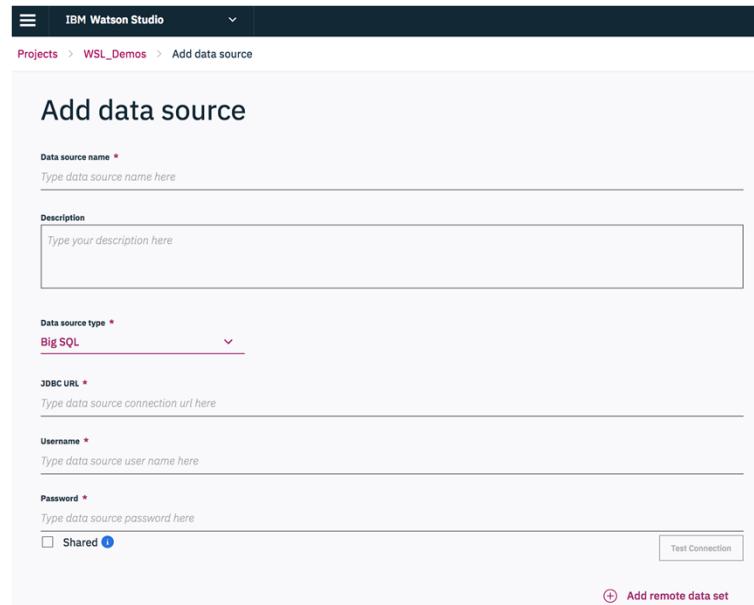
- To create a data source and remote data set:

1. From your project, go to the **Data Sources** page and click **add data source**
2. In the Add Data Source window, specify all of the required fields

\*\*If you select the **Shared** check box, the data source and its credentials become accessible to all collaborators in the project.

\*\*If you do not select **Shared**, then each collaborator must provide their own credentials to use the data source.

**Important:** When a data source is shared, it cannot be made private again



The screenshot shows the 'Add data source' interface in IBM Watson Studio. At the top, there's a navigation bar with 'IBM Watson Studio' and a dropdown menu. Below it, the path 'Projects > WSL\_Demos > Add data source' is visible. The main area is titled 'Add data source'. It contains several input fields:

- 'Data source name \*' with placeholder 'Type data source name here'
- 'Description' with placeholder 'Type your description here'
- 'Data source type \*' dropdown currently set to 'Big SQL'
- 'JDBC URL \*' with placeholder 'Type data source connection url here'
- 'Username \*' with placeholder 'Type data source user name here'
- 'Password \*' with placeholder 'Type data source password here'
- A checkbox labeled 'Shared' with a help icon, which is checked by default.
- A 'Test Connection' button next to the 'Shared' checkbox.
- A '+ Add remote data set' button at the bottom right.

## Add Data Sources and Remote Data Sets

- 3. Select the Data Source Type, then use the following format for the URL field

On-premise database or service	JDBC URL format	Example
Db2 for Linux, UNIX, and Windows	<code>jdbc:db2://{{HOSTNAME}}:{{PORT}}/{{DATABASE}};</code>	<code>jdbc:db2://9.87.654.321:50000/SAMPLE:user=DB2INST1;</code>
Db2 with SSL	<code>jdbc:db2://{{HOSTNAME}}:{{PORT}}/{{DATABASE}}:sslConnection=true;sslTrustStoreLocation={{TRUSTSTOREPATH}}</code>	<code>jdbc:db2://9.87.654.321:50001/SAMPLE:sslConnection=true;sslTrustStoreLocation=/user-home/_global/_security/customer-truststores/cacerts;</code>
Db2 for z/OS	<code>jdbc:db2://{{HOSTNAME}}:{{PORT}}/{{DATABASE}};</code>	<code>jdbc:db2://dtec649.vmec.ibm.com:446/STLEC2;</code>
Db2 Warehouse on Cloud	<code>jdbc:db2://{{HOSTNAME}}:{{PORT}}/{{DATABASE}};</code>	<code>jdbc:db2://dashdb-entry-yp-dal09-07.services.dal.bluemix.net:50000/BLUDB;</code>
Netezza	<code>jdbc:netezza://{{HOSTNAME}}:{{PORT}}/{{DATABASE}}</code>	<code>jdbc:netezza://9.87.654.321:5480/SYSTEM;</code>
Informix	<code>jdbc:informix-sqli://{{HOSTNAME}}:{{PORT}}/{{DATABASE}}:INFORMIXSERVER={{SERVERNAME}};DELMIDENT=y;</code>	<code>jdbc:informix-sqli://9.87.654.321:9088/stores_demo:INFORMIXSERVER=dev;DELMIDENT=y;</code>
Oracle	<code>jdbc:oracle:thin:{{USERNAME}}/{{PASSWORD}}@{{HOSTNAME}}:{{PORT}}:{{SID}}</code>	<code>jdbc:oracle:thin:system/oracle@9.87.654.321:1521:xe</code>
Big SQL	<code>jdbc:db2://{{HOSTNAME}}:{{PORT}}/{{DATABASE}}</code>	<code>jdbc:db2://9.87.654.321:32051/bigsql</code>
Big SQL with SSL	<code>jdbc:db2://{{HOSTNAME}}:{{PORT}}/{{DATABASE}}:sslConnection=true;sslTrustStoreLocation={{TRUSTSTOREPATH}}</code>	<code>jdbc:db2://other2.fyre.ibm.com:51000/bigsql:sslConnection=true;sslTrustStoreLocation=/user-home/global/_security/customer-truststores/cacerts;</code>
MS SQL (custom)	<code>jdbc:sqlserver://{{HOSTNAME}}:{{PORT}};databaseName={{DBNAME}}</code>	<code>jdbc:sqlserver://9.30.54.105:1433;databaseName=master</code>

## Add Data Sources and Remote Data Sets

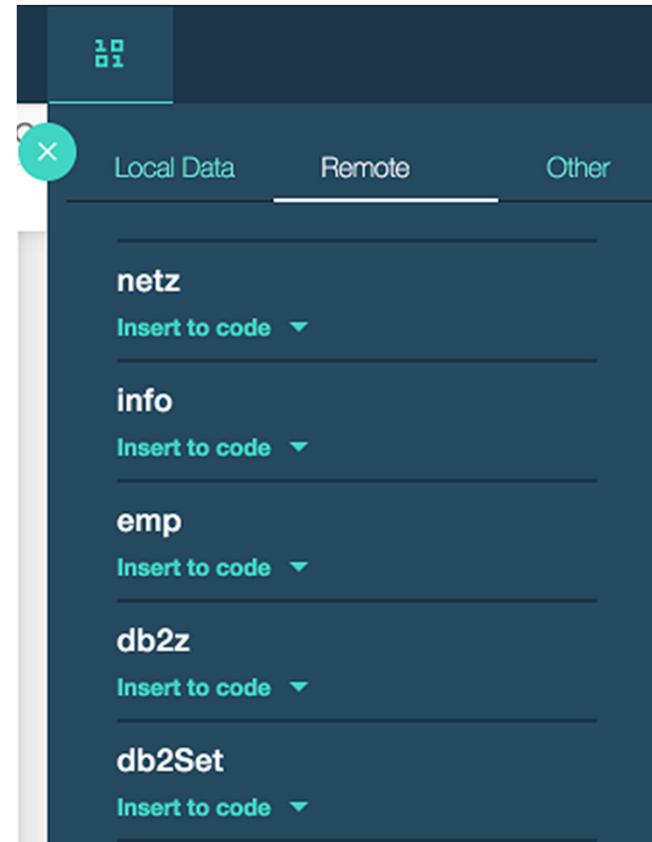
- **A few additional notes on data sources and remote data sets:**

- All collaborators in a project can use a shared data source in their notebook, or add their own private data source.
- Collaborators with viewer privileges cannot edit data sources.
- You can add or edit remote data sets that connect directly to a schema and table name in the data source

## Add Data Sources and Remote Data Sets

- **To load data from a remote data set:**

1. You can automatically load data into a data frame in a notebook by using the **Find and Add Data** icon (- 2. Then click the **Remote** tab
- 3. Find the data set that you want, and click **Insert to Code**



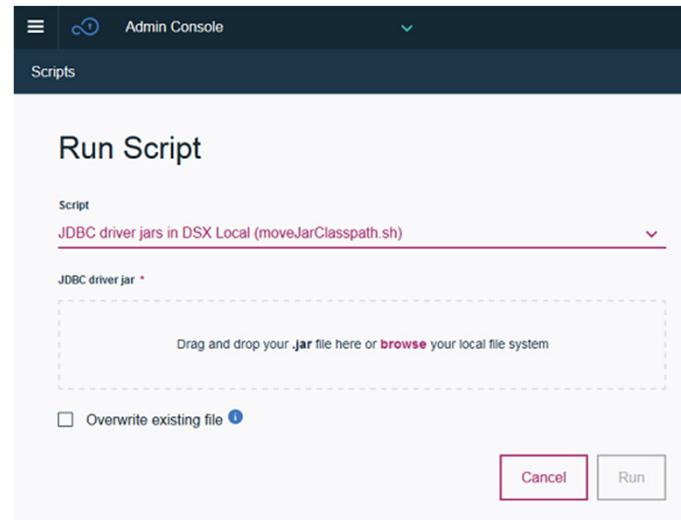
## Import a custom JDBC data source

- If a WSL user plans to write notebook or RStudio code that uses a relational database beyond the data source options available in DSX Local that were listed on a previous slide, then the DSX Local administrator must complete the following steps:
  1. Import the new JDBC Driver
  2. Create a custom data source and remote data set
  3. Load data in the notebook
- We will run through each of these steps on the following slides

## Import a custom JDBC data source

### ▪ 1. Import a JDBC driver:

- To import a JDBC driver using the Admin Console, click the menu icon ( ) and click **Scripts**
- In the **Scripts** pull-down menu, select **JDBC driver jars in WSL (moveJarClasspath.sh)**, and click **add jar**
- Drag and drop the .jar file or browse to it and click the **Run** button



## Import a custom JDBC data source

- **2. Create a custom data source and remote data set:**
  - From your project, go to the **Data Sources** page and click **add data source**
  - In the Create Data Source window, specify all of the required fields
  - In the **Data Source Type** pull-down menu, select **Custom JDBC**
  - Type in the **JDBC driver class name**
  - Type in the **JDBC URL**
  - Determine whether or not you want to setup this data source as **Shared** or not

Add data source

**Data source name \***  
Type data source name here

**Description**  
Type your description here

**Data source type \***  
Custom JDBC

**JDBC driver class name \***  
Type data source jdbc driver class name here

**JDBC URL \***  
Type data source connection url here

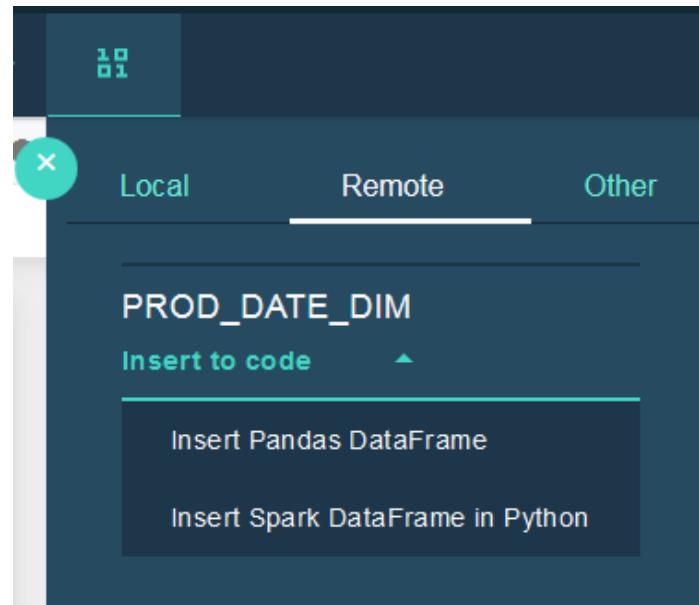
**Username \***  
Type data source user name here

**Password \***  
Type data source password here

Shared 

## Import a custom JDBC data source

- **3. Load data in the notebook:**
  - To automatically load data into a data frame in a notebook, open the notebook and click the **Find and Add Data** icon () in the toolbar
  - Then, click the **Remote** tab
  - Find the data set that you want, and click **Insert to code**



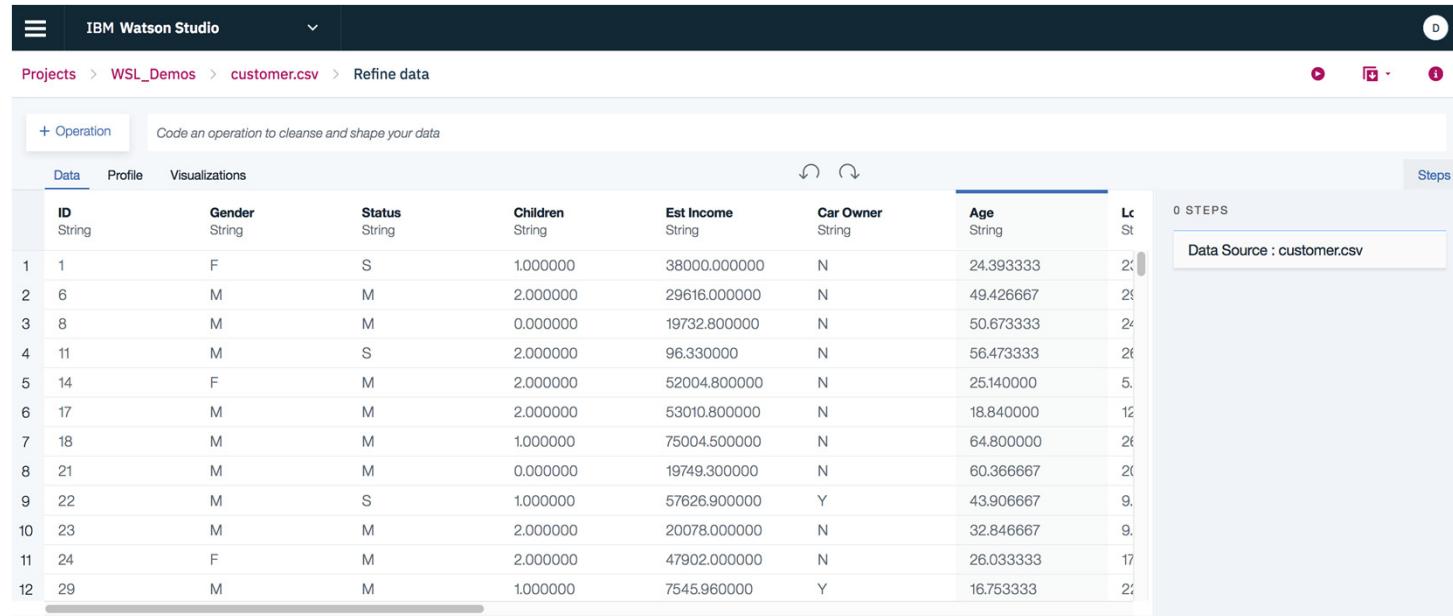
## Refine Data in WSL

- Before you analyze local data sets in WSL, you can refine the data by using the Data Refinery in the following ways:
  - **Cleanse the data:** fixing or removing data that is incorrect, incomplete, improperly formatted, or duplicated.
  - **Shape the data:** customize data by filtering, sorting, combining, or removing columns, and performing operations.
- The Data Refinery is a visual tool for “clickers”
- You can refine the data in real-time to build a customized data flow, and then save the data flow as either a separate JSON file or an R script.

## Refine Data in WSL

### ▪ Create a data flow:

1. Go to your project assets page and click on the name of the local data set
2. Click the **Data** tab. Note that for large data sets, you will only see a subset of the rows even though the operations will run on the entire data set



The screenshot shows the IBM Watson Studio interface for refining data. The top navigation bar includes 'IBM Watson Studio' and a 'D' icon. Below it, the breadcrumb navigation shows 'Projects > WSL\_Demos > customer.csv > Refine data'. The main area has tabs for '+ Operation' (selected), 'Data', 'Profile', and 'Visualizations'. A message says 'Code an operation to cleanse and shape your data'. The 'Data' tab displays a preview of 12 rows from the 'customer.csv' file. The columns are: ID, Gender, Status, Children, Est Income, Car Owner, Age, and Lk St. The preview shows various values like 'F', 'S', 'M', 'N', and numerical values for age and income. To the right, a 'Steps' panel indicates '0 STEPS' and lists 'Data Source : customer.csv'.

ID	Gender	Status	Children	Est Income	Car Owner	Age	Lk St
1	F	S	1.000000	380000.000000	N	24.393333	21
2	M	M	2.000000	29616.000000	N	49.426667	23
3	M	M	0.000000	19732.800000	N	50.673333	24
4	M	S	2.000000	96.330000	N	56.473333	26
5	F	M	2.000000	52004.800000	N	25.140000	5.
6	M	M	2.000000	53010.800000	N	18.840000	12
7	M	M	1.000000	75004.500000	N	64.800000	26
8	M	M	0.000000	19749.300000	N	60.366667	20
9	M	S	1.000000	57626.900000	Y	43.906667	9.
10	M	M	2.000000	20078.000000	N	32.846667	9.
11	F	M	2.000000	47902.000000	N	26.033333	17
12	M	M	1.000000	7545.960000	Y	16.753333	21

## Refine Data in WSL

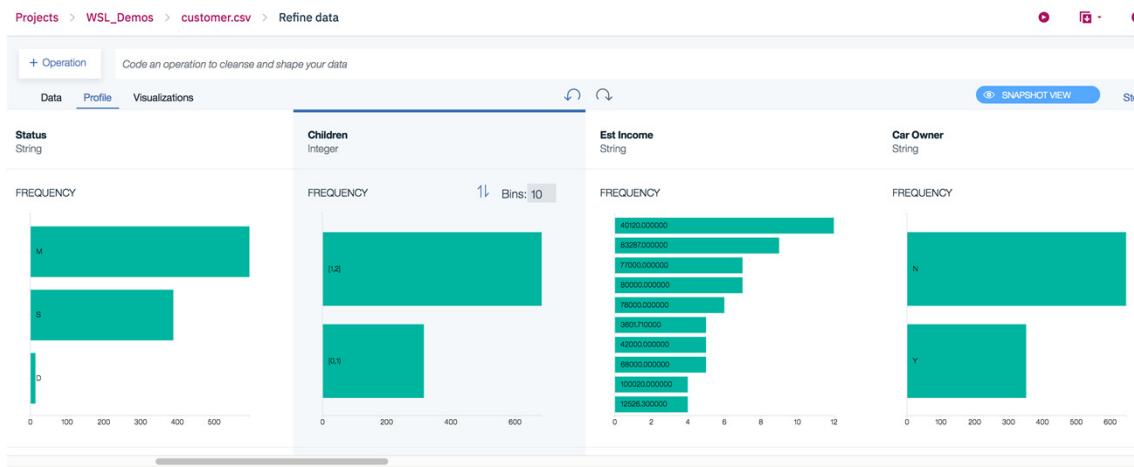
- 3. Click the  icon in a column and select an operation to cleanse, shape, or enrich the column. Alternatively, click Operations to browse operation categories or search for a specific operation. You can also manually enter R code in the command line, and let autocomplete assist you with the syntax.
- 4. Each operation you add to the data flow appears as a step under the Steps pane. You can click each step for a snapshot of how the operation will refine the data. You can also undo, redo, edit, or delete steps. To insert an operation between two existing steps, click the step before the position and then apply the new operation.
- 5. To save the data flow as a separate JSON file in the Data Flows section of your assets page, click the  icon and click Save Data Flow. To save the flow as an R script in the Scripts section of your assets page, click the  icon and click Save R Script



## Refine Data in WSL

### ▪ Validate Data:

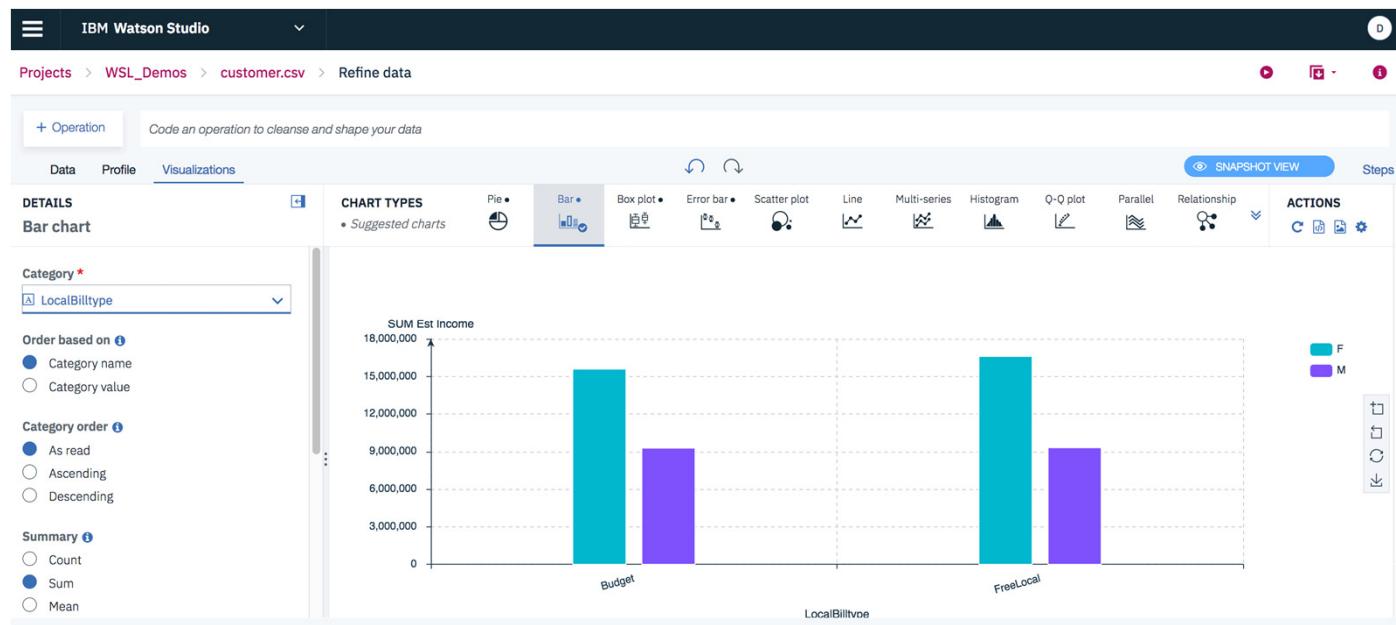
- Click the Profile tab for frequency and summary statistics for each of your columns. Hover over a bar to see the data values
- Frequency is the number of times that a value, or a value in a specified range, occurs. Each frequency distribution (bar) shows the count of unique values in a column
- Statistics are a collection of quantitative data. Depending on a column's data type, the statistics for each column will vary slightly



## Refine Data in WSL

### ▪ Visualize Data:

- Click the **Visualizations** tab to input columns and view data visualizations that provide insights and uncover patterns, trends, and correlations in your data.



## Write to Data Sources

- The following Python 2.7 example writes a Pandas data frame to data sources from a Jupyter notebook

Python example for Db2 and BIGSQL

```
import dsx_core_utils, os, io
import pandas as pd
from sqlalchemy import create_engine

#Read csv to pandas
df_data_1 = pd.read_csv('../datasets/userdatap1.csv')

dataSet = dsx_core_utils.get_remote_data_set_info('db2Set')
dataSource = dsx_core_utils.get_data_source_info(dataSet['datasource'])
#SQL Alchemy URL
sqla_url= "db2+ibm_db://" + dataSource['user']+ ':' + dataSource['password'] + "@9.30.57.224:50000/SAMPLE"

#Pandas does not support many databases so we use recommended sqlalchemy
engine = create_engine(sqla_url, pool_size=10, max_overflow=20)
conn = engine.connect()

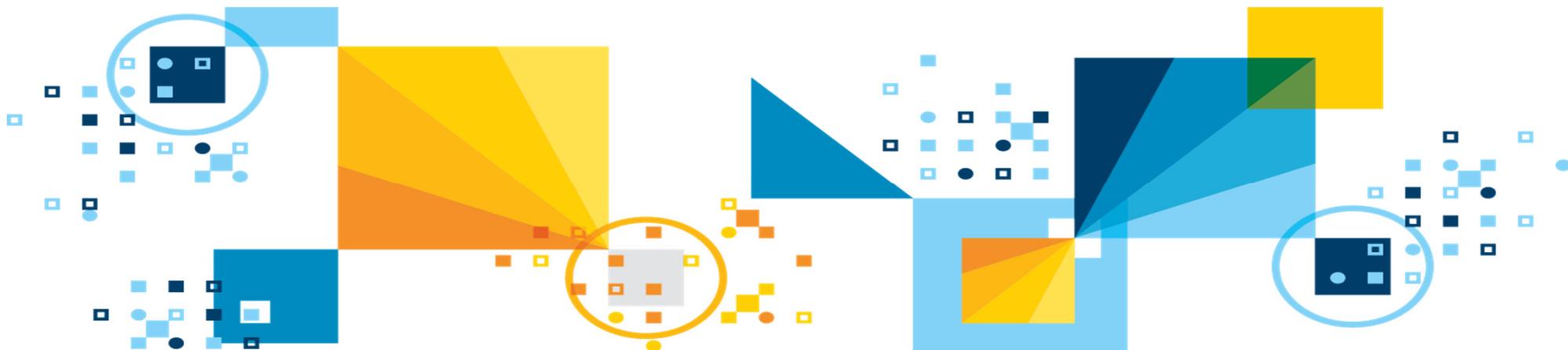
##Write to database
df_data_1.to_sql(dataSet['table'], engine, schema=dataSet['schema'], if_exists='replace')

##Read back again to confirm it has been written
query = 'select * from ' + dataSet['schema'] + '.' + dataSet['table']
df_data_1 = pd.read_sql(query, con=conn)
df_data_1.head()
```

- where db2Set is the name of the data set, 9.30.57.224 is the sample IP of a DB2 database server, SAMPLE is the example database name, 50000 is the db2 non-SSL port number, and '../datasets/userdatap1.csv' is a sample csv file used to create a data frame which is then written to a table

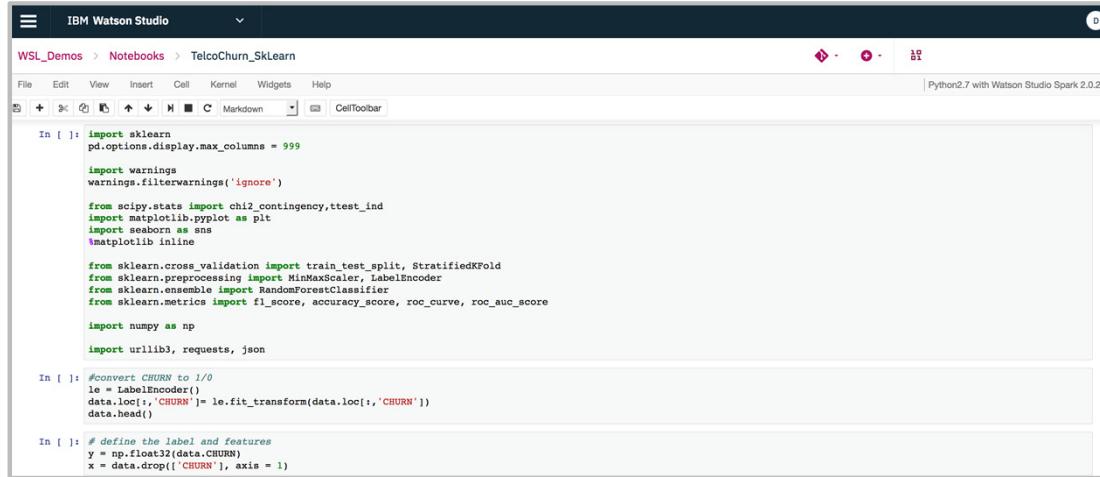
# Watson Studio Local (WSL) Training

## Analyze Data



## Notebooks

- A **Jupyter notebook** is a web-based environment for interactive computing. You can run small pieces of code that process your data, and you can immediately view the results of your computation.
- Notebooks include all of the building blocks you need:
  - The data
  - The code computations that process the data
  - Visualizations of the results
  - Text and rich media to enhance understanding



The screenshot shows the IBM Watson Studio interface with a Jupyter Notebook open. The notebook is titled 'WSL\_Demos > Notebooks > TelcoChurn\_SkLearn'. The code in the first cell is:

```
In [ 1]: import sklearn  
pd.options.display.max_columns = 999  
  
import warnings  
warnings.filterwarnings('ignore')  
  
from scipy.stats import chi2_contingency,tttest_ind  
import matplotlib.pyplot as plt  
import seaborn as sns  
%matplotlib inline  
  
from sklearn.cross_validation import train_test_split, StratifiedKFold  
from sklearn.preprocessing import MinMaxScaler, LabelEncoder  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import f1_score, accuracy_score, roc_curve, roc_auc_score  
  
import numpy as np  
  
import urllib3, requests, json
```

The second cell starts with:

```
In [ 1]: #convert CHURN to 1/0  
le = LabelEncoder()  
data.loc[:, 'CHURN']= le.fit_transform(data.loc[:, 'CHURN'])  
data.head()
```

The third cell starts with:

```
In [ 1]: # define the label and features  
y = np.float32(data.CHURN)  
x = data.drop(['CHURN'], axis = 1)
```

## Notebooks

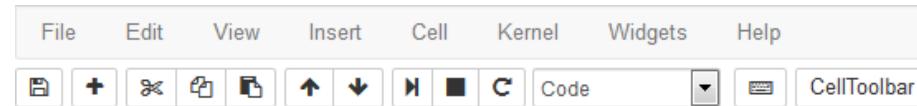
- Code computations can build upon each other to quickly unlock key insights from your data.
- Notebooks record how you worked with data, so you can understand exactly what was done, reproduce computations reliably, and share your findings with others.
- With WSL, you can create Python, Scala, and R notebooks to analyze your data.
- You can collaborate with others on your notebooks, add comments, and view a history of your notebooks

## Notebooks - The Parts of a Notebook

- The tools to work with Jupyter and Zeppelin notebooks have easy-to-learn features, including menu and toolbar, action bar, and cells or paragraphs
- We will cover the following features in the upcoming slides:
  - Menu bar and toolbar
  - Notebook action bar
  - The cells in a Jupyter notebook
  - The paragraphs in a Zeppelin notebook
  - Spark job progress bar

## Notebooks – Menu Bar and Toolbar

- The **Menu Bar** and tool bar are shown in the image below:



- You can select notebook features that affect the way the notebook functions and perform the most-used operations within the notebook by clicking an icon in the toolbar.

## Notebooks – Action Bar

- The **Action Bar** is shown in the image below:



- You can select features that enhance notebook collaboration
- From the Action Bar you can:
  - Perform Git actions such as pull and push
  - Create new assets
  - Find and add data sets

## Notebooks – The cells in a Jupyter Notebook

- A **Jupyter Notebook** consists of a sequence of cells
- The flow of a notebook is sequential. You enter code into an input cell, and when you run the cell, the notebook runs the code and prints the output of the computation to an output cell
- You can change the code in an input cell and re-run the cell as often as you like. In this way, the notebook follows a read-evaluate-print loop paradigm
- The behavior of a cell is determined by a cell's type. The different types of cells include:
  - Code cells
  - Markdown cells
  - NBConvert cells

## Notebooks – Code Cells

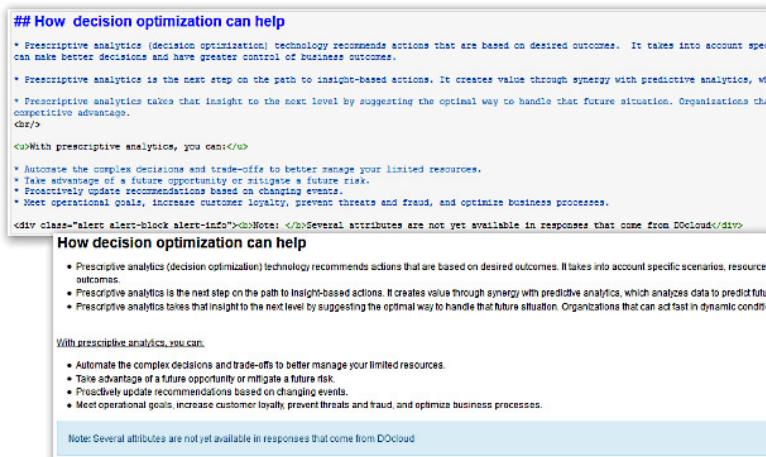
- You write and edit code in **Code Cells**, an example can be seen below:

```
In [6]: import pandas as pd
gaspd = pd.DataFrame([(gas_names[i],int(gas_data[i][0]),int(gas_data[i][1]),int(gas_data[i][2]),int(gas_data[i][3]))
                      for i in range_gas])
oilpd = pd.DataFrame([(oil_names[i],int(oil_data[i][0]),int(oil_data[i][1]),int(oil_data[i][2]),oil_data[i][3])
                      for i in range_oil])
gaspd.columns = ['name','demand','price','octane','lead']
oilpd.columns= ['name','capacity','price','octane','lead']
```

- If we refer to the above image we can talk about what's going on from a high level:
  - 1. They are importing a library known as pandas
  - 2. They are defining two dataframes, one called “gaspd” the other called “oilpd”
  - 3. We see that there is a number 6 off to the left of the code cell. That signifies that this cell has completed running and it was the sixth cell in your notebook to have run
- **Note** - If you want to insert a comment to a code cell you must preface the line by using the # symbol

## Notebooks – Markdown Cells

- You can document the computational process in **Markdown Cells**
- You can also:
  - Input headings to structure your notebook hierarchically
  - Add and edit image files as attachments to the notebook
- The markdown code and images are rendered when the cell is run
- So if we look at the images below, the top image is what it looks like while you're inputting information into a Markdown cell and the bottom image is what the cell looks like after it is run



## Notebooks – Markdown Cell “Cheatsheet”

- Here is a URL that will give you all the information you should need to use Markdown cells appropriately:
  - <https://content-dsxlocal.mybluemix.net/docs/content/analyze-data/markd-jupyter.html>

## Notebooks – NBConvert Cells

- You can use *raw NB Convert cells* to write output directly or save code that you don't want to run.  
Raw cells are not evaluated by the notebook
- So an example use of a raw NB Convert cell would be to use it as sort of a “clip board” and keep re-usable / popular lines of code that you can copy and paste into your Notebook

```
from docplex.cp.model import *
url = "https://api-oaas.doccloud.ibmcloud.com/job_manager/rest/v1"
key = "api_5bf69af0-ba5f-4236-b8eb-4c9a2f909436"
```

## Notebooks – Paragraphs in a Zeppelin Notebook

- **Zeppelin Notebooks** are similar to Jupyter notebooks but can contain a mixture of Scala, Python, and R
- The flow of a notebook is also sequential, and you enter code into an input paragraph (instead of cells)
- When you run the paragraph the notebook runs the code and prints the output of the computation to an output paragraph
- You can change the code in an input paragraph and re-run the paragraph as often as you like. In this way, the notebook follows a read-evaluate-print loop paradigm



Let's create a dataframe that just identifies the correct and incorrect predictions.  
(1 is correct and 0 is incorrect)

```
val matches = udf((A : Int, B: Int) => {
  if (A+B == 1) 0
  else 1
})

val total = prediction.count
val rightWrong = prediction.withColumn("matches", matches($"prediction", $"binaryLabel")).groupBy("matches").count.toDF
rightWrong.registerTempTable("rightWrong")
rightWrong.show
```

## Notebooks – Supported Runtimes

- WSL provides many different runtime images for you to use
- They are:
  - Jupyter with Python 2.7, Scala 2.11, R 3.4.3, Spark 2.0.2
  - Jupyter with Python 3.5, Scala 2.11, R 3.4.3, Spark 2.2.1
  - Jupyter with Python 3.5 for GPU
  - H2O Flow
  - Zeppelin with Anaconda2, Python 2.7
- There is no right or wrong runtime for you to use, whichever is most comfortable and supports the features you are looking for in terms of your development

## Notebooks – The Jupyter and Spark Environment

- Jupyter Notebooks run on **Jupyter Kernels** and, if the notebook uses Spark APIs, those kernels run in Spark engines
- Jupyter Kernels:
  - When you open a notebook in edit mode, exactly one interactive session connects to a Jupyter kernel for the notebook language and Spark version that you select
  - This kernel executes code that you send and returns the computational results. You can switch the kernel to change the notebook language or Spark version.
  - If necessary, you can restart or reconnect to the kernel. When you restart a kernel, the kernel is stopped and then started with the same session, but all execution results are lost.
  - When you reconnect to a kernel after losing a connection, the notebook is connected to the same kernel session, and all previous execution results are available
  - The kernel remains active even if you leave the notebook or close the web browser window. When you reopen the same notebook, the notebook is connected to the same kernel.

## Notebooks – Create a Notebook

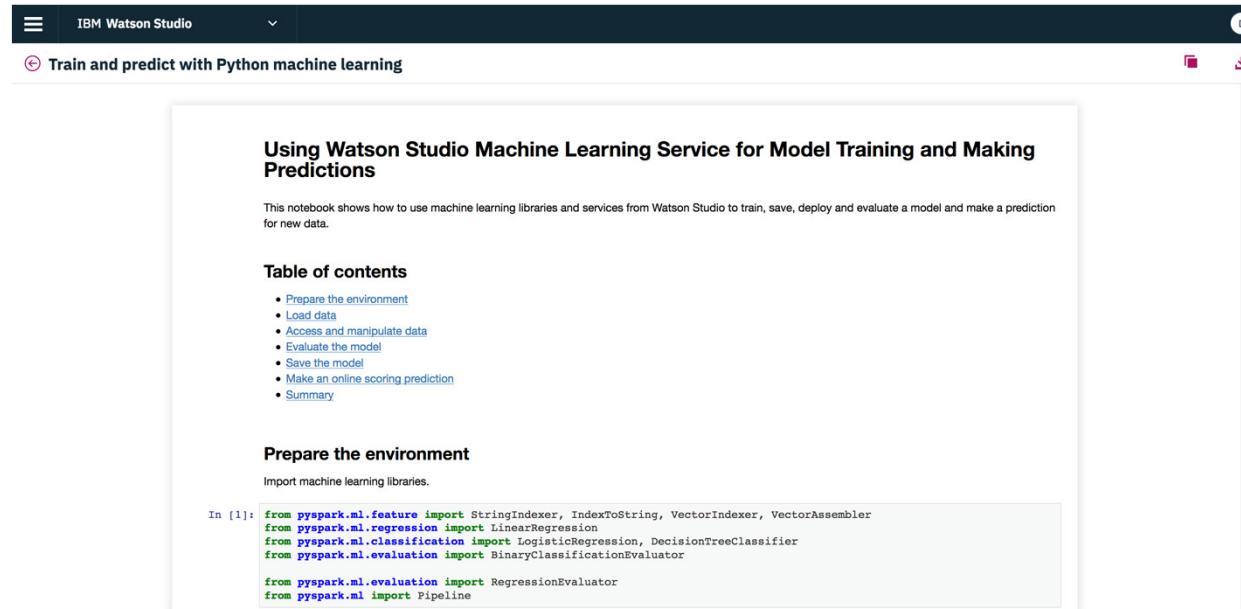
- To create a notebook in WSL you need a project configured
- Then from the project assets view, click the **Add Notebook** link
- In the **Create Notebook Window**, specify the method to use to create your notebook
- You have 3 options:
  - **Blank:** Create a brand new notebook
  - **From File:** Upload a notebook from your local file system
  - **From URL:** Upload a notebook from a URL



- Specify the rest of the details for your notebook
- Click **Create Notebook**

## Notebooks – Create a Notebook

- Alternatively, you can copy a sample notebook from the **Community** page
- To work with a copy of the sample notebook, click the Open icon (  ) and the Notebook will open in “Edit” mode so you can run cells and copy code
- You can open a variety of sample notebooks to help you get started



The screenshot shows the IBM Watson Studio interface. The top navigation bar includes a menu icon, the text "IBM Watson Studio", and a search bar. Below the navigation is a header bar with a circular icon and two small icons on the right. The main content area displays a notebook titled "Using Watson Studio Machine Learning Service for Model Training and Making Predictions". The notebook's purpose is explained: "This notebook shows how to use machine learning libraries and services from Watson Studio to train, save, deploy and evaluate a model and make a prediction for new data." A "Table of contents" section lists eight items: "Prepare the environment", "Load data", "Access and manipulate data", "Evaluate the model", "Save the model", "Make an online scoring prediction", and "Summary". The "Prepare the environment" section contains Python code for importing necessary libraries:

```
In [1]: from pyspark.ml.feature import StringIndexer, IndexToString, VectorIndexer, VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.classification import LogisticRegression, DecisionTreeClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml import Pipeline
```

## Notebooks – Install Custom or 3<sup>rd</sup> Party Libraries

- You can add processing capabilities to Apache Spark in WSL by installing external or third-party libraries
- WSL includes many pre-installed libraries
- Before you install a library, check the list of preinstalled libraries. Run the appropriate command from a notebook cell:
  - Python: !pip list --isolated
  - R: installed.packages()
- If the library you want to use is not listed, then you can use the commands on the following slides to install it

## Notebooks – Install a Python Library

- **1. Use the Python `pip package installer` command to install Python libraries to your notebook**
  - For example, using the command “`!pip install prettyplotlib`” installs the prettyplotlib library into a pod’s conda and removes the package once the pod is terminated
  - Or using the command “`!pip install –user prettyplotlib`” installs the prettyplotlib library in your user home and the package exists even after the pod is terminated
- **2. Use the Python `import command` to import the library components**
  - Going off our example above, you would then use the command “`import prettyplotlib as ppl`”
- **3. Restart the kernel**

## Globally Install Libraries and Packages

- A WSL administrator can install Python or R packages in global directories. These packages are then made available to all users of the cluster
- **To install a global Python library:**
  - Log in to WSL as admin and create a Python notebook
  - Use the Python pip package installer command to install Python libraries to your notebook for example see the image below:

```
!pip install --target  
/user-home/_global_/python-2.7 prettyplotlib
```

- The installed packages can be used by all notebook users **that use the same Python version in the Spark service**
- Notebook users can now use the Python import command to import the library components.
  - For example, users can run the following command in a code cell:

```
import prettyplotlib as ppl
```

## Globally Install Libraries and Packages

- **To install a global R package:**

- Log in to WSL as admin and create an R notebook
  - Use the R `install.packages()` function to install new R packages. For example, running the following command would install the `ggplot2` package

```
install.packages("ggplot2")
```

- The imported package can be used by all R notebooks **that are running in the Spark service**
- Now, users can use the **R library() function** to load the installed package
  - For example, a user can run the following command in a code cell:

```
library("ggplot2")
```

## R Studio Overview

- R is a popular statistical analysis and machine-learning package that enables data management and includes tests, models, analyses, and graphics, and enables data management.
- RStudio, included in WSL, provides an IDE for working with R.
- An RStudio session created in WSL includes 2 GB of storage and 5 GB of memory available for your use

## R Studio – Change Spark Version

- When you're in R Studio the default Spark version is 2.0.2
- If you want to change to a different version it's not as easy as switching to a different kernel like it was in Notebooks
- You have the option of calling one of the two below libraries and performing a few commands to do so:
  - Sparklyr
  - SparkR

## R Studio – Change Spark Version

### ▪ **Sparklyr:**

- You will call the `spark_connect()` function, you will use the below line of code to connect to the Spark 2.2.1 cluster:

To connect to the Spark 2.2.1 cluster:

```
sc <- spark_connect( master =
  "spark://spark-master221-svc:7077",
  spark_home="/usr/local/spark-2.2.1-bin-hadoop2.7" )`
```

- To change back to the Spark 2.0.2 cluster use the following line of code:

To connect to the Spark 2.0.2 cluster:

```
sc <- spark_connect( master =
  "spark://spark-master-svc:7077" )
```

## R Studio – Change Spark Version

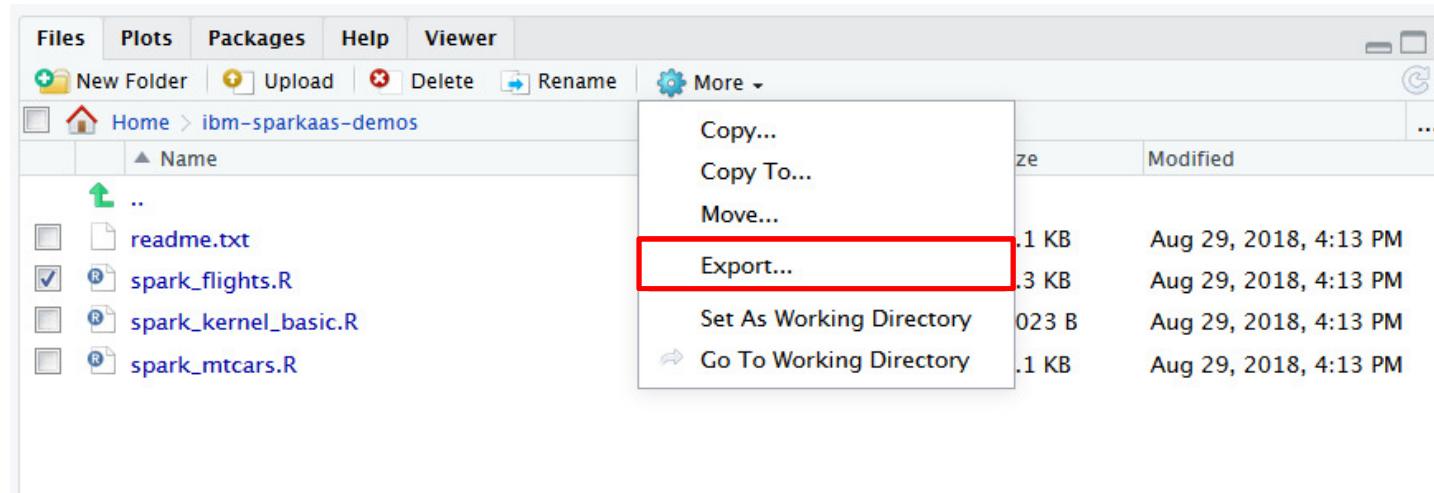
### ▪ SparkR

- You will use the \$SPARK\_HOME environment variable to specify the Spark 2.2.1 installation location in RStudio

```
Sys.setenv("SPARK_HOME"="/usr/local/spark-2.2.1-bin-hadoop2.7")
# import SparkR
library(SparkR, lib.loc =
"/usr/local/spark-2.2.1-bin-hadoop2.7/R/lib")
# initial sc
sc = sparkR.session(master="spark://spark-master221-svc:7077",
appName="dsxlRstudioSpark221")
```

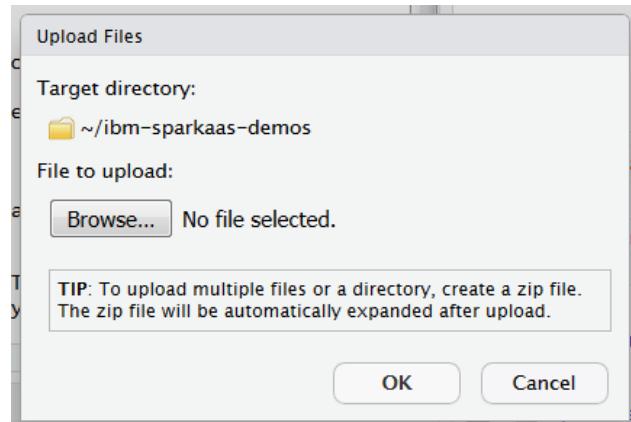
## R Studio – Transfer Files to/from your User Project Folder

- Using the File Explorer in R Studio a WSL user can upload and download files between their project folder and a local disk outside of the cluster
- To download an R Studio file, select it by clicking the check box to the left of the file and then click on **More** and selecting **Export...**
- This will allow you to save the file to your local disk



## R Studio – Transfer Files to/from your User Project Folder

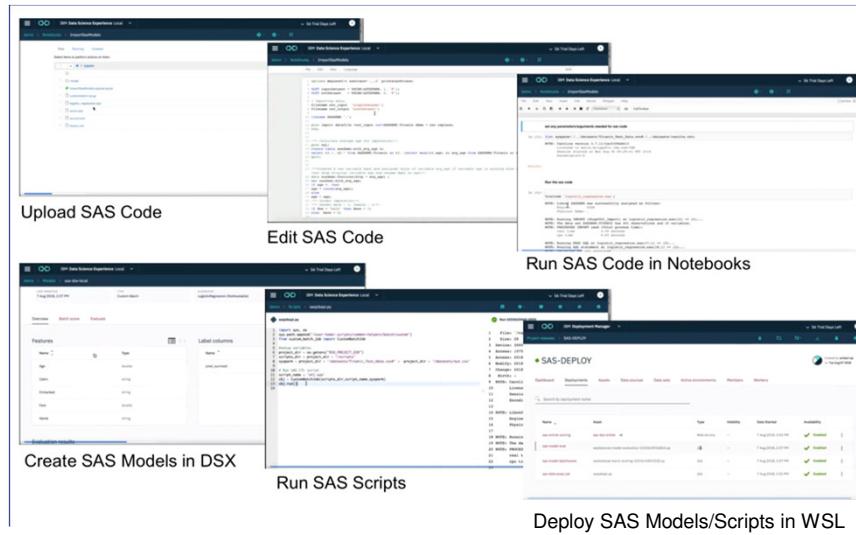
- To upload an R Studio file click **Upload**. You will be taken to the screen shown below



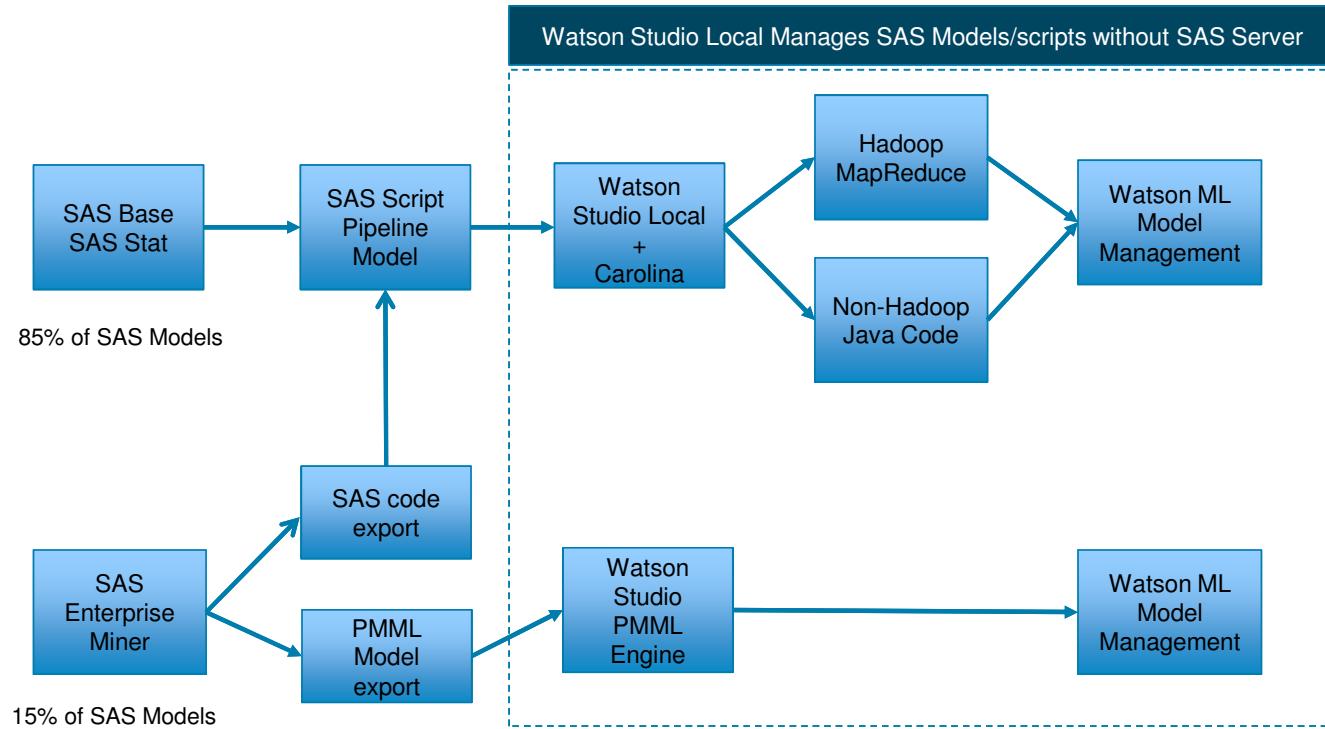
- Selecting **Browse...** allows you to browse your local file system and choose the file you would like to upload

## SAS on WSL

- WSL offers the capability to execute SAS scripts/models, via Carolina
- You can move existing SAS scripts/models to open platforms without having to recode the models/scripts in a different language
- This facilitates SAS-based data scientists to collaborate with open source-based data scientists on a single platform
- It supports real-time scoring of models



# Approach



## Current Supported (WSL + Carolina)

- Data preparation
  - Edit and run SAS base code
- Model editing and re-training
  - Re-train SAS STAT/EM models in SAS Desktop and re-import to WSL
  - Convert SAS STAT/EM training code to user maintainable Python code
- Model scoring
  - Score SAS STAT/EM models
- Model lifecycle management

## Visualizations

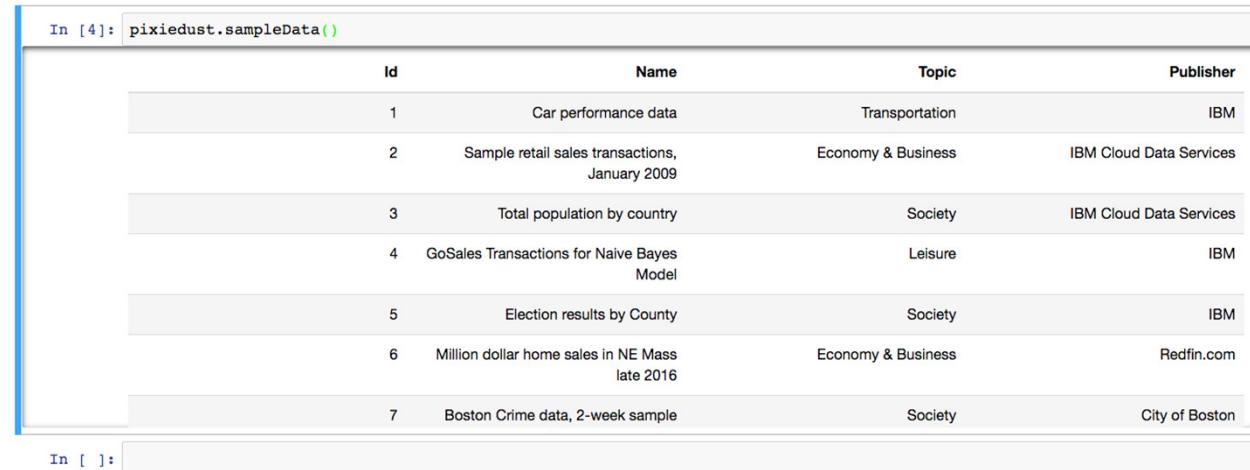
- Use visualizations in your notebooks to present data visually to help identify patterns, gain insights, and make decisions
- Many of your favorite open source visualization libraries, such as matplotlib, are pre-installed on WSL
- You can also install other visualization libraries and packages
- We will cover using these libraries:
  - PixieDust
  - Brunel

## PixieDust Visualizations

- PixieDust is an open source Python helper library that works as an add-on to Jupyter notebooks to improve the user experience of working with data
- It also fills a gap for users who have no access to configuration files when a notebook is hosted on the cloud
- PixieDust brings you the power to create compelling visuals fast, no coding necessary
- **Note:** PixieDust makes data visualization much easier, but does require basic knowledge of how to load/access and manipulate data within a Python notebook

## Load Data with PixieDust

- PixieDust comes with sample data
- To start playing with the display() API and other PixieDust features, load and then visualize one of our many sample data sets
- To call the list of data sets, run the following command in your notebook:
  - `Pixiedust.sampleData()`
- You get a list of datasets included with PixieDust



In [4]:	pixiedust.sampleData()			
	Id	Name	Topic	Publisher
	1	Car performance data	Transportation	IBM
	2	Sample retail sales transactions, January 2009	Economy & Business	IBM Cloud Data Services
	3	Total population by country	Society	IBM Cloud Data Services
	4	GoSales Transactions for Naive Bayes Model	Leisure	IBM
	5	Election results by County	Society	IBM
	6	Million dollar home sales in NE Mass late 2016	Economy & Business	Redfin.com
	7	Boston Crime data, 2-week sample	Society	City of Boston

## Load Data with PixieDust

- To create a pySpark DataFrame for one of the samples, just enter its number in the following command. For example, to load Set 6, Million Dollar Home sales, run the command:
  - `Home_df = pixiedust.sampleData(6)`

## Load Data with PixieDust

- You can also replace the number with a URL. If you have a CSV file online, access it by entering the URL in the parenthesis, like this:
  - Home\_df = pixiedust.sampleData(<https://openobjectstore.mybluemix.net/misc/milliondollarhomes.csv>)

## Load Data with PixieDust

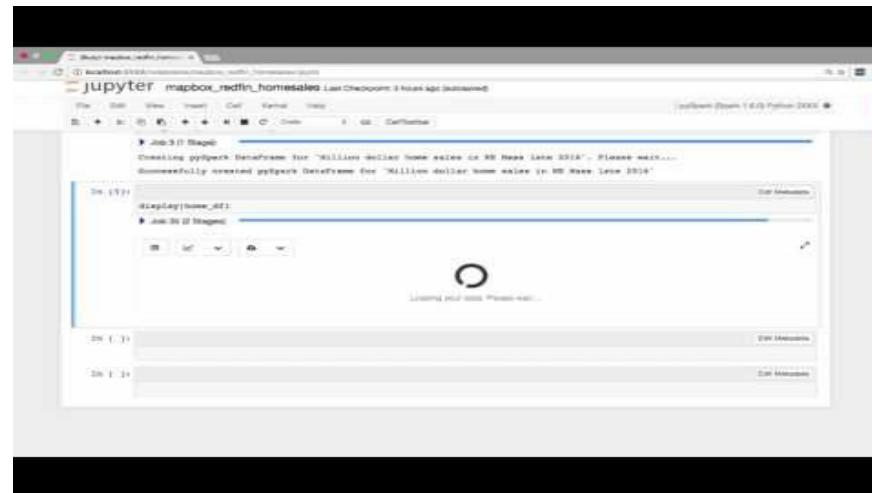
- Loading a CSV file from your local file system is equally simple. Just specify the file path inside the parenthesis
  - `Home_df = pixiedust.sampleData(file:///Users/bradfordnoble/pixiedust/data/nz/csv)`

## Load Data with PixieDust

- PixieDust provides these sample data sets as a convenience to help you get started fast
- To load or connect to your own data source, follow the steps you normally would from within a notebook
- Our team has created some notebook tutorials which show how to connect to Cloudant, Twitter, and other data sources
- Reference:
  - [Predict Flight Delays with Apache Spark MLLib, FlightStats, and Weather Data](#)
  - [Sentiment Analysis of Twitter Hashtags](#)

## Display Data with PixieDust

- PixieDust lets you visualize your data in just a few clicks. There's no need to write the complex code that you used to need to generate notebook graphs.
- Just call PixieDust's display() API and any lay person can render and change complex table and chart displays
- You can even choose from multiple rendering engines, without needing to know any of the code that makes them run



## Getting Started with PixieDust display()

- Once you've imported the PixieDust module, start with the data. Here's some sample code that creates a data frame

```
#import pixiedust display module
from pixiedust.display import *

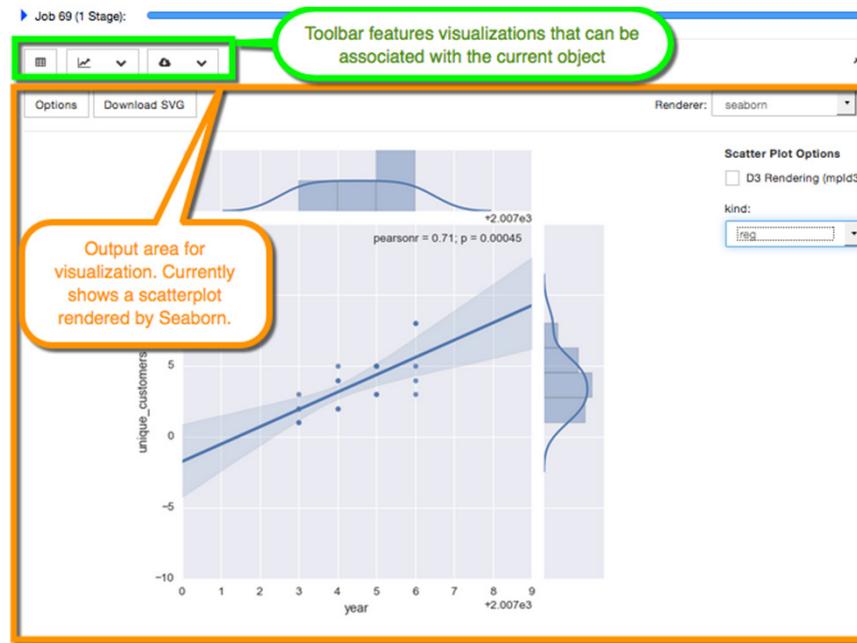
#Create a dataframe with Quarterly sales results
sqlContext = SQLContext(sc)
dd = sqlContext.createDataFrame(
    [(2010, 'Camping Equipment', 3),
     (2010, 'Golf Equipment', 1),
     (2010, 'Mountaineering Equipment', 1),
     (2010, 'Outdoor Protection', 2),
     (2010, 'Personal Accessories', 2),
     (2011, 'Camping Equipment', 4),
     (2011, 'Golf Equipment', 5),
     (2011, 'Mountaineering Equipment', 2),
     (2011, 'Outdoor Protection', 4),
     (2011, 'Personal Accessories', 2),
     (2012, 'Camping Equipment', 5),
     (2012, 'Golf Equipment', 5),
     (2012, 'Mountaineering Equipment', 3),
     (2012, 'Outdoor Protection', 5),
     (2012, 'Personal Accessories', 3),
     (2013, 'Camping Equipment', 8),
     (2013, 'Golf Equipment', 5),
     (2013, 'Mountaineering Equipment', 3),
     (2013, 'Outdoor Protection', 8),
     (2013, 'Personal Accessories', 4)],
    ["year", "zone", "unique_customers"])
```

## Getting Started with PixieDust display()

- Then, in a single command, you display that dataframe

```
#call a simple display api to visualize the data  
display(dd)
```

- `display()` looks up into its internal registry to build a list of visualizations that can handle a Spark DataFrame and generates a menu toolbar for each of them. The cell output looks like below:

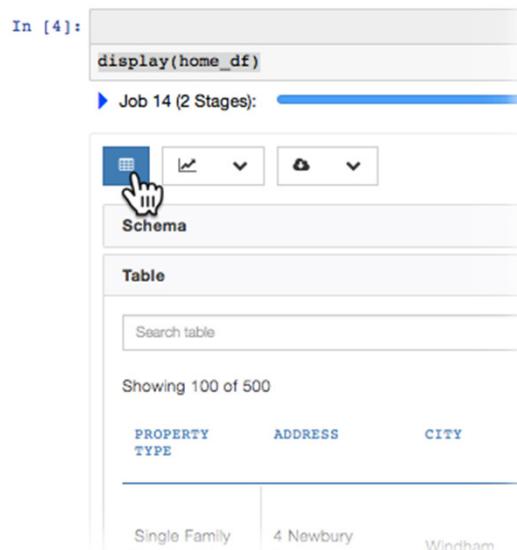


## Getting Started with PixieDust display()

- PixieDust spins up a robust user interface that contains all the features you need to create sophisticated visualizations in just a few clicks.
- It contains dropdown lists and dialogs you can use to change chart type, data content, grouping, and more
- **How does this all happen?**
  - `display()` uses `matplotlib` to generate the charts and then `mpld3` to transform the charts into D3 generated interactive charts that let users zoom, choose menus, see tooltips, and more.
- `display()` simplifies notebook charting in one important way: It takes only one cell to generate hundreds of visualization options
- Unlike traditional notebooks where you build a series of visualizations over several cells, PixieDust needs only one cell to generate an interactive widget which lets you turn knobs to explore the data in a myriad of ways

## Work with Tables, Charts, and Maps

- A great place to start is to view your data in a simple table format. To do so, click the **table** button:



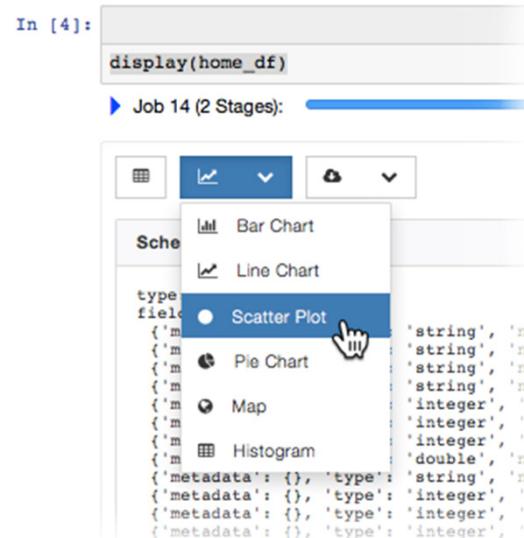
The screenshot shows a Jupyter Notebook cell with the code `In [4]: display(home_df)`. Below the code, a progress bar indicates "Job 14 (2 Stages)" is running. The interface then transitions to a table view. At the top, there are three buttons: a grid icon (selected), a line chart icon, and a scatter plot icon. Below these buttons is a dropdown menu labeled "Schema". The main area is titled "Table" and contains a search bar with the placeholder "Search table". Below the search bar, it says "Showing 100 of 500". A table header row is shown with columns labeled "PROPERTY TYPE", "ADDRESS", and "CITY". Underneath this header, a single data row is visible: "Single Family" under PROPERTY TYPE, "4 Newbury" under ADDRESS, and "Windham" under CITY.

- You see extended information about your Spark DataFrame in 2 view options:
  - Schema gives detailed information about the DataFrame schema
  - Table displays a sample of the data in an easy-to-read table format

# Work with Tables, Charts, and Maps

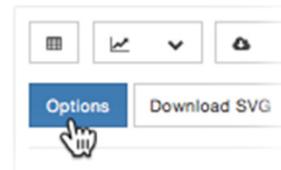
## ▪ Set Chart Content:

- Click the Chart dropdown menu and choose a chart type:



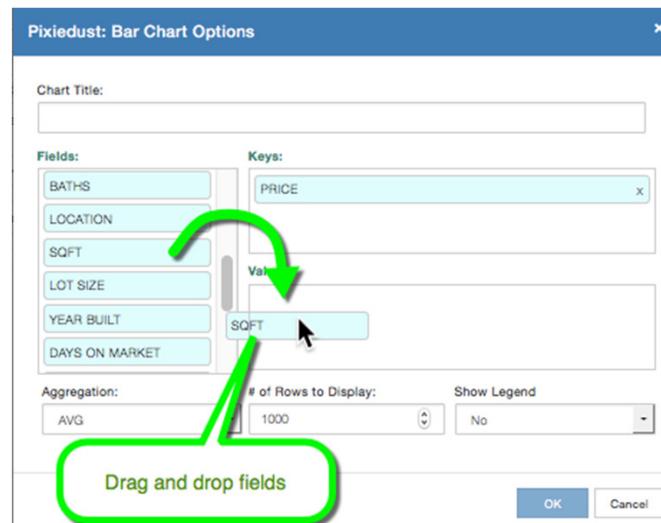
## Work with Tables, Charts, and Maps

- Configure the content of the chart by clicking the Options button



## Work with Tables, Charts, and Maps

- The options dialog that opens contains a set of common configuration choices for every chart, plus a set of options specific to the chart type you selected. For example, Bar Chart shows the following options dialog



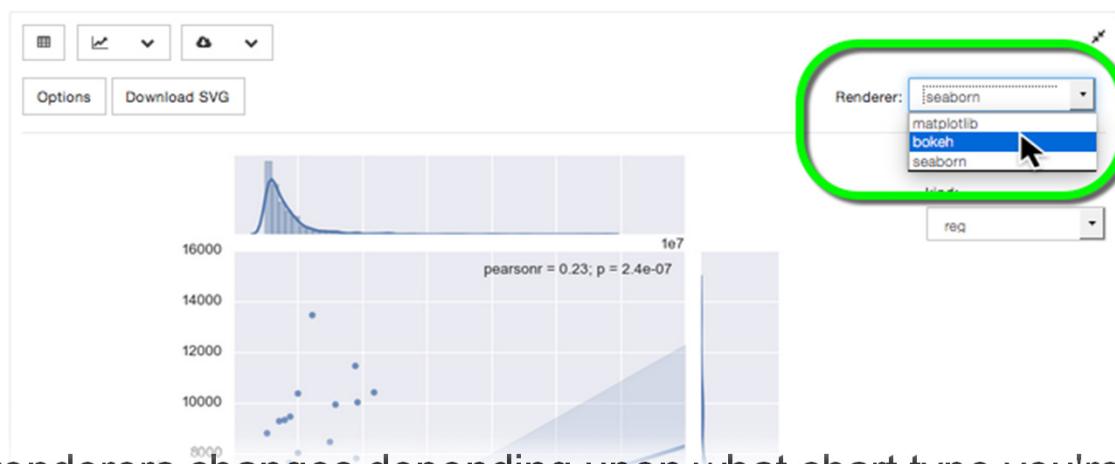
- To set keys and values, drag fields from the Fields list on the left and drop them where you want them

## Edit Cell Metadata Directly

- To directly access your option settings, you can edit cell metadata. From the Jupyter menu, choose **View > Cell Toolbar > Edit Metadata**
- On the upper right of your display() cell, click the **Edit Metadata** button
- Jupyter shows you the cell's JSON which you can edit and save
- Set these common options for every chart:
  - Chart Title: Enter an apt, descriptive title
  - Fields: List of available field names derived from your DataFrame schema
  - Keys: Field(s) to serve as the x-Axis
  - Values: Field(s) to serve as the y-Axis
  - Aggregation
    - SUM
    - AVG
    - MIN
    - MAX
    - COUNT

## Choose a Renderer

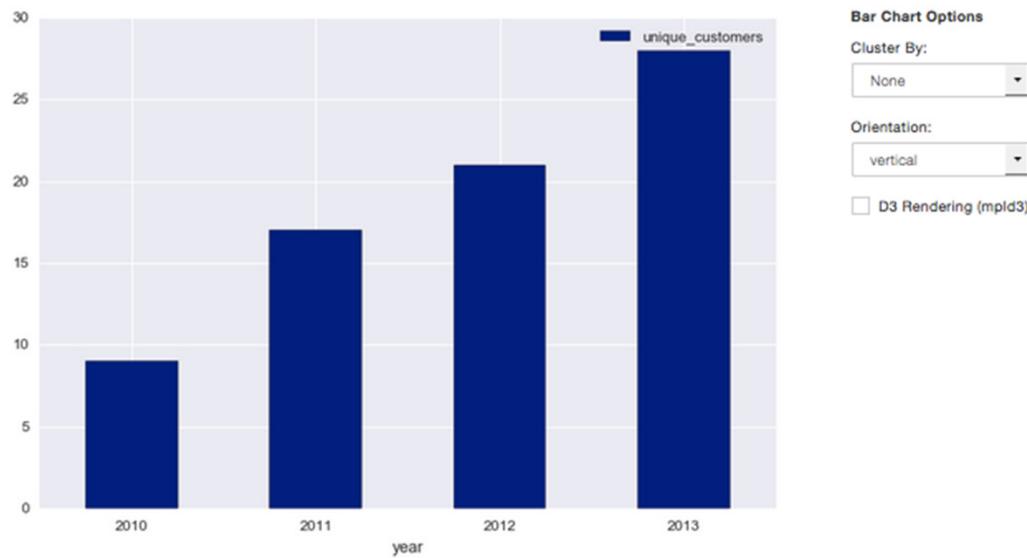
- PixieDust offers several different rendering engines you can use out-of-the-box to display your data



- The list of available renderers changes depending upon what chart type you're viewing
- Currently matplotlib, Seaborn, Brunel, Mapbox, and Google Maps are the built in renderers

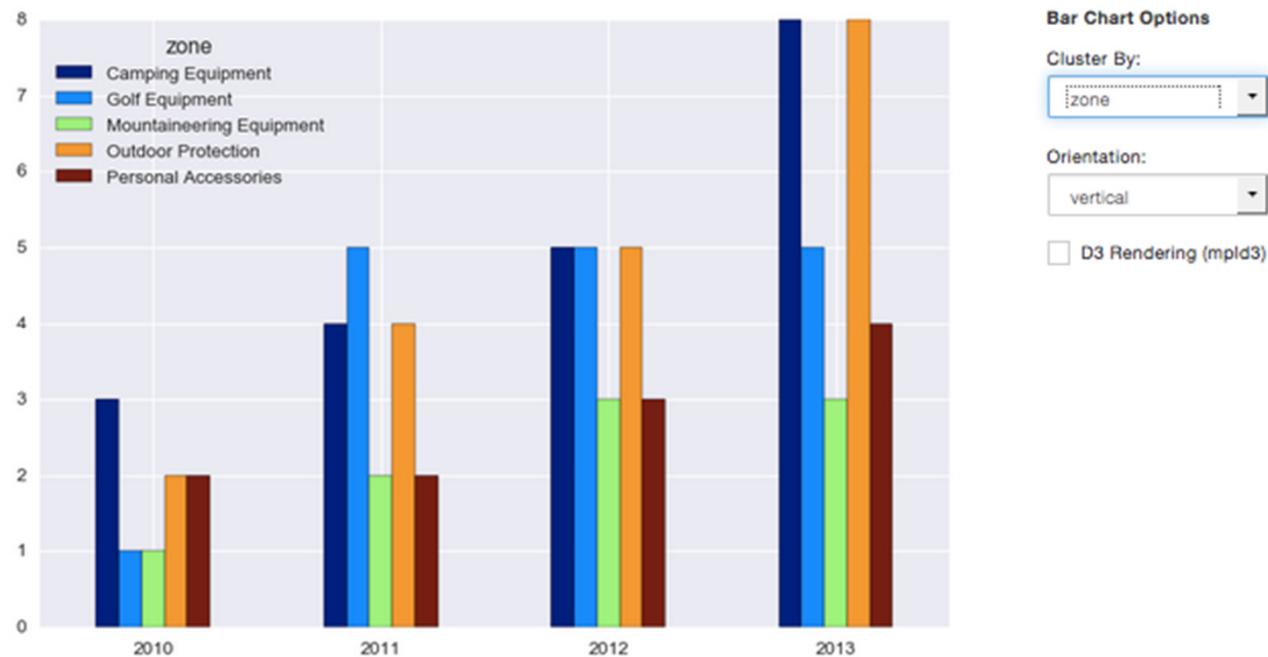
## Bar Chart

- Bar charts are handy for comparing items side-by-side
- In the Options dialog, set:
  - Keys: Choose a numeric field for the x-axis
  - Value: Choose a numeric field for the y-axis
  - Aggregation: Choose to sum



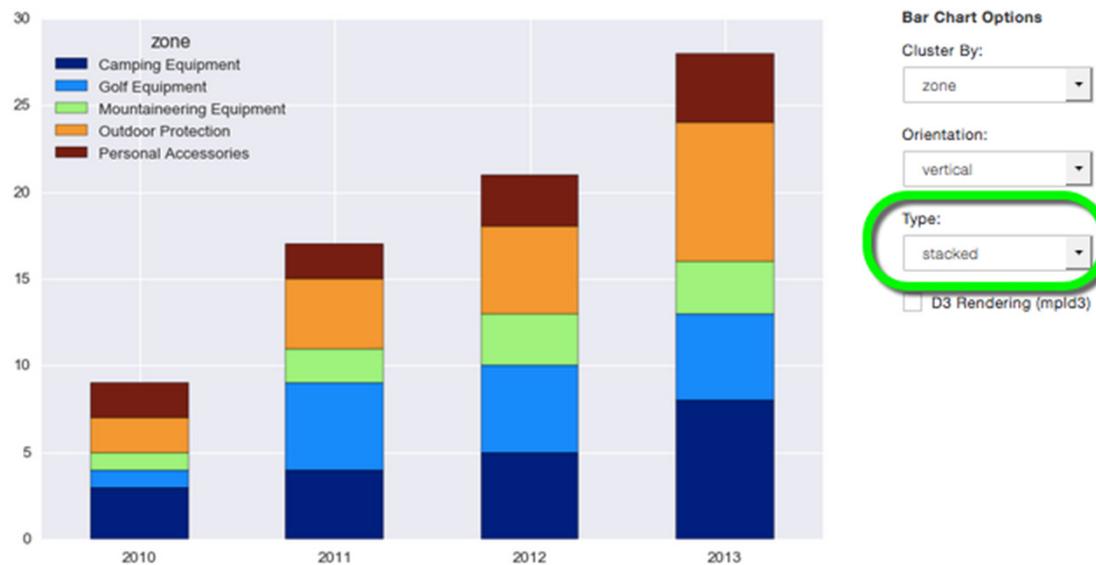
## Bar Chart

- To see another dimension, click the **Cluster by** dropdown and choose a field
- Here, clustering by zone, shows individual bars for each department/zone



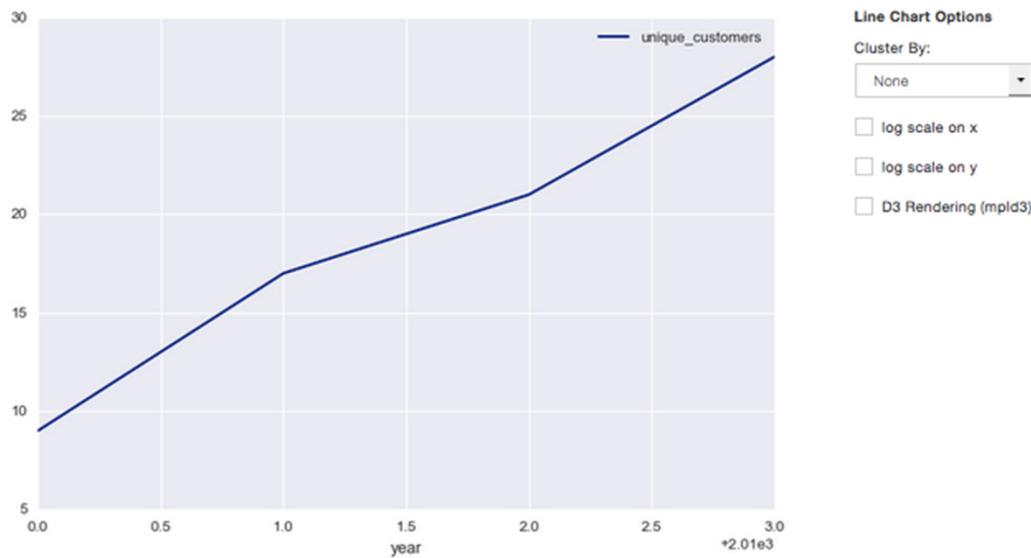
## Bar Chart

- You can show that cluster in different ways. Click the Type dropdown and choose one of the following:
  - Grouped** to see bars for each cluster grouped together, as you just saw in the previous
  - Stacked** to show clustered items in the same column split by color-coded segments or bands image
  - Subplots** to see each cluster in its own chart



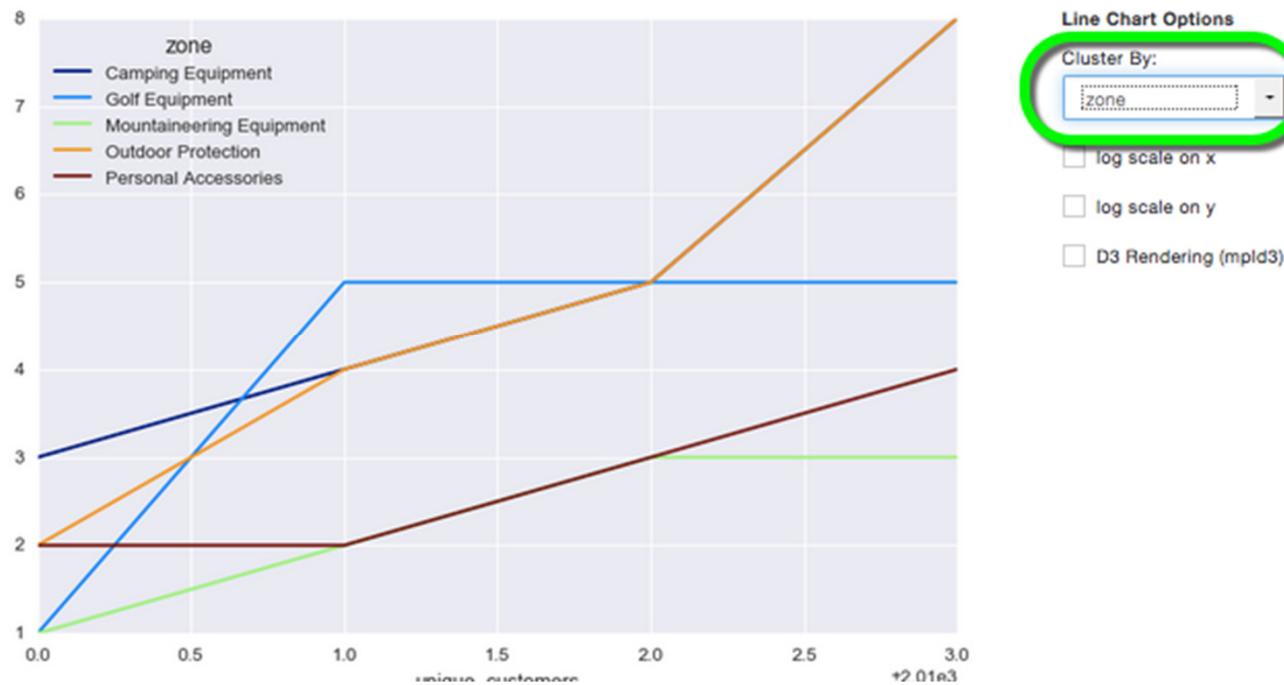
## Line Chart

- In the **Options** dialog, set:
  - Keys: Choose a numeric field for the x-axis
  - Value: Choose a numeric field for the y-axis
  - Aggregation: Choose to sum
- Like bar charts, line charts let you cluster results to see trends in an additional dimension. This chart shows customers rising steadily over time:



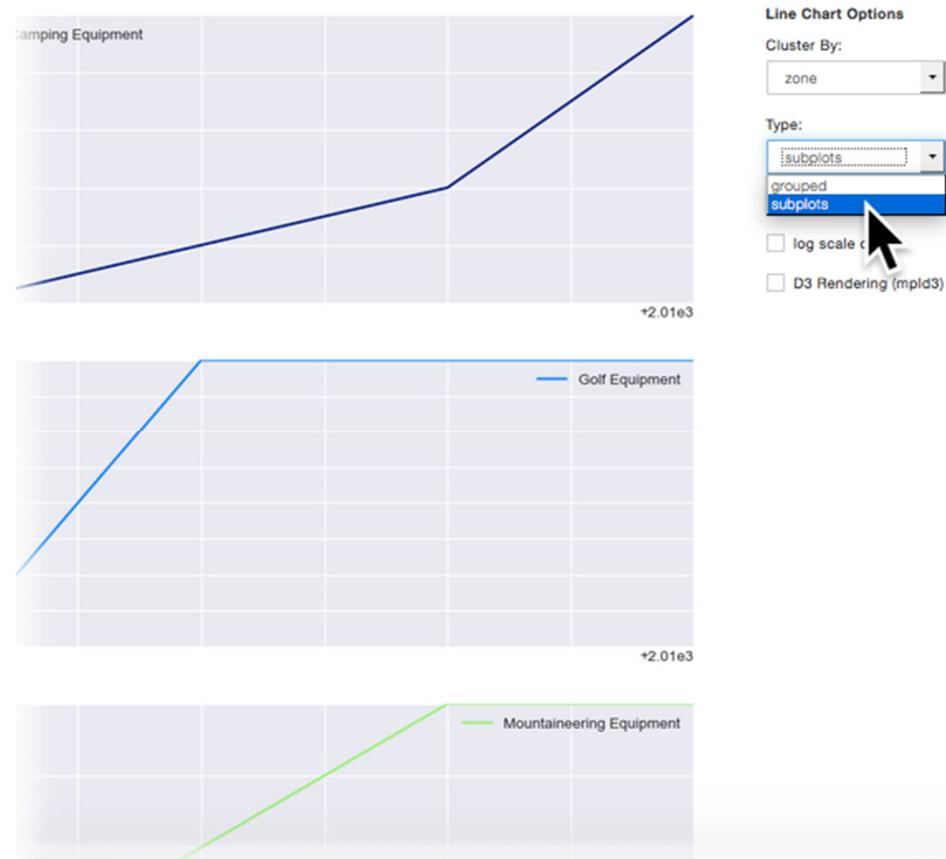
## Line Chart

- When you cluster the same chart by zone, you can see how each individual department/zone is doing



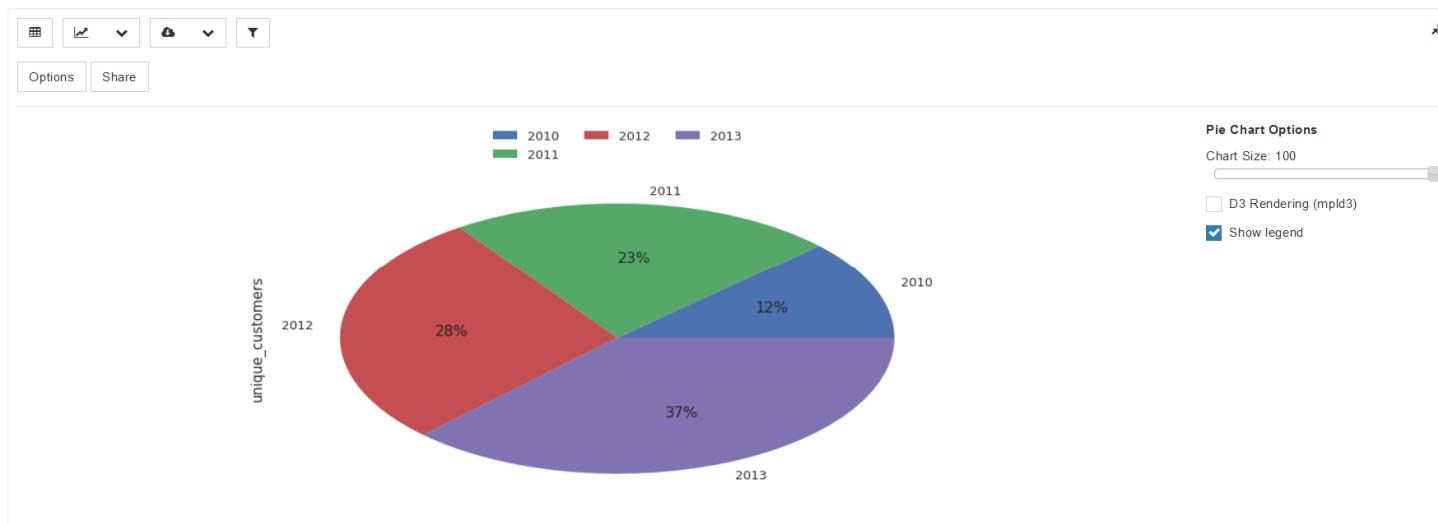
## Line Chart

- To show each cluster in its own chart, click the **Type** dropdown and choose **subplots**



## Pie Chart

- A pie chart is a circle graph which shows data as portions of a whole. In the **Options** dialog:
  - **Keys:** Choose the field that you want to be the labeled wedges of pie
  - **Values:** Choose a numeric field that you want to aggregate on. When you put more than one field in Value, you get a separate chart for each one
  - **Renderers:** matplotlib only

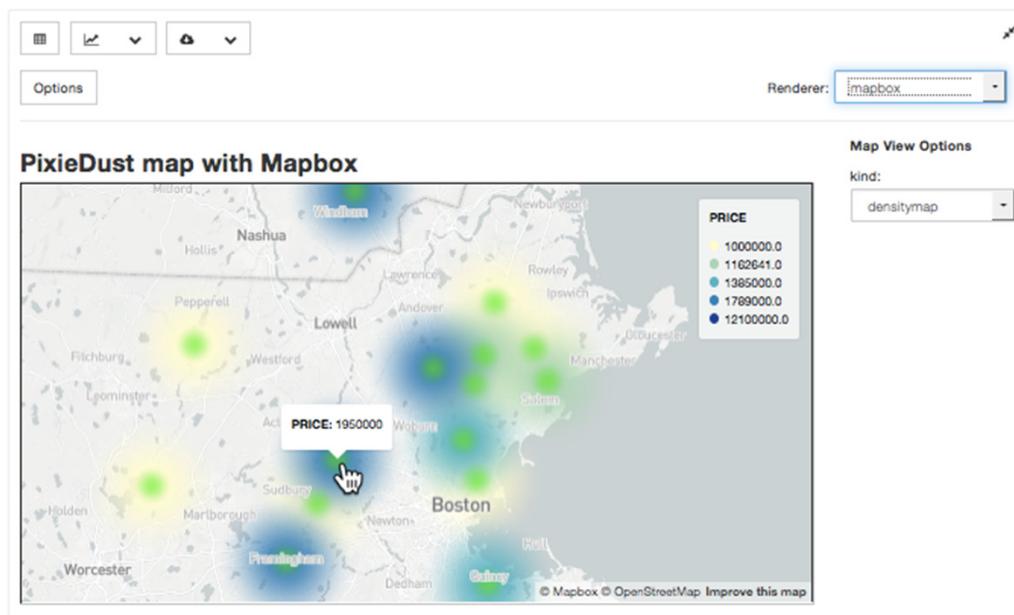


## Maps - Mapbox

- Configuring your map, depends upon which rendering engine you choose: Mapbox or Google Maps
- The Mapbox renderer lets you create a map of geographic point data. Your DataFrame needs at least the following 3 fields in order to work with this renderer:
  - A latitude field named latitude, lat, or y
  - A longitude field named longitude, lon, long, or x
  - A numeric field for visualization
- To use the Mapbox renderer you need a free API key from Mapbox
- You can get one from here:
  - <https://www.mapbox.com/signup/>
- You enter your key in the **Options** dialog box

## Maps - Mapbox

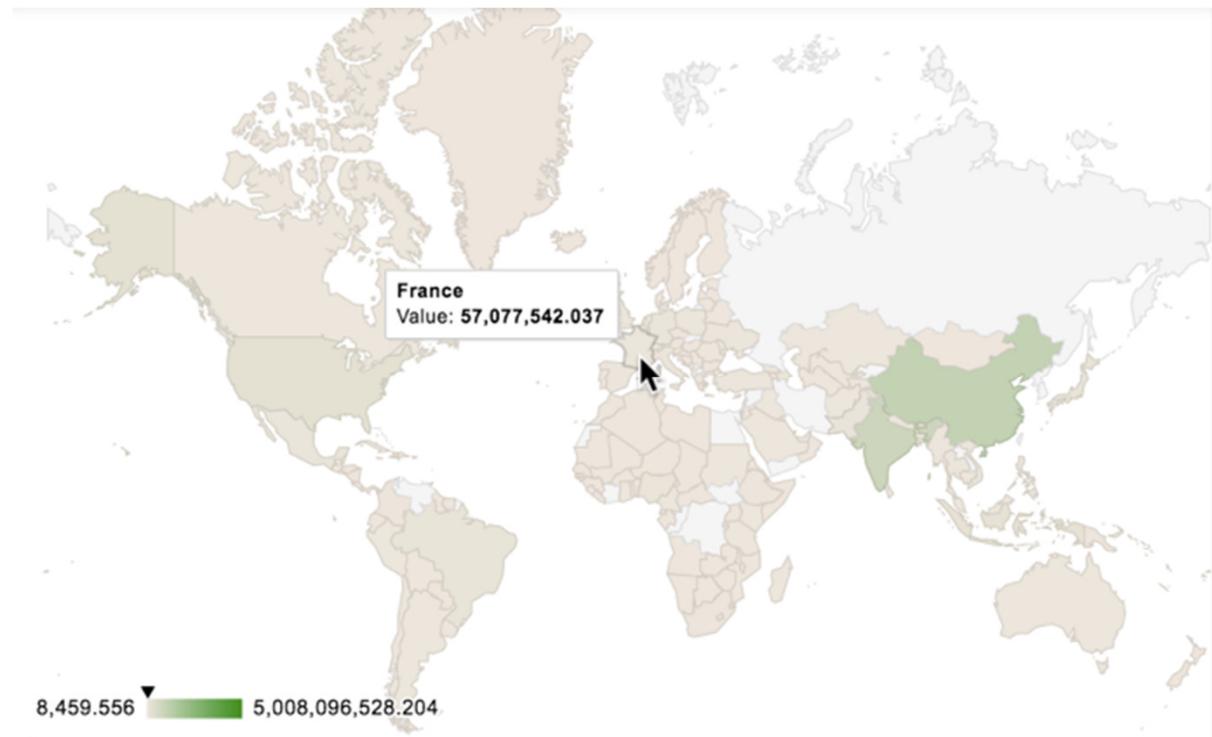
- In the **Options** dialog, drag both your latitude and longitude fields into **Keys**
- Then choose any numeric fields for **Values**
- Only the first one you choose is used to color the map thematically, but any other fields specified in Values appear in a pop-up information bubble when you hover your mouse over a data point on the map



## Maps – Google Maps

- In addition to mapping *geographic points* with Mapbox, Pixiedust also lets you use Google's API to create *GeoCharts*, which are maps that show region blocks identified in various ways
- To create a GeoChart in Pixiedust, open **Options** and drag the field that has place names into **Keys**
- Then for the **Values** field, choose any numeric field you want to visualize
- Within the **Display Mode** menu, choose
  - **Region**: to color the entire area of your named places e.g. countries, provinces, or states.
  - **Markers**: to place a circle in the center of the region which is scaled according to the data selected for the **Value** field.
  - **Text**: to label regions with labels like *Russia* or *Asia*
- The chart is shown on the next slide

## Maps – Google Maps



## Histogram

- Use a histogram if the values on your x-axis are numeric, like age or price, and you want to show them in ranges
- For example, here's PixieDust's Million Dollar Home Sales sample data set displayed in a histogram.
- Squarefeet ranges appear on the x-axis
- In **Options** choose:
  - **Values:** Choose a numeric field that you want to segment along the x-axis
  - **Renderers:** matplotlib, seaborn, or brunel
- The chart is shown on the next slide

# Histogram

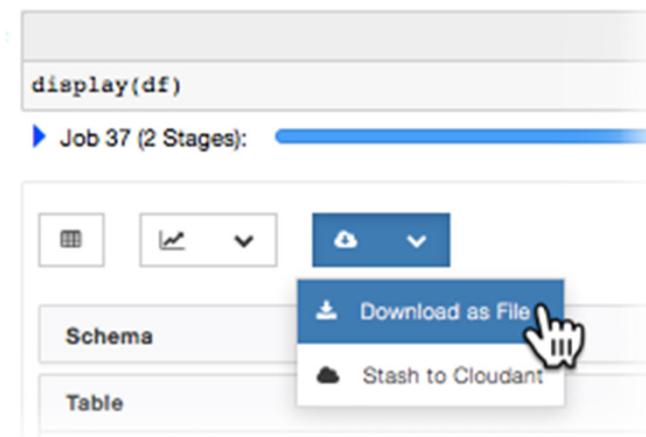


## Download Data with PixieDust

- PixieDust lets you download the data from your notebook
- If you've been playing with some charts, you can also save and download in SVG format.

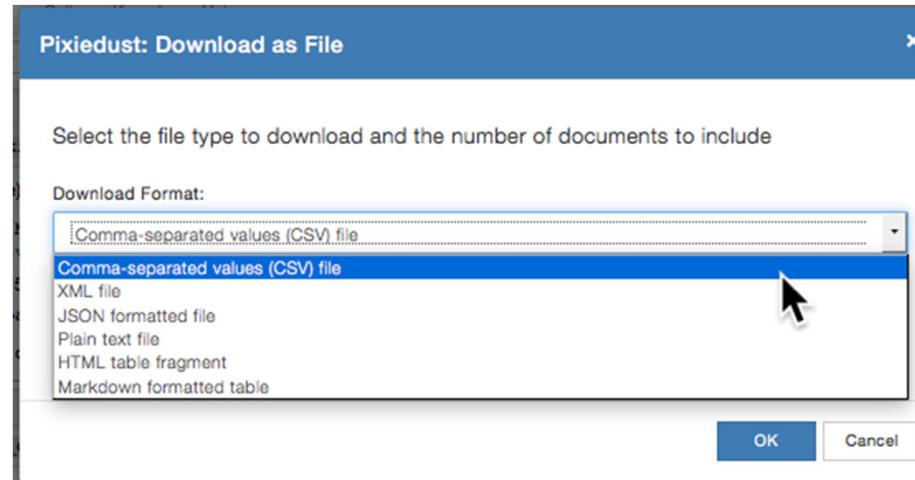
## Save Data to a File

- You can save a data set to a number of different file formats, including CSV, JSON, XML, and more.
- You do so within the user interface controls that the display API generates
  - Above the table and charts display, click the Download dropdown arrow. You see the following menu:



## Save Data to a File

- Choose **Download as File**, specify the format you want, and specify the number of records to download



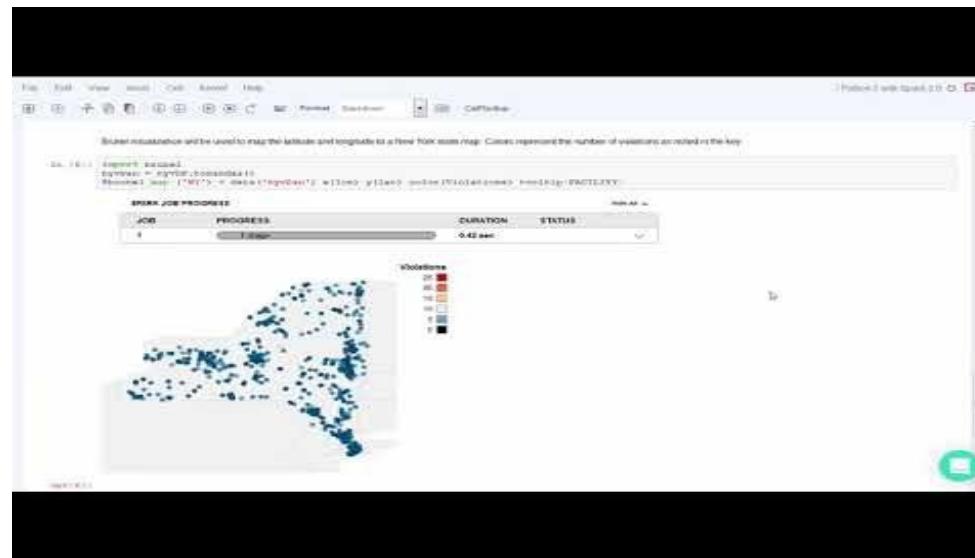
- Click **Ok**

## Logging with PixieDust

- When you're working in your notebook and something goes awry, the issue could be your data schema, the choices you made while building a chart, or something else.
- To understand what went wrong, check the logs!
- PixieDust comes complete with logging to help you troubleshoot issues. To invoke it, just enter the following command in a notebook cell and run:
  - `%pixiedustLog -l debug`
- PixieDust will then show you a log of recent activities

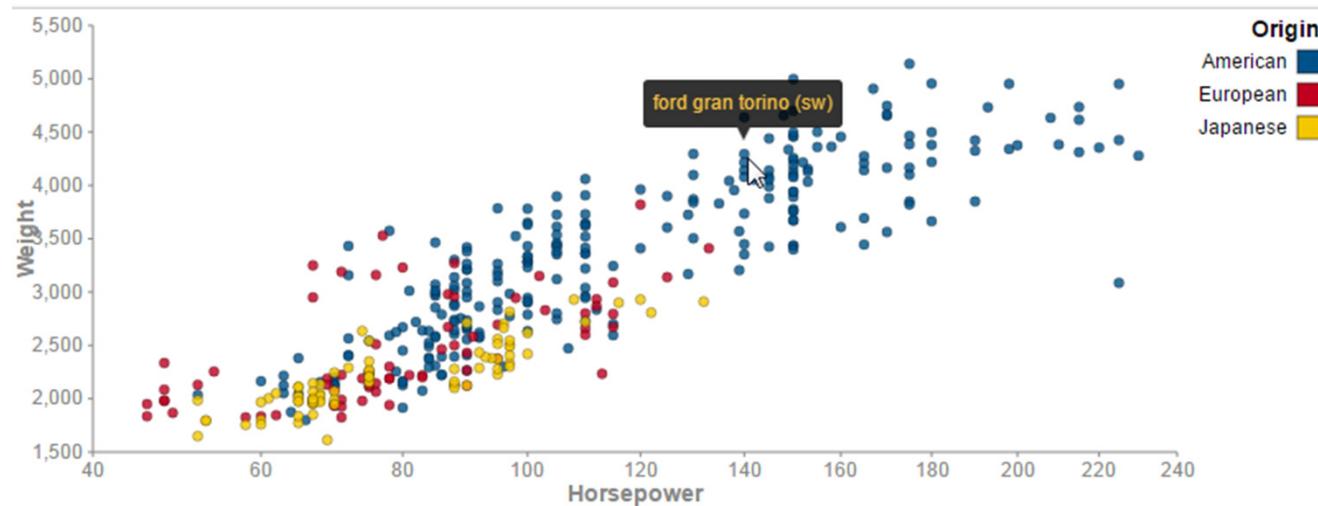
## Brunel Visualizations

- You can use the Brunel visualization language in Python notebooks to build interactive charts and diagrams
- Here is a quick demo video to showcase some of this packages capabilities



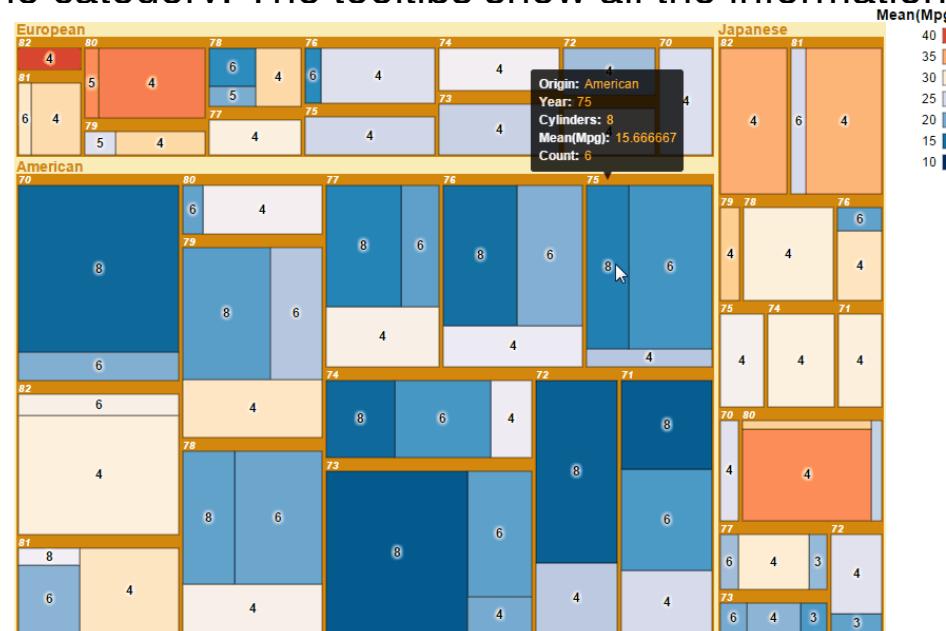
## Brunel Visualizations

- To get started with Brunel visualizations, open the sample notebook “*Visualize car data with Brunel*”
- Many types of Brunel graphs are interactive:
  - Meaning you can zoom and pan across the graph.
- The following scatter plot shows the relationship between the horsepower and the weight of the cars in a scatter plot. The color is based on the origin of the cars. The tooltips show the name of the cars



## Brunel Visualizations

- You can also create more complex graphs that display many dimensions
- For example, the following treemap groups vehicles by their origin, year of manufacture, and number of cylinders. The color indicates the average gas mileage of the vehicles in each block. The numbers in each block are the number of cylinders. The size of the blocks reflects the number of vehicles in the category. The tooltips show all the information.

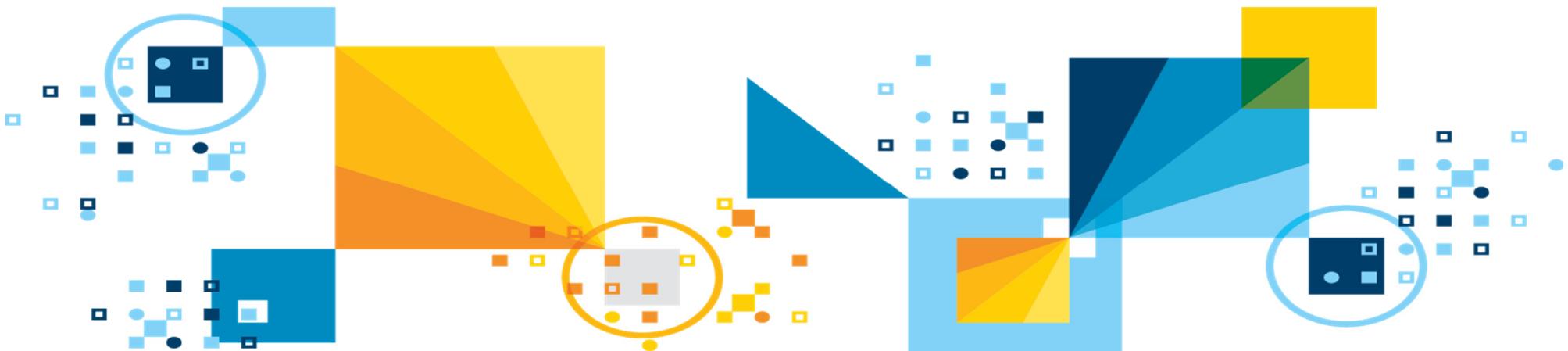


## Brunel Visualizations

- **To learn more:**

- Common charts and the Brunel syntax to create them: [Brunel Visualization Cookbook](#)
- Information about the Brunel visualization language: [Introduction to Brunel](#)
- Brunel blog: [Working Vis - Perspectives on Actionable Visualization](#)

# Watson Studio Local (WSL) Training Model Lifecycle



# IBM Cloud Private for Data – End-to-end AI workflow

## Collect, Connect & Access Data

## Govern, Search and Find Data

## Understand & Prepare Data for Analysis

## Build and Train ML/DL Models

## Deploy Models

## Monitor, Analyze and Manage

**Connect** and discover content from multiple data sources in the cloud or on premises. Bring **structured** and **unstructured** data to one toolkit.

**Find** data (structured, unstructured) and AI assets (e.g., ML/DL models, notebooks, Watson Data Kits) in the **Project Assets** with intelligent search and giving the right access to the right users.

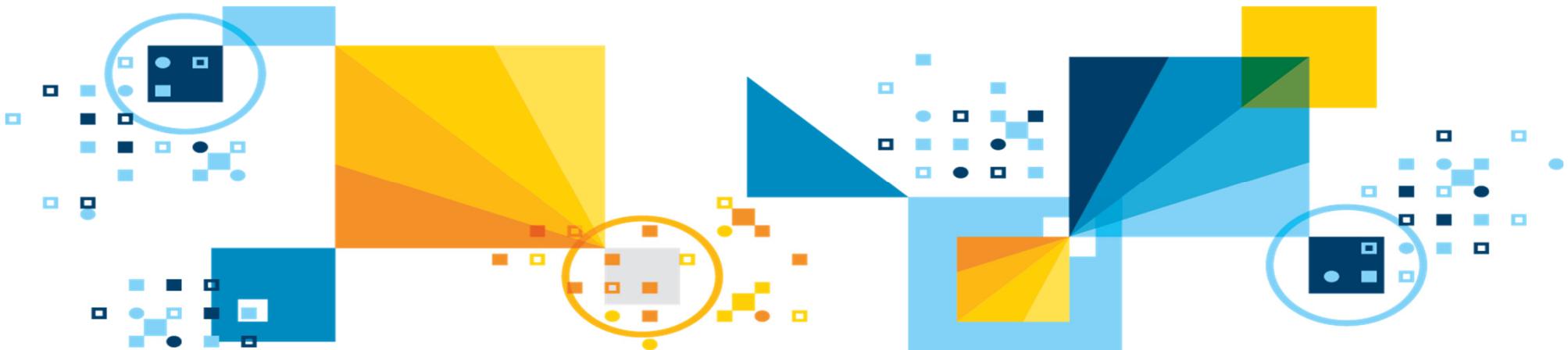
Clean and prepare your data with **Data Stage or ETL Tooling**. Use popular open source libraries to prepare unstructured data.

**Democratize** the creation of ML and DL models. Design your AI models **programmatically** or **visually** with the most popular **open source** and IBM ML/DL frameworks or leverage transfer learning on **pre-trained** models using **analytics tools** to adapt to your business domain. Train at scale on **GPUs** and

Deploy your models easily and have them **scale automatically** for online, batch or streaming use cases

Monitor the performance of the models in production and trigger automatic retraining and redeployment of models. Build **Enterprise Trust** with Bias Detection, Mitigation Model **Robustness** and Testing Service Model **Security**.

# Watson Studio Local (WSL) Training Model Management & Deployment



## Model Management and Deployment in Watson Studio Local

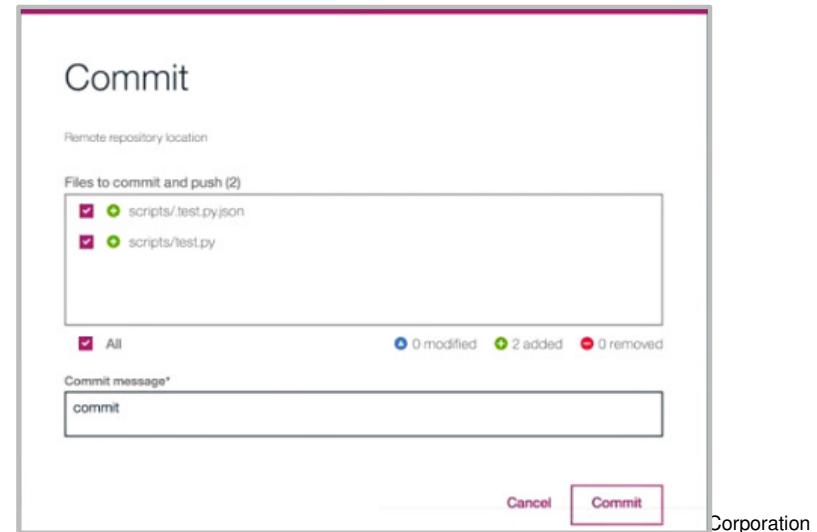
- To expose a checkpoint of assets to outside users, a Deployment Admin can create a project release and deploy the assets within it.
- A project release represents a project tag that can be launched as a production environment within Watson Studio Local.
- The Watson Machine Learning client requires Deployment Admin permissions to create a project release.
- Only the WSL administrator can assign **Deployment Admin** permissions to a WSL user.
- **Prerequisite:** At least one WSL deployment node must be installed and available for a project release to launch successfully. If no deployment node was installed, the WSL administrator can add a new deployment node (that meets the software requirements) from the Nodes page of the Admin Console.

## Model Management and Deployment in Watson Studio Local

- The Watson Machine Learning client provides the following tasks:
  - Creating a Project Release
  - Assign Members
  - Deploy Assets
  - Launch a Project Release
  - Manage a Project Release
  - Manage Deployments
  - View Data Sources
  - View Active Environments
  - Edit workers
- We will go over each of these tasks in the following slides

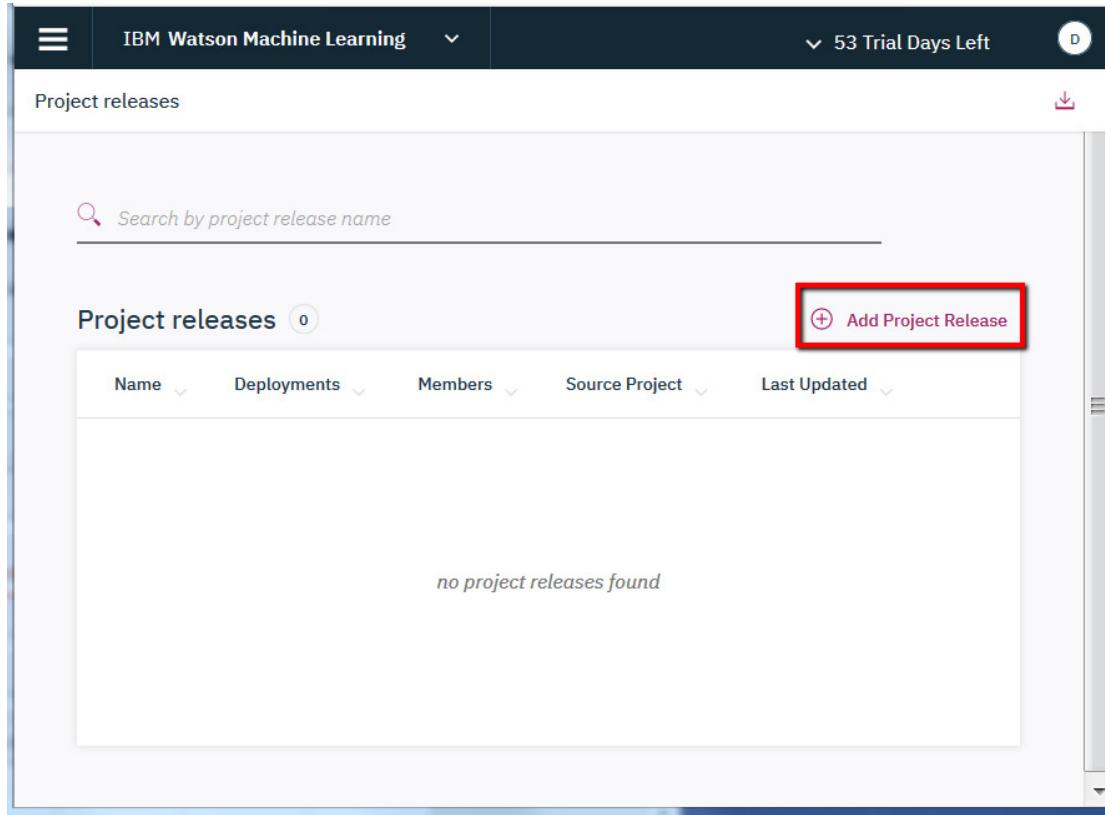
## Model Management & Deployment – Create a Project Release

- This is the first step required when wanting to deploy an asset
- A Deployment Admin creates a project release out of the state of a project at a given point in time as denoted by a user-defined tag.
- After the project release is created, assets within this project release can then be deployed
- If you create a project release from a Watson Studio Local source project, this source project must have a tag (specified from the Commit and push window)



# Model Management & Deployment – Create a Project Release

1. In the Watson Machine Learning client, go to the Project releases page and click Add Project Release.

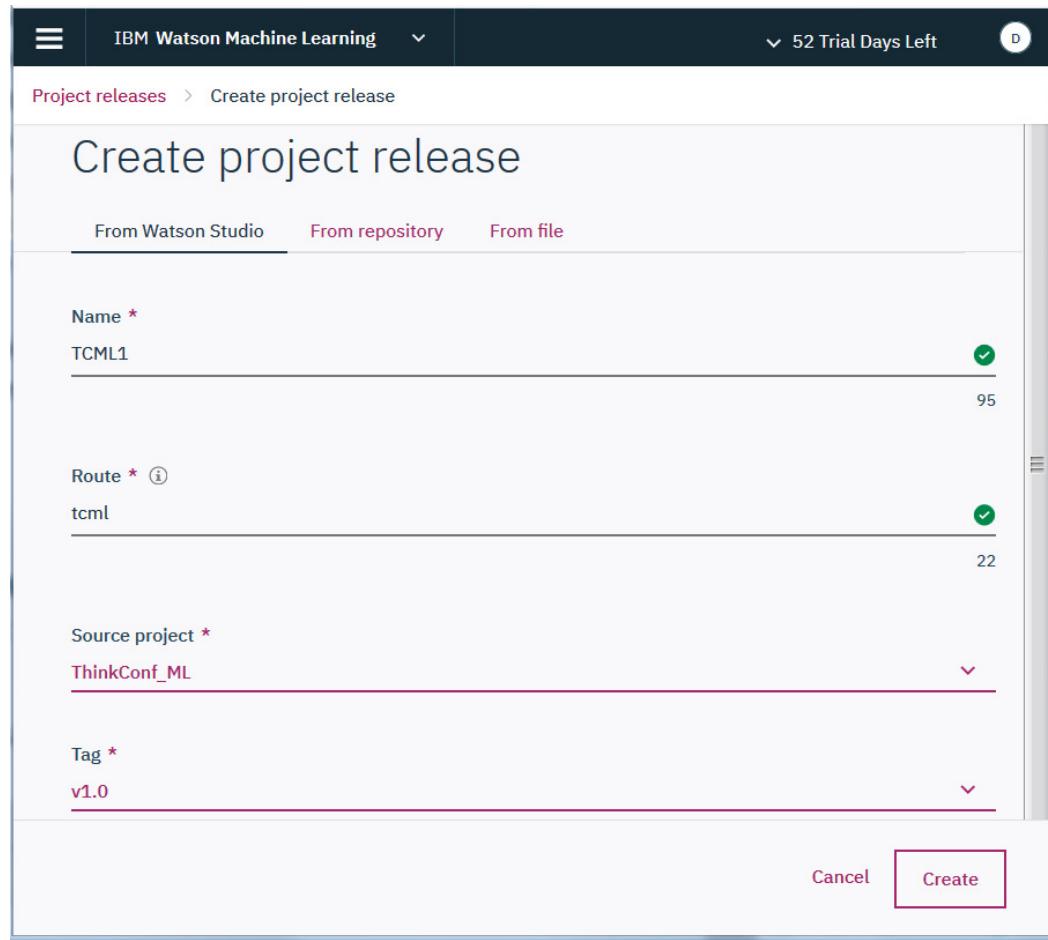


The screenshot shows the 'Project releases' page in the IBM Watson Machine Learning interface. At the top, there is a search bar labeled 'Search by project release name'. Below the search bar, the heading 'Project releases' is followed by a small circular icon with the number '0'. To the right of this, there is a red-bordered button with a plus sign and the text 'Add Project Release'. The main area of the page displays a table header with columns: Name, Deployments, Members, Source Project, and Last Updated. Below the table, a message reads 'no project releases found'.

## Model Management & Deployment – Create a Project Release

2. Click **From Watson Studio Local** and select a source project and tag in WSL to associate the release to. Click **From repository** to import a release from a Git repo (a personal access token is required). Click **From file** to import a pre-existing project from your local device (note that the file must be an exported project). Ensure the file name and release name do not contain a space.
3. Type in a name that is descriptive and unique. The name can contain hyphens but not special characters such as a period (.).
4. Type the route to use. The route is the unique ID for the project release, and is used within the deployments' REST paths and URLs. It can contain at least 2 and at most 26 lowercase alphanumeric characters and hyphens, and must start with a letter and end with a letter or number
5. Click Create
6. Watson Machine Learning creates an offline release, unlocked for editing.

# Model Management & Deployment – Create a Project Release



The screenshot shows the 'Create project release' form in the IBM Watson Machine Learning interface. The top navigation bar includes the 'IBM Watson Machine Learning' logo and a '52 Trial Days Left' indicator. The main title is 'Create project release'. Below it, there are three tabs: 'From Watson Studio' (selected), 'From repository', and 'From file'. The form fields are as follows:

- Name \***: TCML1 (highlighted with a red border)
- Route \***: tcml
- Source project \***: ThinkConf\_ML
- Tag \***: v1.0

At the bottom right are 'Cancel' and 'Create' buttons. A vertical sidebar on the right side of the form displays numerical values: 95, 22, and a downward arrow.

## Model Management & Deployment – Deploy Assets

- An Admin can deploy an asset into a project release
- The deployment action creates a read-only snapshot of the asset
- The following types of assets can be deployed:

Asset	Job	Web service	App	Notes
Notebooks	Y	N	Y	Only Jupyter notebooks can be deployed. Zeppelin and H2O Flows are not supported by Watson Machine Learning.
Models	N	Y	N	Requires that you associate a batch scoring script with it. You cannot deploy Custom Batch models as a web service (must be Custom Online).
R Shiny	N	N	Y	Deploys the R code inside of an R pod.
Flows	Y	N	N	Flows from SPSS Modeler.
Decision Optimization Models	N	Y	N	Provides a REST API to submit and execute optimization jobs.
Scripts	Y	Y	N	
Model groups	N	Y*	N	* Can be deployed as a web service group.

## Model Management & Deployment – Deploy Assets

- When the project release launches, each enabled deployment automatically generates either a permalink URL or REST API endpoint where the asset can be accessed:
  - **Job:** Generates a REST API endpoint to start, stop, and get status for that job. The job can be scheduled (but not unscheduled), and is visible to authenticated users only
  - **Web Service:** Generates a REST API endpoint for an external application to call. The web service is visible to authenticated users only

## Model Management & Deployment – Deploy Assets

- **Web Service Group:** Generates a REST API endpoint for an external application to call. The web service group is visible to authenticated users only. In the Edit test settings page, you can set the leader and enable the following routing configurations:
  - **Leader:** Allows API requests to the leader of the group. The Routing-Option header in the API request must be set to leader. This is the default.
  - **Specific Model:** Allows API requests to a specific model version in the group. The Routing-Option header in the API request must be set to specific.
  - **Random Model:** Allows API requests to a randomly selected model version in the group. The Routing-Option header in the API request must be set to random.
  - **All Models:** Allows requests to every enabled model version in the group. The Routing-Option header in the API request must be set to all.

## Model Management & Deployment – Deploy Assets

- **App:** Generates a URL that runs the asset code as an interactive UI session. You can specify the Shared with setting to be either:
  - **Anyone with the link:** Anyone outside person can view it
  - **Any authenticated user:** Only signed users can view it
  - **Deployment Admin:** Only the creator of the project release can view it.

## Model Management & Deployment – Deploy Assets

A web service generates a REST API endpoint for an external application to call.

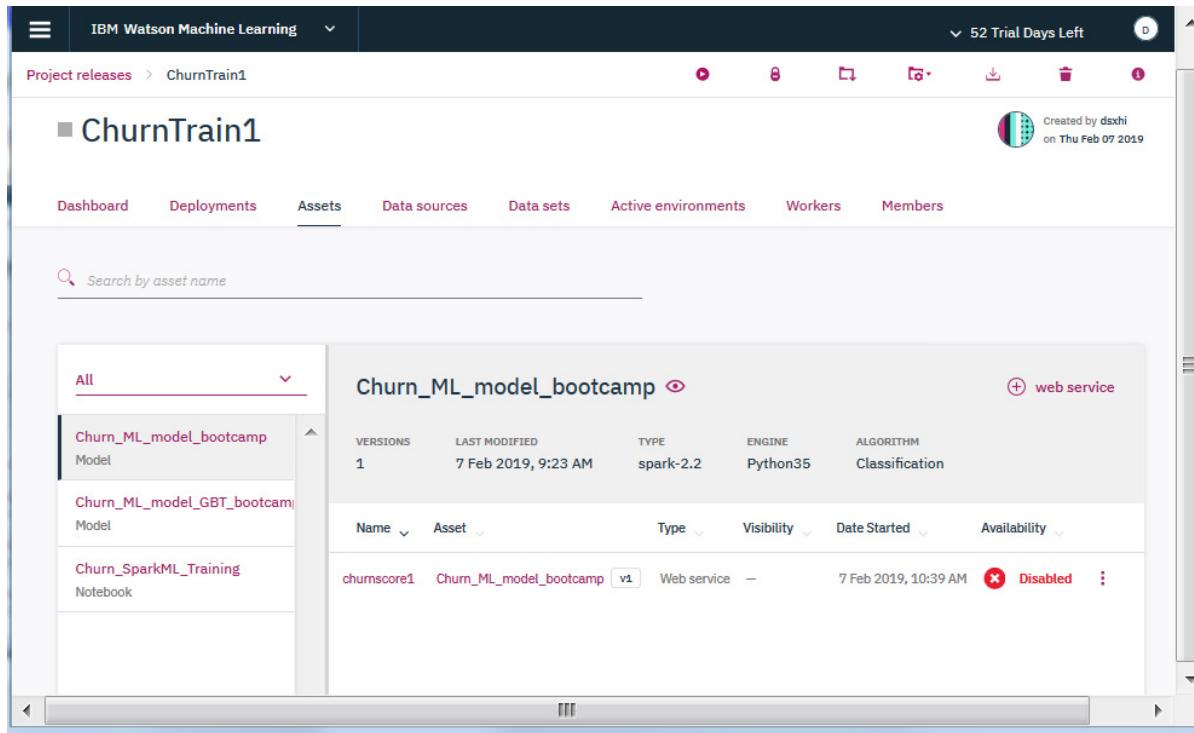
The following asset types can be deployed as a web services:

- Models
- Scripts

Restriction: You cannot deploy custom batch models as web service (must be custom online)

# Model Management & Deployment – Deploy Assets

- To deploy an asset, complete the following steps:
  1. In the Watson Machine Learning client, go to your project release and click the **Assets** tab.



The screenshot shows the Watson Machine Learning interface with the 'Assets' tab selected. A project named 'ChurnTrain1' is displayed. On the left, a sidebar lists assets: 'Churn\_ML\_model\_bootcamp' (Model), 'Churn\_ML\_model\_GBT\_bootcamp' (Model), and 'Churn\_SparkML\_Training' (Notebook). The main panel shows a deployed asset named 'Churn\_ML\_model\_bootcamp'. The details for this asset are:

VERSIONS	LAST MODIFIED	TYPE	ENGINE	ALGORITHM
1	7 Feb 2019, 9:23 AM	spark-2.2	Python35	Classification

Below this, a table lists deployment details:

Name	Asset	Type	Visibility	Date Started	Availability
churnscore1	Churn_ML_model_bootcamp v1	Web service	—	7 Feb 2019, 10:39 AM	<span>Disabled</span>

## Model Management & Deployment – Deploy Assets

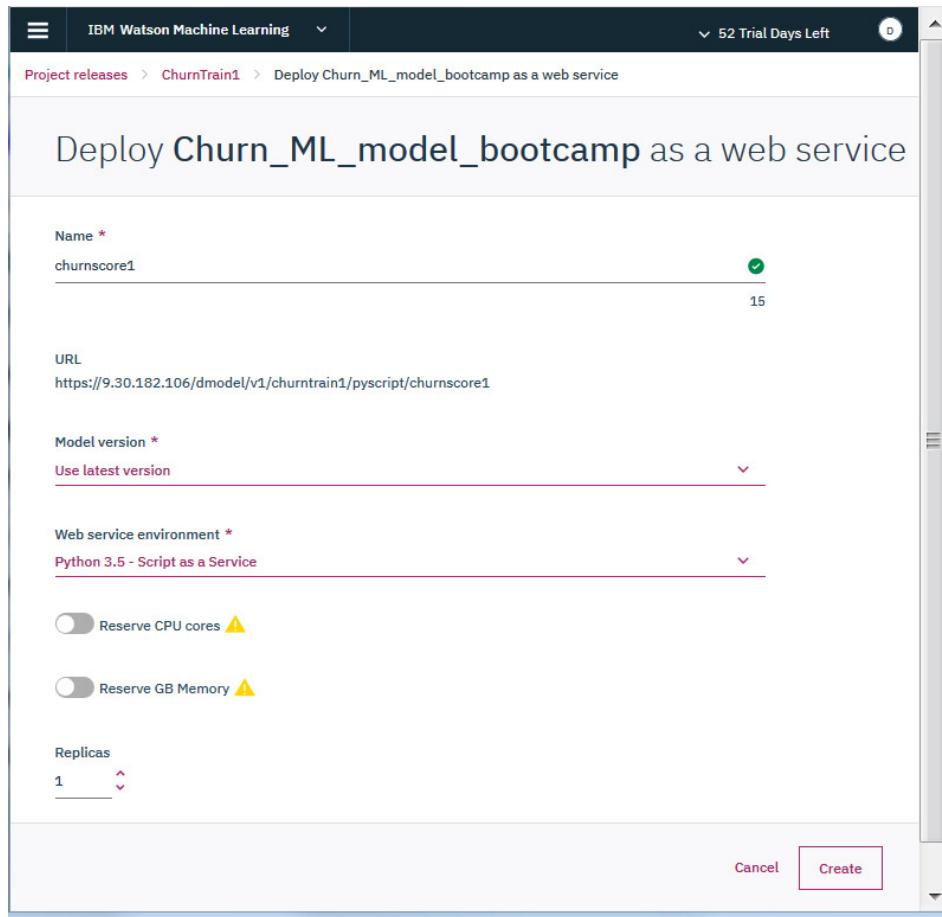
2. Select a model from the left side of the assets table, and then, in the right side, click the button to create a web service. The Create web service deployment window opens.

You are prompted with the following fields:

- Name – Specify a name for the asset that will be deployed. Use only lowercase alphanumeric characters and hyphens, up to 26 characters.
- URL – View the generated URL for the web service.
- Model version – Select from available versions. The latest is the default.
- Web service environment – Select from one of the generated scripts, or select Custom for a custom script that you generate.
- Reserve resources
- Replicas

3. Click the **Create** button

# Model Management & Deployment – Deploy Assets



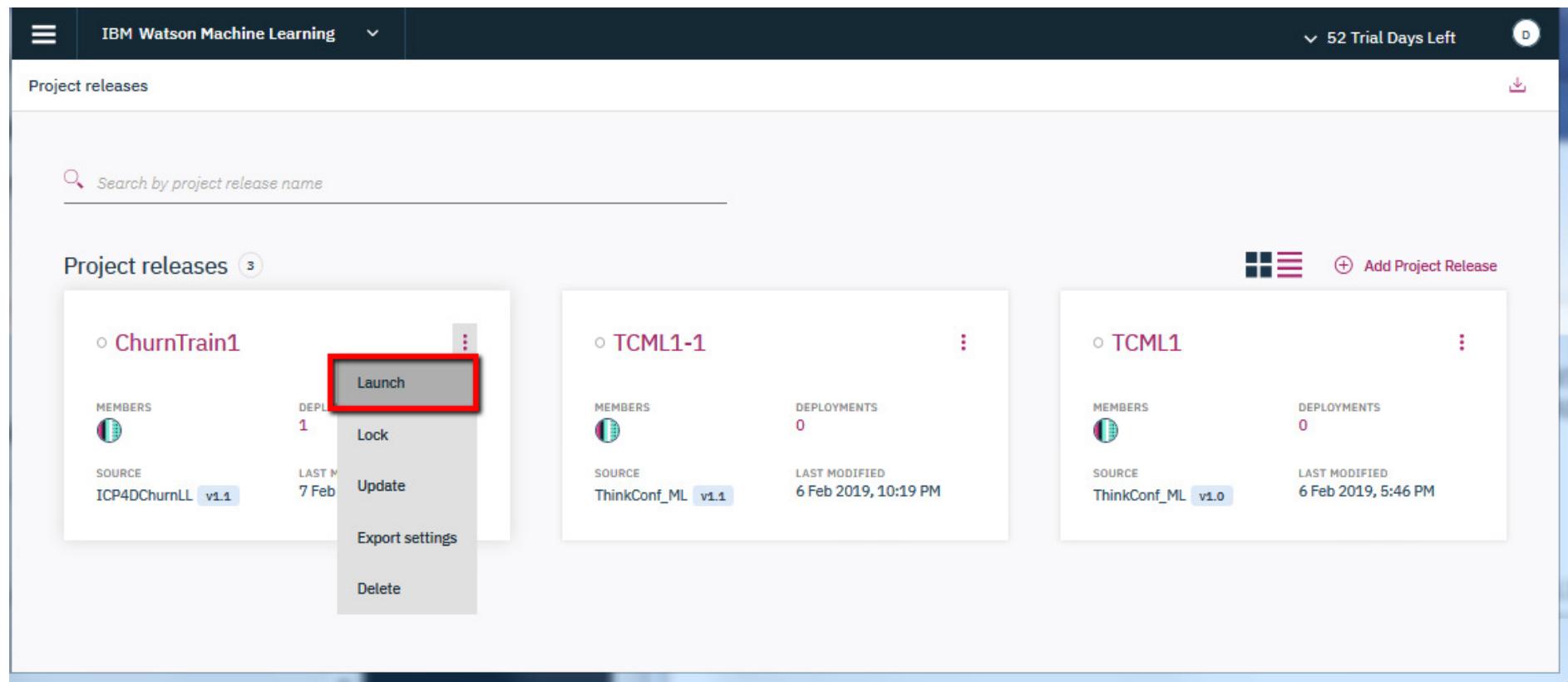
The screenshot shows the 'Deploy Churn\_ML\_model\_bootcamp as a web service' configuration page. The form includes fields for Name (churnscore1), URL (https://9.30.182.106/dmodel/v1/churntrain1/pyscript/churnscore1), Model version (Use latest version), Web service environment (Python 3.5 - Script as a Service), and deployment settings (Reserve CPU cores, Reserve GB Memory). The Replicas field is set to 1. At the bottom are 'Cancel' and 'Create' buttons.

- All deployments remain **disabled** until the project release is launched
- If the project release is already online, your newly created deployment starts immediately

## Model Management & Deployment – Launch a Project Release

- An Admin can click the Launch icon to bring the release online
- Wait about a minute for the pods to start and each enabled deployment to activate (you can monitor the status in the Active environments panel) before you click any actions for the deployment
- You might also see a similar time delay when you stop or start a script for model deployment
- If you deployed a Job:
  - Click it to view the REST API endpoint for an external application or to run the job on demand
  - When the release is online, you can click the API tab and submit a request to Start, Stop, or get Status for the job
  - You can also click generate code to copy/paste the curl command with the deployment bearer token in it.
  - You cannot take a release offline; you can only delete the release

# Model Management & Deployment – Launch a Project Release



The screenshot shows the IBM Watson Machine Learning interface for managing project releases. At the top, there's a navigation bar with the title "IBM Watson Machine Learning" and a dropdown indicating "52 Trial Days Left". Below the header, the main area is titled "Project releases" and features a search bar with the placeholder "Search by project release name".

The central part of the screen displays three project releases:

- ChurnTrain1**: This release has 1 member, 1 deployment, and was last modified on 7 Feb. It includes options to Launch, Lock, Update, Export settings, and Delete.
- TCML1-1**: This release has 1 member and 0 deployments. It was last modified on 6 Feb 2019, 10:19 PM.
- TCML1**: This release has 1 member and 0 deployments. It was last modified on 6 Feb 2019, 5:46 PM.

On the right side of the interface, there are icons for adding a new project release and a download button.

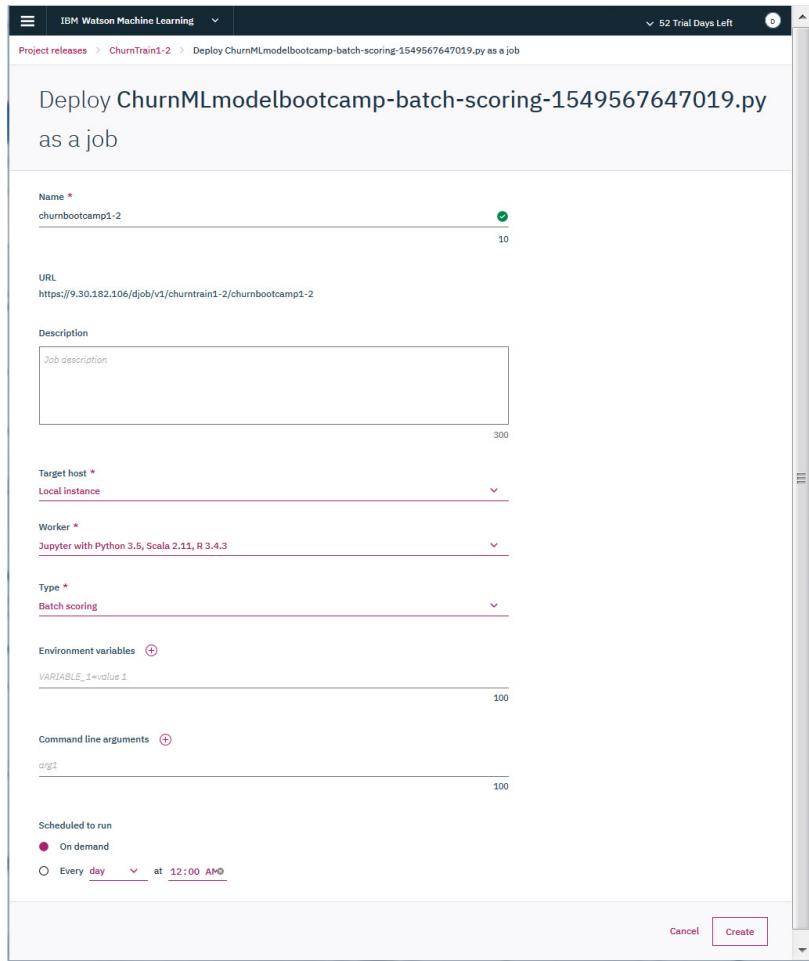
## Model Management & Deployment – Launch a Project Release

- If you deployed a Job:
  - A job deployment generates a REST API endpoint to start, stop, and get status for that job. The job can be scheduled (but not unscheduled), and is visible to authenticated users only.
  - The following asset types can be deployed as a job:
    - Scripts
    - Notebooks

## Model Management & Deployment – Launch a Project Release

- Deploy a job following this steps:
  - Open the Assets tab in the Project release details page. Select a script or notebook from the left side of the assets table, and then, in the right side, click the button to create a job. The Create job deployment window opens.
  - Specify a name for the job that contains lowercase alphanumeric characters and hyphens, up to 26 characters. Then specify other details for the job, including workers, job type, environment variables, and command line arguments. You can set up a schedule for the job.
  - After you create a job, you will see the job deployment in the deployments list.
  - To use the test API page in the Deployment details page, you must first bring the release online by clicking the Launch button from the Project release details page.

# Model Management & Deployment – Launch a Project Release



The screenshot shows the 'Deploy ChurnMLmodelbootcamp-batch-scoring-1549567647019.py as a job' page in the IBM Watson Machine Learning interface. The page includes fields for Name (churnbootcamp1-2), URL (https://9.30.182.106/djob/v1/churntrain1-2/churnbootcamp1-2), Description (Job description), Target host (Local instance), Worker (Jupyter with Python 3.5, Scala 2.11, R 3.4.3), Type (Batch scoring), Environment variables (VARIABLE\_1=value 1), Command line arguments (arg1), and Scheduled to run (On demand). A 'Create' button is at the bottom right.

- All deployments remain **disabled** until the project release is launched
- If the project release is already online, your newly created deployment starts immediately

## Model Management & Deployment – Launch a Project Release

- If you deployed a Web Service:
  - It will have a REST API endpoint for an external application to call
  - From the deployment details API tab, you can submit a JSON request or click generate code to copy/paste the curl command with the deployment bearer token in it
  - For an R script, ensure RStudio environment is in running state before you run the curl command (otherwise you might receive a ‘502 Bad Gateway’ error)
  - You cannot take a release offline; you can only delete the release.

## Model Management & Deployment – Launch a Project Release

- If you deployed a Job:
  - Click it to view the REST API endpoint for an external application or to run the job on demand
  - When the release is online, you can click the API tab and submit a request to Start, Stop, or get Status for the job
  - You can also click generate code to copy/paste the curl command with the deployment bearer token in it.
  - You cannot take a release offline; you can only delete the release

## Model Management & Deployment – Manage a Project Release

- **Update:**

- To pull in the latest assets from the source project and update the deployments with them, click **Update** next to the release
  - This action only updates modified assets and adds new assets to the release. It does not delete assets from the release

- **Export:**

- To export the deployment settings of a release (for use in another release), click **Export deployment settings** next to it
  - You can then upload these same deployment setting into another offline release by clicking the Import deployment settings icon and dragging the settings file into the Input pane.
  - Click the **Validate File** button to compare these imported settings with the current settings for your release
  - If the **Differences** look acceptable, you can click **Submit** to update your release manifest accordingly. Existing settings will be updated, new settings will be added, and unaffected settings will remain unchanged

## Model Management & Deployment – Manage a Project Release

- **Lock:**

- To lock a release so that no one can edit it, click **Lock** next to it
  - You can click **Unlock** to reopen the release for editing.

- **Delete:**

- To delete a release and all of its deployments, click **Delete** next to it
  - Alternatively, you can click the **Delete** icon from the release details page.

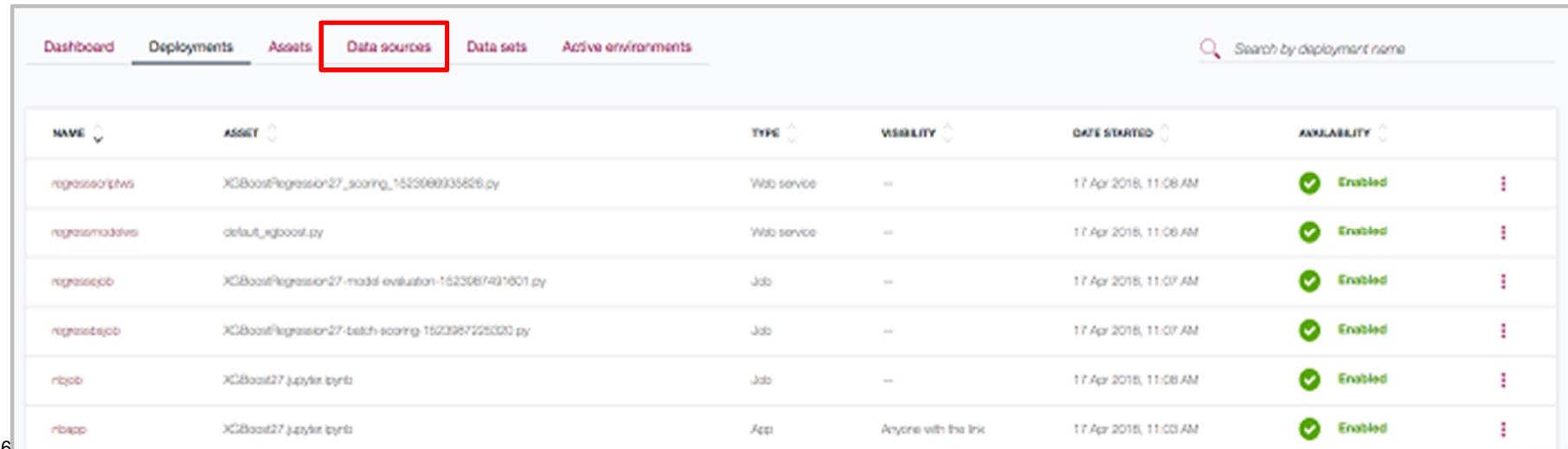
## Model Management & Deployment – Manage Deployments

- From the Deployments tab of your project release, an Admin, Developer, or Viewer can click a deployment for an overview of its details including request metrics captured at the function level
- Only the Admin can start, stop, delete, or redeploy any deployment in the release. Note that if you redeploy a new version of any of the assets, the URL does not change
- For an app, the Admin can edit its settings, edit its visibility, and view its route

Dashboard	Deployments	Assets	Data sources	Data sets	Active environments	Search by deployment name
Name	Asset	Type	Visibility	Date Started	Availability	
regressoripws	XGBoostRegression27_scoring-152300035826.py	Web service	--	17 Apr 2018, 11:08 AM	Enabled	
regressormodels	default_xgboost.py	Web service	--	17 Apr 2018, 11:08 AM	Enabled	
regressorjob	XGBoostRegression27-model-evaluation-1523067491601.py	Job	--	17 Apr 2018, 11:07 AM	Enabled	
regressorjob	XGBoostRegression27-batch-scoring-1523987225302.py	Job	--	17 Apr 2018, 11:07 AM	Enabled	
rjob	XGBoost27_jupyter.ipynb	Job	--	17 Apr 2018, 11:08 AM	Enabled	
rjob	XGBoost27_jupyter.ipynb	App	Anyone with the link	17 Apr 2018, 11:00 AM	Enabled	

## Model Management & Deployment – View Data Sources

- In the IBM Deployment Manager, click the Data sources to view your project data sources
- You can also edit the production data source connection string and credentials by clicking Edit credentials next to the data source
- Important: Database credentials must be input again after creating a project release. Otherwise, the data source might not work

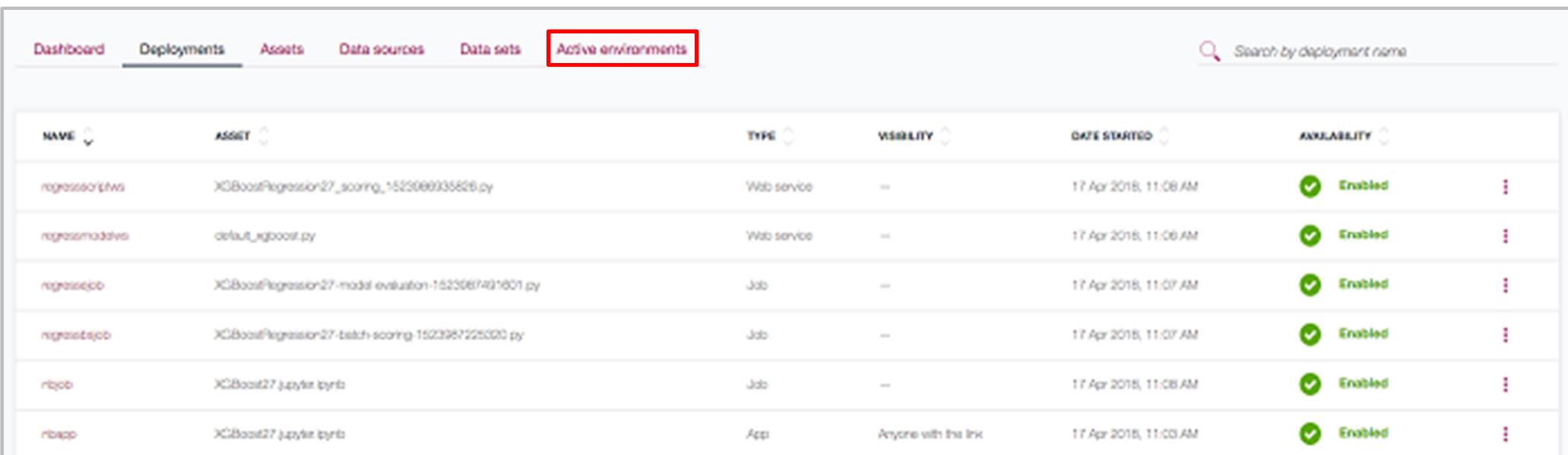


The screenshot shows the IBM Deployment Manager dashboard. The top navigation bar includes links for Dashboard, Deployments, Assets, **Data sources** (which is highlighted with a red box), Data sets, and Active environments. A search bar is located at the top right. Below the navigation is a table with deployment details. The columns are: NAME, ASSET, TYPE, VISIBILITY, DATE STARTED, and AVAILABILITY. The rows list various deployments, all of which are currently enabled.

NAME	ASSET	TYPE	VISIBILITY	DATE STARTED	AVAILABILITY
regressscriptws	XGBoostRegression27_scoring_1523069355826.py	Web service	--	17 Apr 2018, 11:08 AM	Enabled
regressmadlib	default_xgboost.py	Web service	--	17 Apr 2018, 11:06 AM	Enabled
regressjob	XGBoostRegression27-model-evaluation-1523067491601.py	Job	--	17 Apr 2018, 11:07 AM	Enabled
regressjob	XGBoostRegression27-batch-scoring-1523987225390.py	Job	--	17 Apr 2018, 11:07 AM	Enabled
rjob	XGBoost27_jupyter.ipynb	Job	--	17 Apr 2018, 11:08 AM	Enabled
rapp	XGBoost27_jupyter.ipynb	App	Anyone with the link	17 Apr 2018, 11:03 AM	Enabled

## Model Management & Deployment – View Active Environments

- In the Watson Machine Learning client, click the **Active environments** tab to monitor all runtime environments, workers, and services to WSL
  - This is synonymous with using the menu icon to click on **All Active Environments**



The screenshot shows the 'Active environments' tab selected in the navigation bar. The table lists seven entries:

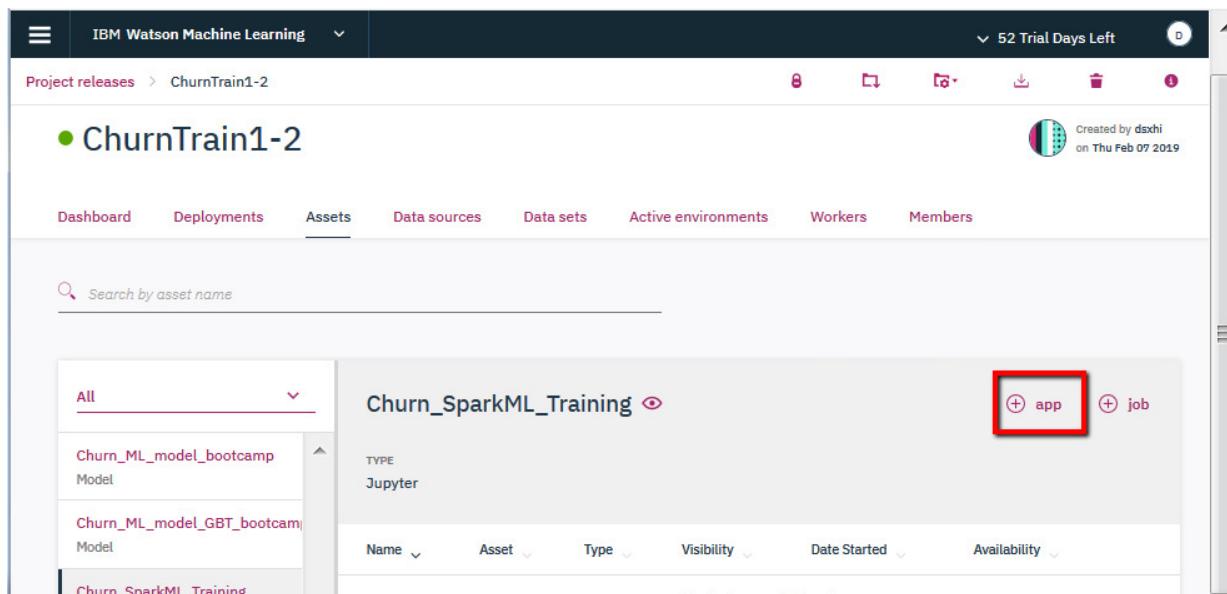
NAME	ASSET	TYPE	VISIBILITY	DATE STARTED	AVAILABILITY	⋮
regressor1ws	XGBoostRegression27_scoring_1523000035826.py	Web service	--	17 Apr 2018, 11:08 AM	Enabled	⋮
regressor2ws	default_xgboost.py	Web service	--	17 Apr 2018, 11:08 AM	Enabled	⋮
regressorjob	XGBoostRegression27-model-evaluation-15230067491001.py	Job	--	17 Apr 2018, 11:07 AM	Enabled	⋮
regressorjob	XGBoostRegression27-batch-scoring-1523987225000.py	Job	--	17 Apr 2018, 11:07 AM	Enabled	⋮
rjob	XGBoost27_jupyter.ipynb	Job	--	17 Apr 2018, 11:08 AM	Enabled	⋮
rapp	XGBoost27_jupyter.ipynb	App	Anyone with the link	17 Apr 2018, 11:03 AM	Enabled	⋮

## Model Management & Deployment – Deploy Asset as an App

- An app deployment generates a URL that runs the asset code as an interactive UI session
- The following asset types can be deployed as an app:
  - Shiny Servers
  - Jupyter Notebooks

## Model Management & Deployment – Deploy Notebook as an App

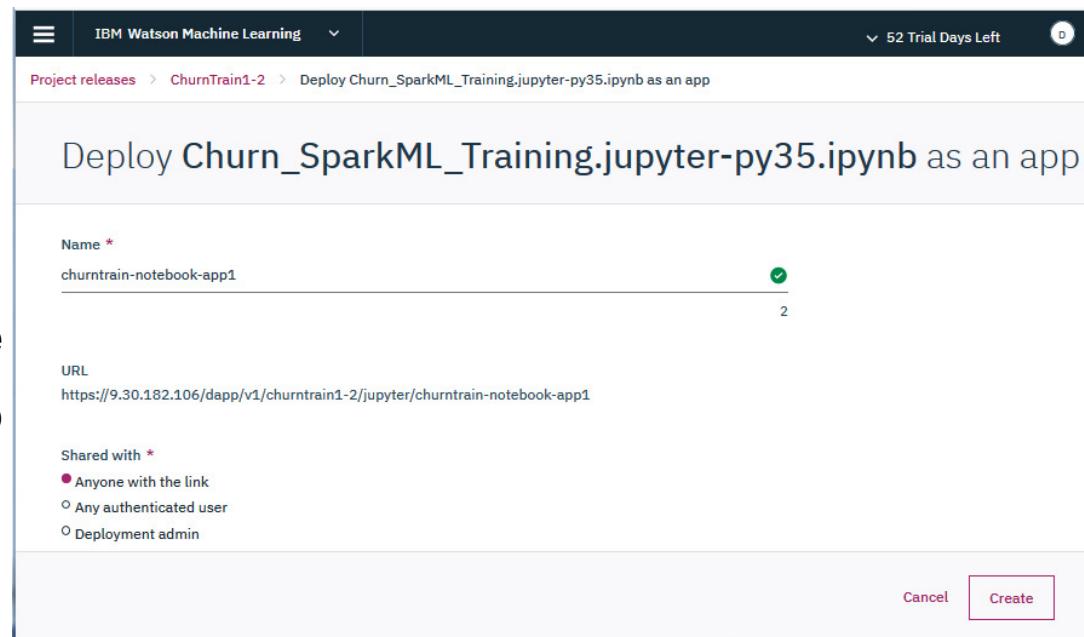
- **Requirement:** Ensure that you run your notebook before you deploy it as an app. Otherwise, cell results such as graphs do not appear
- You can navigate to the Assets tab in the Project release details page
- When a Jupyter notebook is selected in the left panel of the assets table, the right panel of the assets table provides the button to create an app, which brings you to the Create app deployment screen.



The screenshot shows the IBM Watson Machine Learning interface. At the top, there's a navigation bar with 'IBM Watson Machine Learning' and a dropdown showing 'Project releases > ChurnTrain1-2'. Below the navigation is a toolbar with icons for search, refresh, and other operations, followed by a message '52 Trial Days Left'. The main area has tabs for 'Dashboard', 'Deployments', 'Assets' (which is selected), 'Data sources', 'Data sets', 'Active environments', 'Workers', and 'Members'. A search bar says 'Search by asset name'. On the left, a sidebar lists assets: 'All' (selected), 'Churn\_ML\_model\_bootcamp Model', 'Churn\_ML\_model\_GBT\_bootcam Model', and 'Churn\_SparkML\_Training'. The main content area displays 'Churn\_SparkML\_Training' with a 'TYPE' of 'Jupyter'. To the right of the asset details are two buttons: '+ app' (highlighted with a red box) and '+ job'. At the bottom of the asset list are filters for 'Name', 'Asset', 'Type', 'Visibility', 'Date Started', and 'Availability'.

## Model Management & Deployment – Deploy Notebook as an App

- You are prompted with the following fields:
  - A **Name** for the asset to be deployed. It can contain lowercase alphanumeric characters and hyphens, up to 26 characters
  - A generated **URL** is shown
  - You can select the **Visibility** option from the radio group. Visibility determines who is app to interact with the app after it is deployed



The screenshot shows a deployment dialog in the IBM Watson Machine Learning interface. The title bar says "IBM Watson Machine Learning" and "52 Trial Days Left". The URL in the browser is "Project releases > ChurnTrain1-2 > Deploy Churn\_SparkML\_Training.jupyter-py35.ipynb as an app". The dialog form has the following fields:

- Name \***: churntrain-notebook-app1 (with a green checkmark icon)
- URL**: <https://9.30.182.106/dapp/v1/churntrain1-2/jupyter/churntrain-notebook-app1>
- Shared with \***:
  - Anyone with the link
  - Any authenticated user
  - Deployment admin

At the bottom right are "Cancel" and "Create" buttons, with "Create" being highlighted.

## Model Management & Deployment – View App

- When an app deployment is created and the project is live, you can view the app by going to the Deployments tab on the Release details page
- You can click the name of the app deployment and are brought to the app page.

## Model Management & Deployment – Deploy Asset as a Job

- A job generates a REST API endpoint to start, stop, and get status for that job
- The job can be scheduled (but not unscheduled), and is visible to authenticated users only
- The following asset types can be deployed as a job:
  - Scripts
  - Notebooks

# Model Management & Deployment – Deploy Script or Notebook as a Job

- You can navigate to the Assets tab in the Project release details page
- When a script is selected in the left panel of the assets table, the right panel of the assets table provides the button to create a job, which brings you to the Create job deployment page.

Deploy Cars Model Python-model-evaluation-1522797324247.py as a job

Name \*  
cars-model-evaluation 5

URL  
<https://9.30.140.31/djcb/v1/blg/cars-model-evaluation>

Description  
Job description 300

Type \*  
Model evaluation

Worker \*  
Jupyter with Python 2.7, Scala 2.11, R 3.4.3

Target host \*  
Local instance

Environment variables ⊕  
VARIABLE\_1=value 1 100

Command line arguments ⊕  
arg1 100

Scheduled to run  
 On demand  
 Every day ▼ at 12:00 AM ⌚

## Model Management & Deployment – Deploy Script or Notebook as a Job

- You are prompted with the following fields:
  - A **Name** for the job to be deployed. It might contain lowercase alphanumeric characters and hyphens, up to 26 characters
  - A generated **URL** is shown
  - An optional **description** is shown
  - You are able to select the **Worker** from the pull-down of available images
  - You are able to select the **job type** from the pull-down for the various job types
  - You have the option to provide **environment variables** for the job and **command line arguments**
  - You have the option of providing a time for the job to be **scheduled to run**

## Model Management & Deployment – Deploy Script or Notebook as a Job

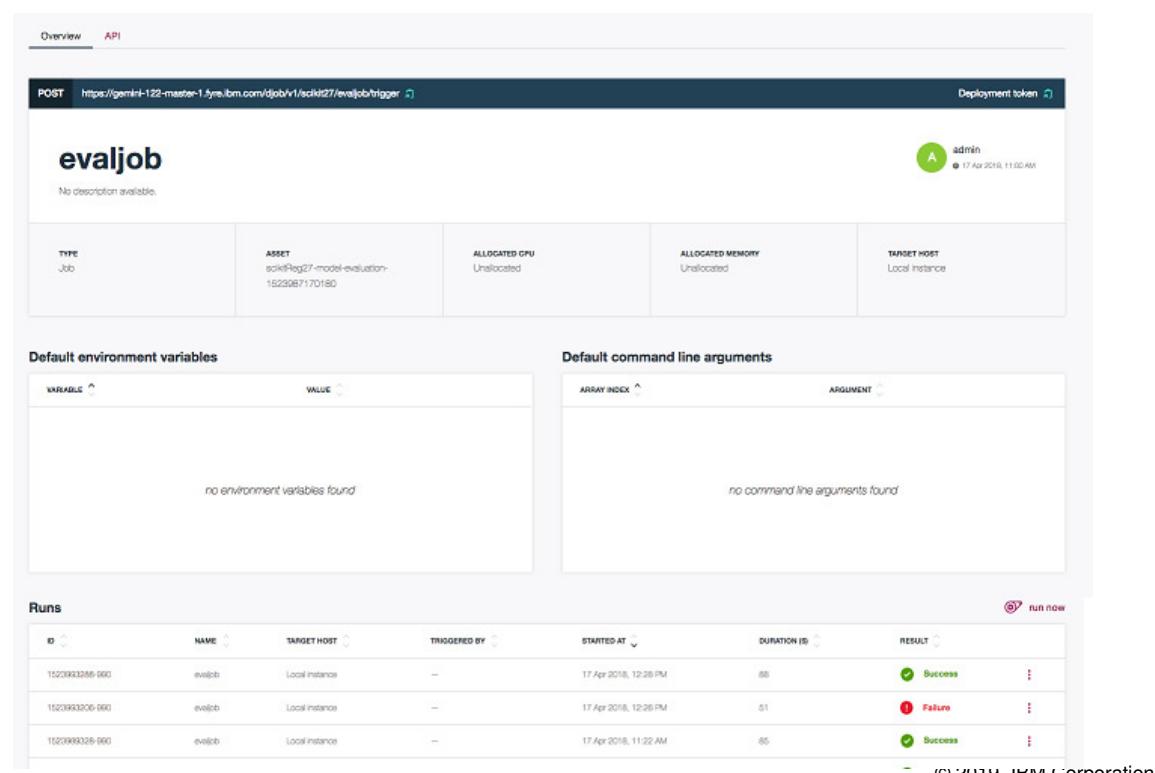
- After you create a job, you will see the job deployment in the deployments list
- To use the test API page in the deployment details page, you must first bring the release online by clicking the Launch button in the upper right of the Project release details page

## Model Management & Deployment – Manage a Deployed Job

- After you create a job deployment and selecting a job deployment from the deployments page, Watson Studio Local displays a Deployment details page
- The Deployment details page has two tabs:
  - Overview
  - API
- **Important:** You cannot start, stop, or get the status of a job if the release is not live

# Model Management & Deployment – Overview Tab

- A table of environments and arguments for the current job
- A table of all the runs for that particular job
- A button run now that triggers a job run



**evaljob**

No description available.

TYPE	ASSET	ALLOCATED GPU	ALLOCATED MEMORY	TARGET HOST
Job	ecelleReg27-model-evaluation-1522987170180	Unallocated	Unallocated	Local instance

**Default environment variables**

VARIABLE	VALUE
no environment variables found	

**Default command line arguments**

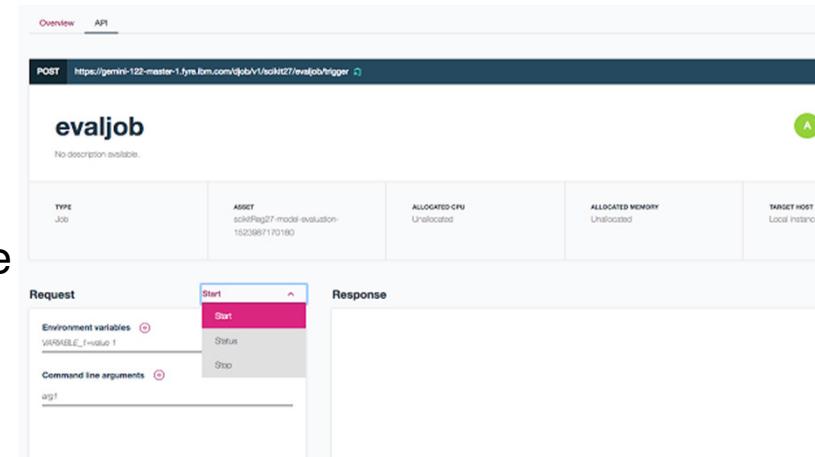
ARRAY INDEX	ARGUMENT
no command line arguments found	

**Runs**

ID	NAME	TARGET HOST	TRIGGERED BY	STARTED AT	DURATION (S)	RESULT	run now
1523993386-980	evaljob	Local instance	—	17 Apr 2018, 12:28 PM	88	<span style="color: green;">Success</span>	
1523993206-980	evaljob	Local instance	—	17 Apr 2018, 12:26 PM	81	<span style="color: red;">Failure</span>	
1523993326-980	evaljob	Local instance	—	17 Apr 2018, 11:22 AM	85	<span style="color: green;">Success</span>	

## Model Management & Deployment – API Tab

- In the API tab, you have three options of API calls to interact with a job with the pull-down menu:
  - Start calls the **trigger** api call to start a job run
  - Stop calls the **cancel** api call to stop a job run
  - Status calls the **status** api call to get information about the status. Note if you do not supply a run ID in the text box when you call the status, the API call gets all the runs for the current job
- A **generate code** button, which displays the proper curl command based on the selected status and the data in the **input** box
- A result box that shows the **JSON** output from the API call

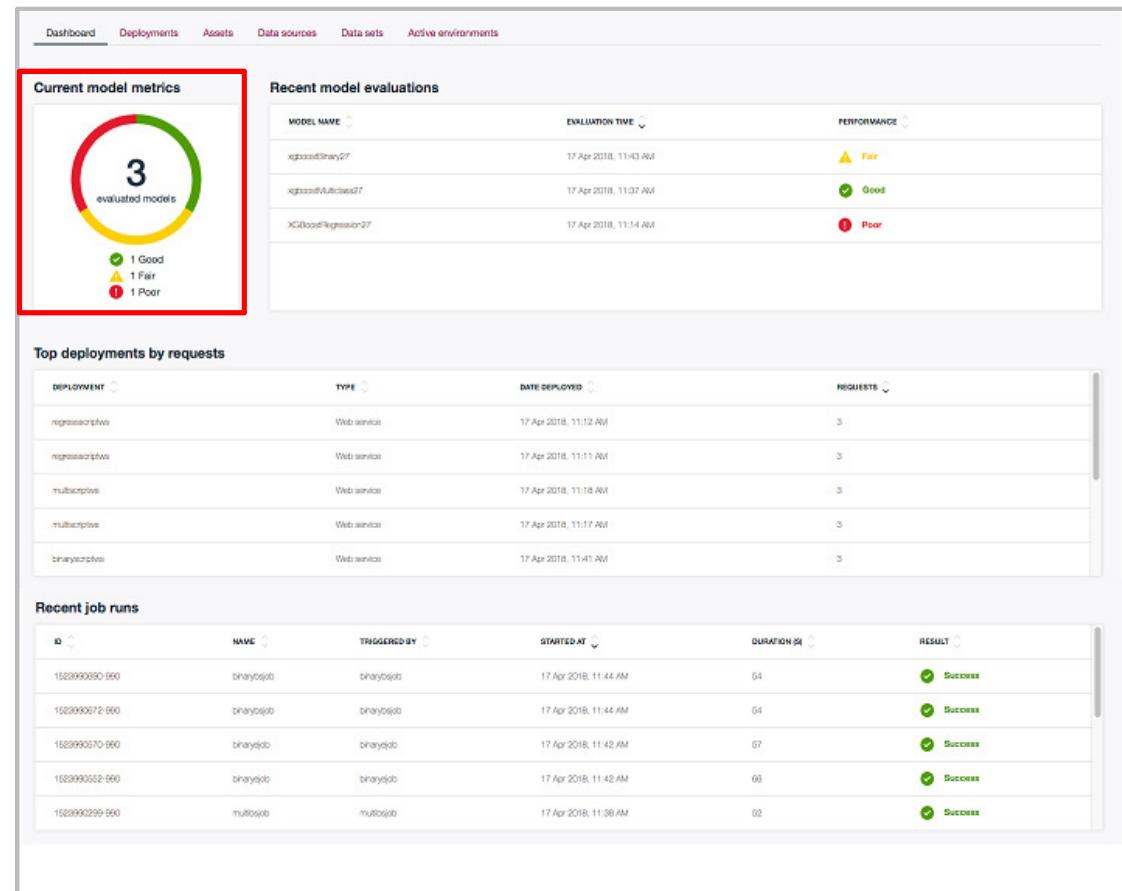


## Model Management & Deployment – Project Release Dashboard

- Deployment dashboard provides an overview about the release
- It has five components:
  - Current Metrics
  - Recent Model Evaluations
  - Top Deployments by Requests
  - Recent Job Runs

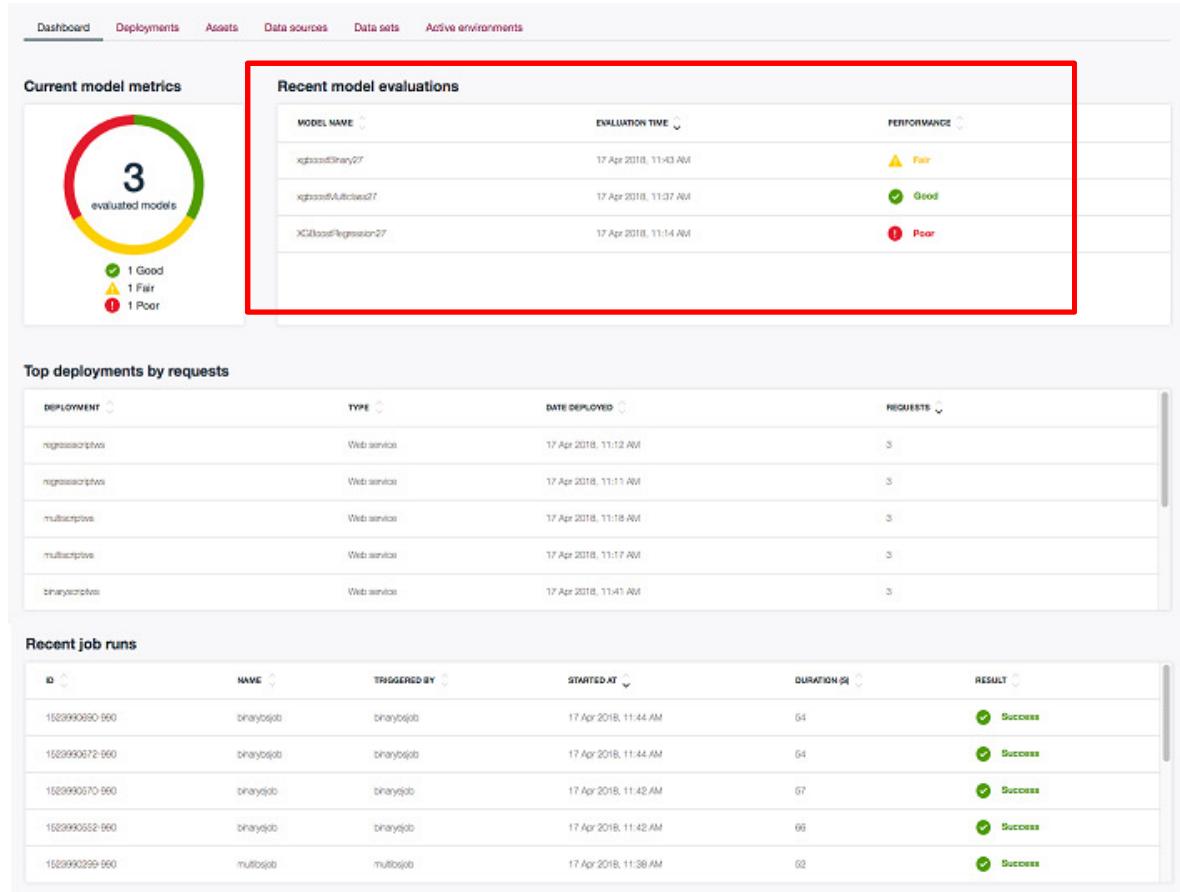
# Model Management & Deployment – Current Model Metrics

- The performance of all models across the release. The model's performance is calculated on model's score and thresholds. There are three different types of performance:
  - Good
    - Green
  - Fair
    - Yellow
  - Poor
    - Red



# Model Management & Deployment – Recent Model Evaluations

- This component shows five latest model's evaluations



The screenshot displays the IBM Model Management & Deployment interface. At the top, a navigation bar includes links for Dashboard, Deployments, Assets, Data sources, Data sets, and Active environments. The main content area is divided into several sections:

- Current model metrics:** A circular gauge indicating "3 evaluated models". Below it, a legend shows 1 Good (green), 1 Fair (yellow), and 1 Poor (red).
- Recent model evaluations:** A table listing three evaluations. The columns are MODEL NAME, EVALUATION TIME, and PERFORMANCE. The data is as follows:

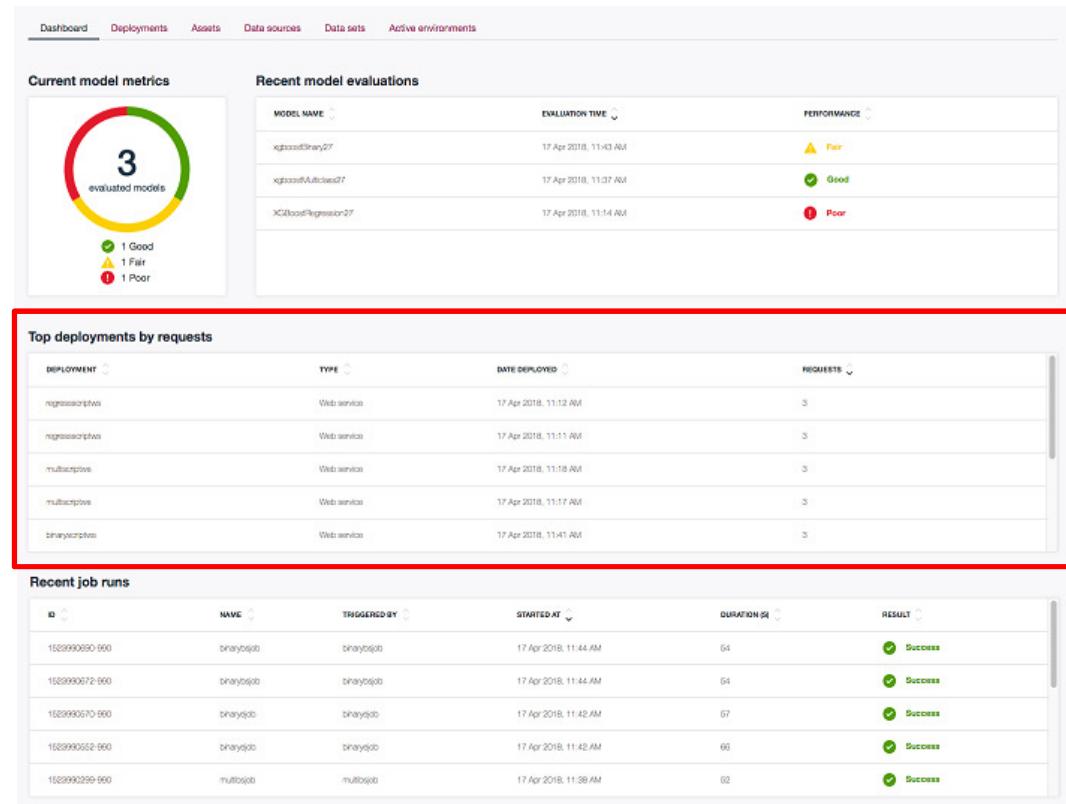
MODEL NAME	EVALUATION TIME	PERFORMANCE
xgbboostBinary2T	17 Apr 2018, 11:43 AM	Fair
xgbboostAutoclass2T	17 Apr 2018, 11:37 AM	Good
XCBoostRegression2T	17 Apr 2018, 11:14 AM	Poor
- Top deployments by requests:** A table listing five deployments. The columns are DEPLOYMENT, TYPE, DATE DEPLOYED, and REQUESTS. All entries show 3 requests.

DEPLOYMENT	TYPE	DATE DEPLOYED	REQUESTS
regressionscripts	Web service	17 Apr 2018, 11:12 AM	3
regressionscripts	Web service	17 Apr 2018, 11:11 AM	3
multiscripts	Web service	17 Apr 2018, 11:18 AM	3
multiscripts	Web service	17 Apr 2018, 11:17 AM	3
binariescripts	Web service	17 Apr 2018, 11:11 AM	3
- Recent job runs:** A table listing six job runs. The columns are ID, NAME, TRIGGERED BY, STARTED AT, DURATION [S], and RESULT. All runs are marked as Success.

ID	NAME	TRIGGERED BY	STARTED AT	DURATION [S]	RESULT
1529990990-990	binayjob	binayjob	17 Apr 2018, 11:44 AM	64	Success
1529990972-990	binayjob	binayjob	17 Apr 2018, 11:44 AM	64	Success
1529990970-990	binayjob	binayjob	17 Apr 2018, 11:42 AM	67	Success
1529990552-990	binayjob	binayjob	17 Apr 2018, 11:42 AM	66	Success
1529990299-990	multibjob	multibjob	17 Apr 2018, 11:39 AM	62	Success

# Model Management & Deployment – Top Deployments by Requests

- This component shows five deployments that have the most number of requests
- There are three different types of deployments:
  - Job
  - Web Service
  - App



The screenshot displays the IBM Model Management & Deployment interface. At the top, there are tabs for Dashboard, Deployments, Assets, Data sources, Data sets, and Active environments. The Deployments tab is selected.

**Current model metrics:** A circular gauge showing 3 evaluated models, with a legend indicating 1 Good (green), 1 Fair (yellow), and 1 Poor (red).

**Recent model evaluations:** A table listing three models with their evaluation times and performance status (Fair, Good, Poor).

MODEL NAME	EVALUATION TIME	PERFORMANCE
kgboostBinary27	17 Apr 2018, 11:43 AM	Fair
kgboostDecision27	17 Apr 2018, 11:07 AM	Good
XCBoostRegressor-27	17 Apr 2018, 11:14 AM	Poor

**Top deployments by requests:** A table listing five deployments categorized as Web services, each with a deployment date and request count of 3.

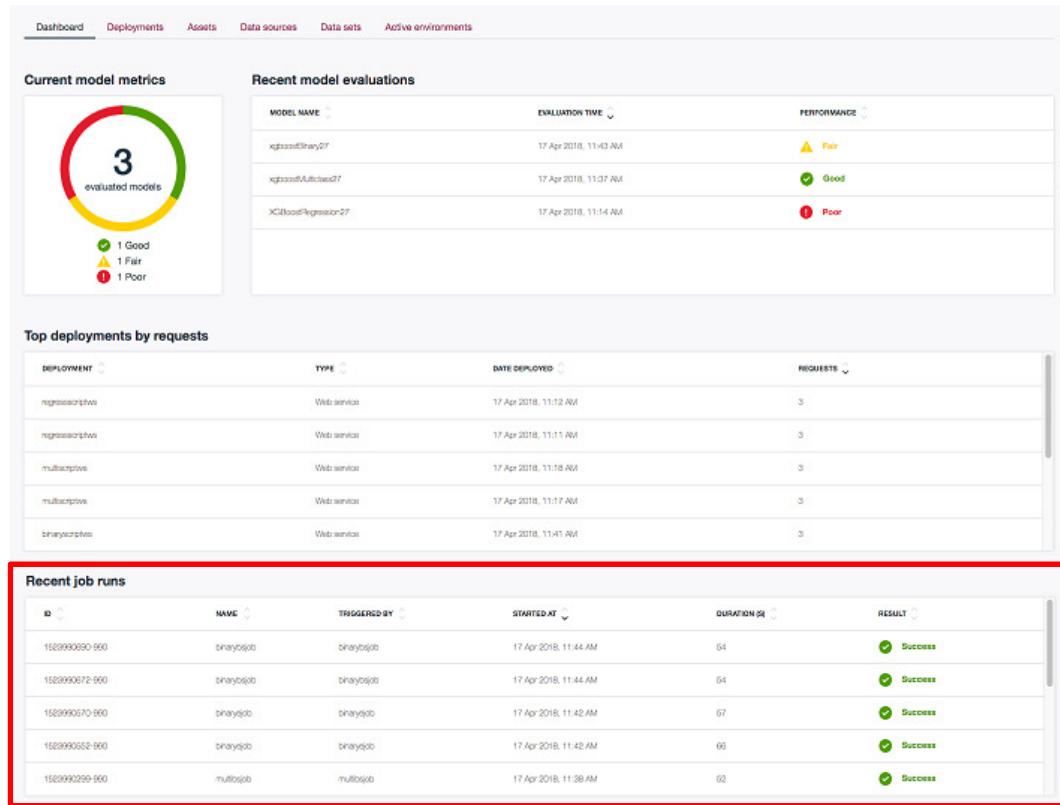
DEPLOYMENT	TYPE	DATE DEPLOYED	REQUESTS
regressionservice	Web service	17 Apr 2018, 11:12 AM	3
regressionservice	Web service	17 Apr 2018, 11:11 AM	3
multiclassservice	Web service	17 Apr 2018, 11:18 AM	3
multiclassservice	Web service	17 Apr 2018, 11:17 AM	3
binaryservice	Web service	17 Apr 2018, 11:41 AM	3

**Recent job runs:** A table listing five job runs with their names, triggered by, start time, duration, and result (all successful).

ID	NAME	TRIGGERED BY	STARTED AT	DURATION (S)	RESULT
15289900690-960	binayjob0	binayjob0	17 Apr 2018, 11:44 AM	64	Success
15289900672-960	binayjob0	binayjob0	17 Apr 2018, 11:44 AM	64	Success
15289900670-960	binayjob0	binayjob0	17 Apr 2018, 11:42 AM	67	Success
15289900562-960	binayjob0	binayjob0	17 Apr 2018, 11:42 AM	66	Success
15289902299-960	multosjob0	multosjob0	17 Apr 2018, 11:38 AM	62	Success

# Model Management & Deployment – Recent Job Runs

- This component shows five most recent job runs



The screenshot displays a dashboard for Model Management & Deployment. At the top, there are tabs for Dashboard, Deployments, Assets, Data sources, Data sets, and Active environments. The Dashboard tab is selected.

**Current model metrics:** A circular gauge showing 3 evaluated models. Below it, a legend indicates 1 Good (green), 1 Fair (yellow), and 1 Poor (red).

**Recent model evaluations:** A table listing three models with their evaluation times and performance scores (Fair, Good, Poor).

MODEL NAME	EVALUATION TIME	PERFORMANCE
kgboostBinary27	17 Apr 2018, 11:43 AM	Fair
kgboostMulticat27	17 Apr 2018, 11:07 AM	Good
XGBBoostRegression-27	17 Apr 2018, 11:14 AM	Poor

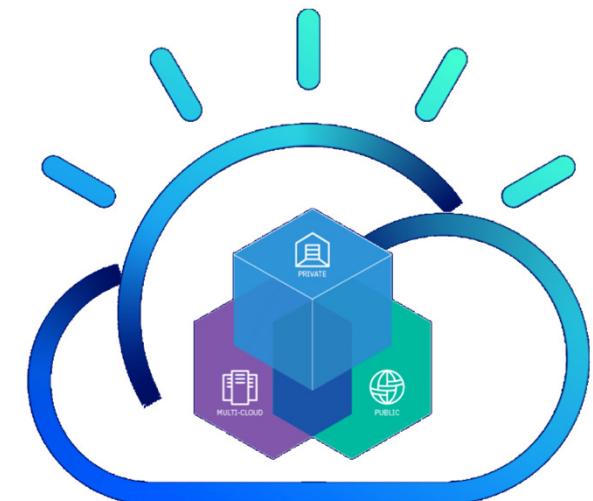
**Top deployments by requests:** A table listing five deployments based on request volume.

DEPLOYMENT	TYPE	DATE DEPLOYED	REQUESTS
regressionscriptsweb	Web service	17 Apr 2018, 11:12 AM	3
regressionscriptsweb	Web service	17 Apr 2018, 11:11 AM	3
multicatscriptsweb	Web service	17 Apr 2018, 11:18 AM	3
multicatscriptsweb	Web service	17 Apr 2018, 11:17 AM	3
binaryscriptsweb	Web service	17 Apr 2018, 11:41 AM	3

**Recent job runs:** A table listing five recent job runs, all of which were successful.

ID	NAME	TRIGGERED BY	STARTED AT	DURATION (S)	RESULT
15289900690-960	binayjob0	binayjob0	17 Apr 2018, 11:44 AM	64	Success
15289900672-960	binayjob0	binayjob0	17 Apr 2018, 11:44 AM	64	Success
15289900670-960	binayjob0	binayjob0	17 Apr 2018, 11:42 AM	67	Success
15289900562-960	binayjob0	binayjob0	17 Apr 2018, 11:42 AM	66	Success
15289902699-960	multosjob0	multosjob0	17 Apr 2018, 11:38 AM	62	Success

# Watson Studio Local Training Kubernetes 101



# Kubernetes 101

- What is Kubernetes?
  - Kubernetes is an open source orchestration tool developed by Google for managing microservices and containerized applications across a distributed cluster.
  - Kubernetes provides highly resilient infrastructure with zero downtime deployment capabilities, automatic rollback, scaling, and self-healing of containers.
  - The main objective of Kubernetes is to hide the complexity of managing a fleet of containers by providing REST APIs for the required functionalities.
  - Kubernetes is portable in nature, meaning it can run on various public or private cloud platforms such as AWS, Azure, OpenStack, Google Cloud, or Apache Mesos. It can also run on bare metal machines.

**FYI.** Kubernetes is a part of Cloud Native Computing Foundation (CNCF) project hosted by The Linux Foundation. <https://www.cncf.io>

# Kubernetes 101

- What is K8s?
  - Developers are lazy and somewhere in the mid-late 80s they started abbreviating the words based on their first letter, last letter, and number of letters in between.
  - For example **i18n** is equivalent to **internationalization** and **I10n** for **localization**.
  - So K8s is Kubernetes. ☺

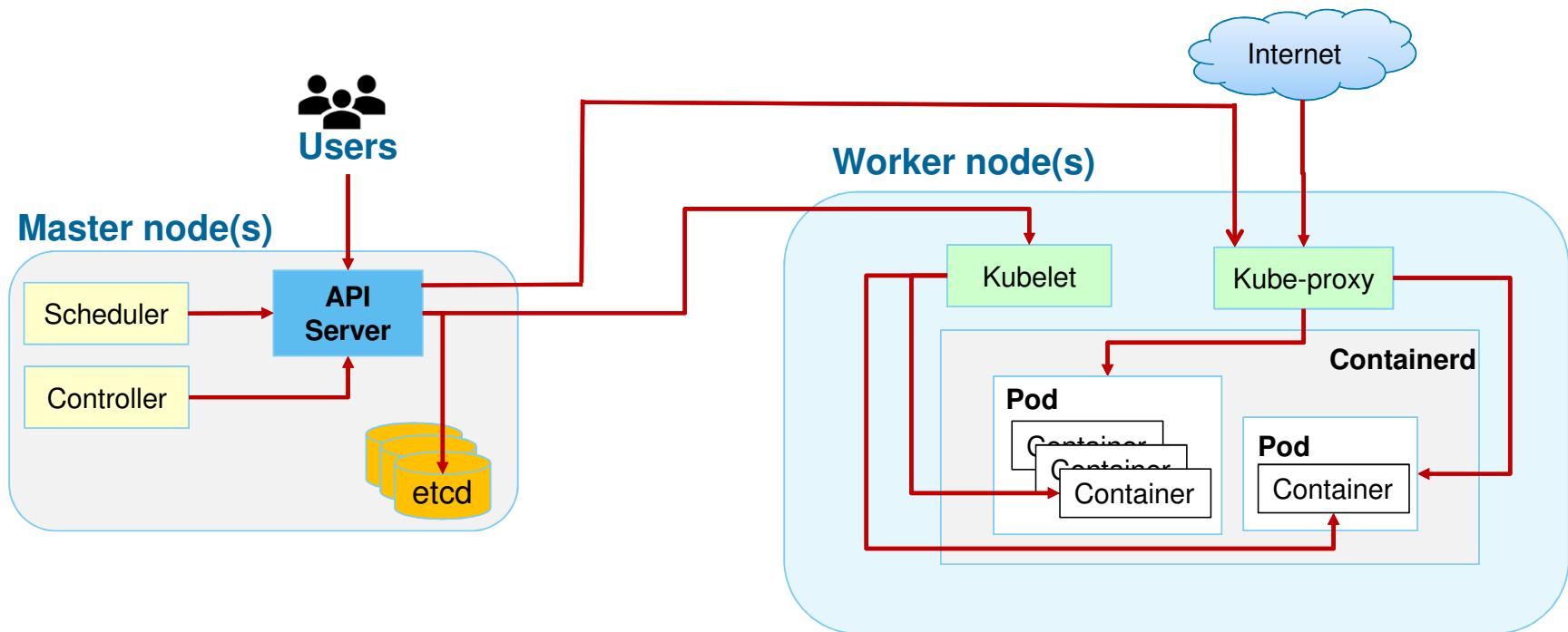
FYI. Kubernetes is an open source project written in Go language, and licensed under the Apache License Version 2.0. The current stable version is 1.9 (as of February 2018)

# Kubernetes 101

- Most popular container orchestrators nowadays
  - Docker Swarm – provided by Docker, Inc. [www.docker.com](http://www.docker.com)
  - Kubernetes – Cloud Native Computing Foundation. <https://www.cncf.io/>
  - Mesos Marathon – Apache Mesos. <http://mesos.apache.org/>
  - Amazon ECS – Amazon EC2 Container Service <https://aws.amazon.com/ecs/>
  - Hashicorp Nomad – provided by HashiCorp. <https://www.hashicorp.com/>

# Kubernetes 101

- Kubernetes Components and Architecture
  - Kubernetes follows a client-server architecture. You can have multi-master setup (for HA).



# Kubernetes 101

- Kubernetes Components and Architecture

- **Master node**

- The master node is responsible for the management of Kubernetes cluster.
    - The master node is the one taking care of orchestrating the worker nodes, where the actual services are running
    - We can communicate to the master node via the CLI, the GUI (Dashboard), or via APIs.
    - Only one of the master nodes would be the leader, performing all the operations. The others are followers.

# Kubernetes 101

- Kubernetes Components and Architecture

- **Worker node**

A worker node is a machine which runs the applications using Pods and is controlled by the master node. Pods are scheduled on the worker nodes, which have the necessary tools. It's a logical collection of one or more containers which are always scheduled together.

- Docker

- Docker runtime (ie. Containerd) runs on each of the worker nodes, and runs the configured pods. It handles downloading the images and starting the containers

- Kubelet

- Kubelet is an agent which runs on each worker node and communicates with the master node. It gets the configuration of a pod from the API server and ensures that the described containers are up and running.

# Kubernetes 101

- Kubernetes Components and Architecture

- Kube-proxy

- Kube-proxy acts as a network proxy and a load balancer for a service on a single worker node. It takes care of the network routing for TCP and UDP packets.
    - It listens to the API server for each Service endpoint creation/deletion. It setup the routes so that it can reach to it.

- Kubectl

- A command line tool to communicate with the API service and send commands to the master node.

# Kubernetes 101

- Kubernetes Components and Architecture

- **API server**

- The API server is the entry point for all the REST commands used to control the cluster. It processes the REST requests, validates them, and executes the bound business logic.

- **Scheduler**

- The scheduler schedules the work to worker nodes.
    - The scheduler has the information regarding resources available on the members of the cluster, as well as the ones required for the configured service to run and hence is able to decide where to deploy a specific service.

# Kubernetes 101

- Kubernetes Components and Architecture
  - **Controller-manager**

- A controller uses API server to watch the shared state of the cluster and make corrective changes to the current state to change it to the desired one.
- You may have multiple different kind of controllers, for example, the replication controller which takes care of number of pods in the system. Other examples like, endpoints controller, namespace controller, and service accounts controller, etc.

# Kubernetes 101

- Kubernetes Components and Architecture

- **Pod**

- Kubernetes targets the management of elastic applications that consist of multiple microservices communicating with each other. Often those microservices are tightly coupled forming a group of containers that would typically, in a non-containerized setup run together on one server. This group, the smallest unit that can be scheduled to be deployed through K8s is called a pod.
    - This group of containers share storage, Linux namespaces, IP addresses. Containers reach each other via localhost.
    - Pods are not intended to live long. They are created, destroyed and re-created on demand, based on the state of the server and the service.

# Kubernetes 101

- Kubernetes Components and Architecture

- **Service**

- As pods have a short lifetime, there is not guarantee about the IP address they are served on. This could make the communication of microservices difficult.
    - K8s has introduced the concept of a service, which is an abstraction on top of a number of pods, communicating with it via a Virtual IP address.
    - This is where you can configure load balancing for your numerous pods and expose them via a service.

# Kubernetes 101

- Kubernetes Components and Architecture

- **etcd storage**

- etcd is a simple, distributed, consistent key-value store. It's mainly used for shared configuration and service discovery.
    - It provides a REST API for CRUD operations as well as an interface to register watchers on specific nodes, which enables a reliable way to notify the rest of the cluster about configuration changes.
    - An example of data stored by Kubernetes in etcd is jobs being scheduled, created and deployed, pod/service details and state, namespaces and replication information, etc.

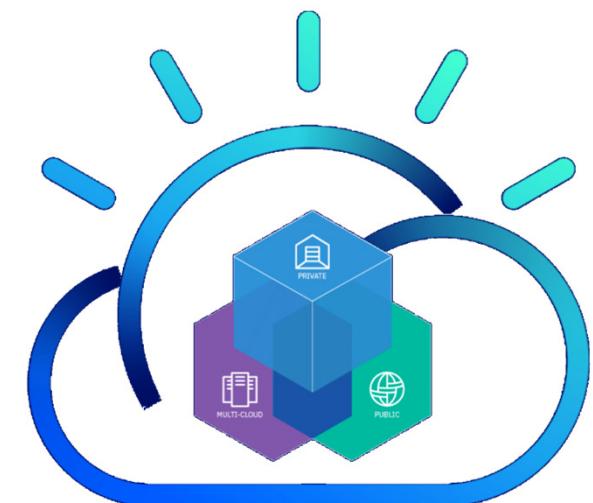
# Kubernetes 101

- Kubernetes Components and Architecture

- **GlusterFS**

- Gluster is a scalable, distributed file system that aggregates disk storage resources from multiple servers into a single global namespace.
    - Advantages
      - Scales to several petabytes
      - Handles thousands of clients
      - POSIX compatible
      - Uses commodity hardware
      - Can use any ondisk filesystem that supports extended attributes
      - Accessible using industry standard protocols like NFS and SMB
      - Provides replication, quotas, geo-replication, snapshots and bitrot detection
      - Allows optimization for different workloads
      - Open source

# Watson Studio Local Administration Security



# IBM Watson Studio Local (WSL) – Security

## Network security

- WSL is exposed via 443 port ( SSL is supported by default )
- All REST APIs require bearer token or model deployment Token
- WSL supports firewall (vyatta) around cluster, not around individual nodes
- SSO is not yet supported

## Data security

- WSL supports full disk encryption via LUKS (must be done prior to install)
- WSL encrypts the credentials for external data sources (AES 256)
- WSL hashes all the user passwords stored in Cloudant

## Authentication

- LDAP/AD(See Compatibility Reports for supported LDAP servers)

## Authorization

- Pre-defined roles: Administrator, Deployment Administrator, User
- Project roles: Administrator, Editor, Viewer

## IBM Watson Studio Local (WSL) – Security

Watson Studio Local comes equipped with an array of security features you can use. Including:

- SAML
  - An administrator can setup SAML2.0 to redirect WSL users to sign in securely through your identity provider's login page
- SSH
  - You can install WSL using a private SSH key file for the credentials
  - A WSL administrator can enable SSHD so that users (using a public SSH key pair) can ssh directly to the WSL cluster through a secure port.
- SSL
- Tokens
- Authentication
- Authorization
- Data

## Audit Log

- WSL logs an audit record of user login attempts (either from the sign in page or using the validateAuth REST API) to the WSL cluster
- The record is stored as JSON, and contains the following information about each login attempt:

### Template:

```
{  
  "DSX_AUDIT_RECORD": {  
    "version": "1.0",  
    "event": "user_auth",  
    "timestamp": Request timestamp in ISO8601 DateTime with  
    Timeszone format  
    "user_auth_object": {  
      "user_auth_object_version": "1.0",  
      "uri": Request URI  
      "username": The user name used to attempt the login,  
      "status": Result Status code (e.g. 200: OK, 401:  
      Authorization failure)  
    }  
  }  
}
```

### Example:

```
{  
  "DSX_AUDIT_RECORD": {  
    "version": "1.0",  
    "event": "user_auth",  
    "timestamp": "2018-03-08T06:38:00+00:00",  
    "user_auth_object": {  
      "user_auth_object_version": "1.0",  
      "uri": "\/v1\/preauth\/validateAuth",  
      "username": "admin",  
      "status": 200  
    }  
  }  
}
```

## Audit Records

- All audit records are retained for a default of 10 days before they are automatically deleted
- The WS administrator can modify the log retention settings by clicking Settings from the admin user profile icon and under Refresh and alert settings, typing in a new duration in the Log retention (days) field
- **Tip:** The admin could download the user login audit records using the REST API periodically, before they are automatically deleted for long term storage

# IBM Watson Studio Local (WSL) – Security

## Get User Bearer Tokens

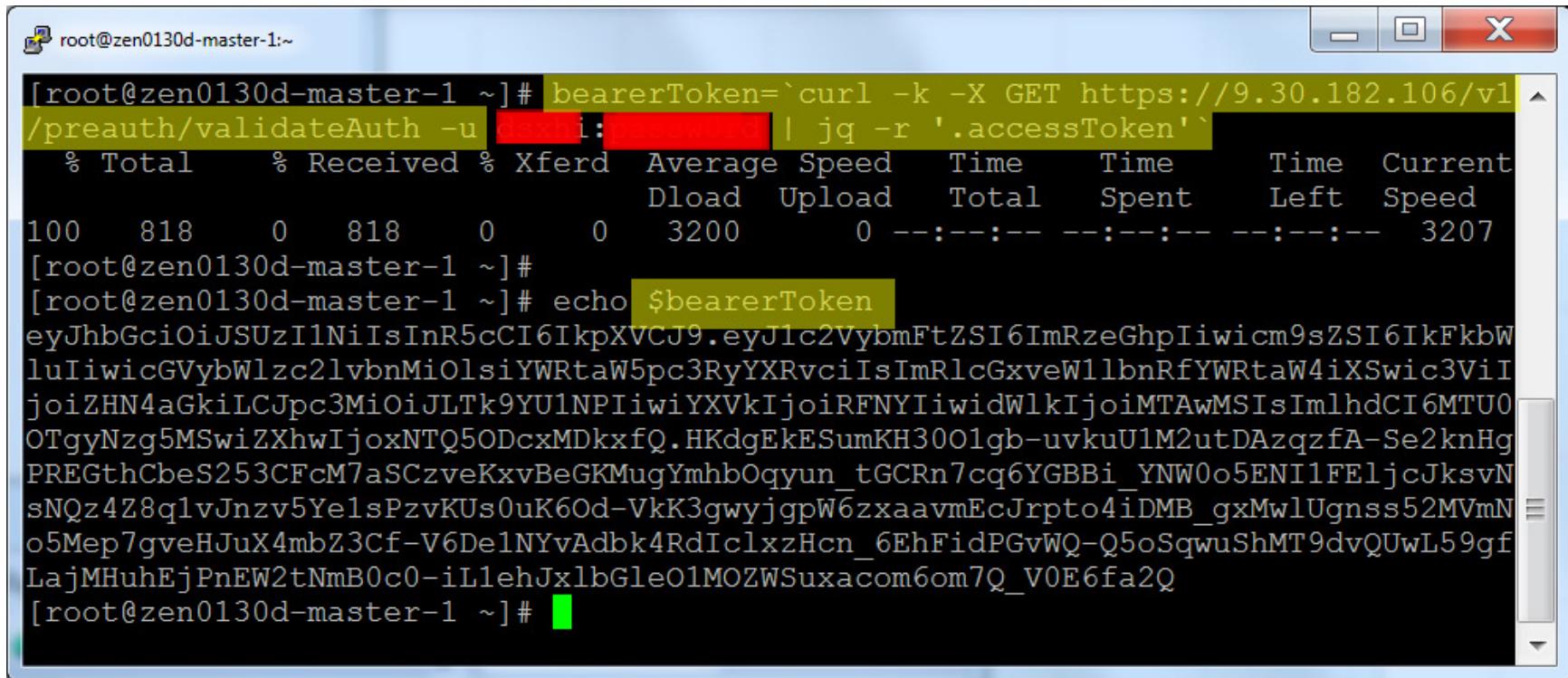
- WSL requires a bearer token to authenticate the user calling the REST APIs.
- The token lasts for 13 hours. The following example shows how to retrieve and set a bearer token from the user management service:

### Example

```
# bearerToken=`curl -k -X GET <WSL_URL>/v1/preauth/validateAuth -u <uid>:<password> | jq -r '.accessToken'
```

# IBM Watson Studio Local (WSL) – Security

## Get User Bearer Tokens



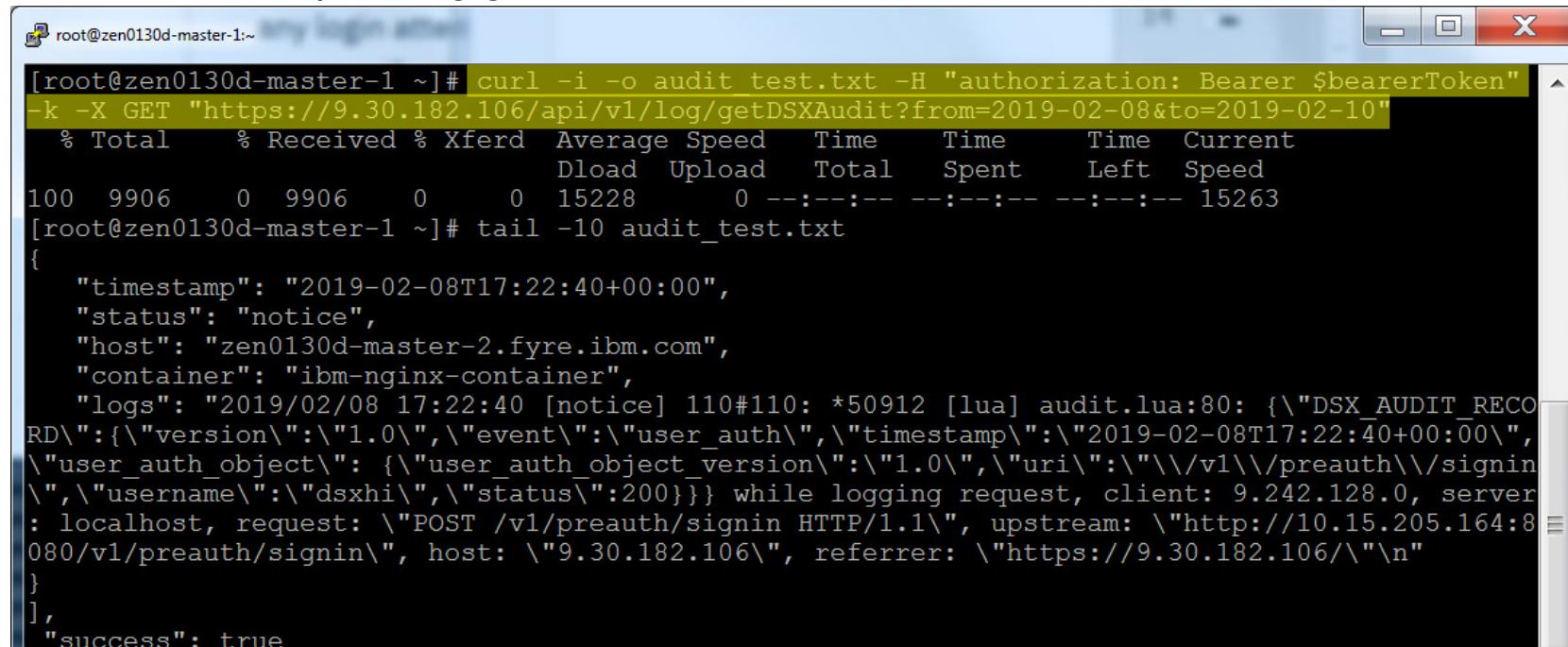
```
root@zen0130d-master-1:~]# bearerToken=`curl -k -X GET https://9.30.182.106/v1/preauth/validateAuth -u [REDACTED]:[REDACTED] | jq -r '.accessToken'`  
% Total    % Received % Xferd  Average Speed   Time     Time      Current  
          Dload  Upload   Total  Spent  Left  Speed  
100  818    0  818    0    0  3200       0 --:--:-- --:--:-- --:--:-- 3207  
[root@zen0130d-master-1 ~]#  
[root@zen0130d-master-1 ~]# echo $bearerToken  
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmcFtZSI6ImRzeGhpIiwicm9sZSI6IkFkbWluIiwicGVybWlzc2lvbnMiOlsiYWRtaW5pc3RyYXRvciiSImRlcGxveW1lbnRfYWRtaW4iXSwic3ViIjoiZHN4aGkiLCJpc3MiOiJLTk9YU1NPIiwiYXVkIjoiRFNYIiwidWlkIjoiMTAwMSIsImhdCI6MTU0OTgyNzg5MSwiZXhwIjoxNTQ5ODcxMDkxfQ.HKdgEkESumKH3001gb-uvkuU1M2utDAzqzfA-Se2knHgPREGthCbes253CFcM7aSCzveKxvBeGKMugYmhboqyuu_tGCRn7cq6YGBBi_YNW0o5ENI1FE1jcJksvNsNQz4Z8q1vJnzb5Ye1sPzvKUs0uK60d-VkK3gwyjgpW6zxaavmEcJrpto4iDMB_gxMwlUgnss52MVmNo5Mep7gveHJuX4mbZ3Cf-V6De1NYvAdbk4RdIclxzHcn_6EhFidPGvWQ-Q5oSqwuShMT9dvQUwL59gfLajMHuhEjPnEW2tNmB0c0-iL1ehJxlbgLe01MOZWSuxacom6om7Q_V0E6fa2Q  
[root@zen0130d-master-1 ~]#
```

# IBM Watson Studio Local (WSL) – Security

## Get Server Audit Logs

### Example

```
# curl -i -o audit_test.txt -H "authorization: Bearer $bearerToken" -k -X GET  
“<WSL_URL>/api/v1/log/getDSXAudit?from=2019-02-08&to=2019-02-10”
```



The screenshot shows a terminal window with the following content:

```
[root@zen0130d-master-1 ~]# curl -i -o audit_test.txt -H "authorization: Bearer $bearerToken" -k -X GET "https://9.30.182.106/api/v1/log/getDSXAudit?from=2019-02-08&to=2019-02-10"  
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current  
          Dload  Upload   Total Spent  Left Speed  
100  9906     0  9906     0      0  15228       0 --:--:-- --:--:-- --:--:-- 15263  
[root@zen0130d-master-1 ~]# tail -10 audit_test.txt  
{  
  "timestamp": "2019-02-08T17:22:40+00:00",  
  "status": "notice",  
  "host": "zen0130d-master-2.fyre.ibm.com",  
  "container": "ibm-nginx-container",  
  "logs": "2019/02/08 17:22:40 [notice] 110#110: *50912 [lua] audit.lua:80: {\"DSX_AUDIT_RECO  
RD\":{\"version\":\"1.0\",\"event\":\"user_auth\",\"timestamp\":\"2019-02-08T17:22:40+00:00\",  
\"user_auth_object\": {\"user_auth_object_version\":\"1.0\",\"uri\":\"\\v1\\preauth\\signin\",  
\"username\":\"dsxhi\",\"status\":200}}} while logging request, client: 9.242.128.0, server:  
localhost, request: \"POST /v1/preauth/signin HTTP/1.1\", upstream: \"http://10.15.205.164:8  
080/v1/preauth/signin\", host: \"9.30.182.106\", referrer: \"https://9.30.182.106/\\n\"  
}  
],  
  "success": true
```

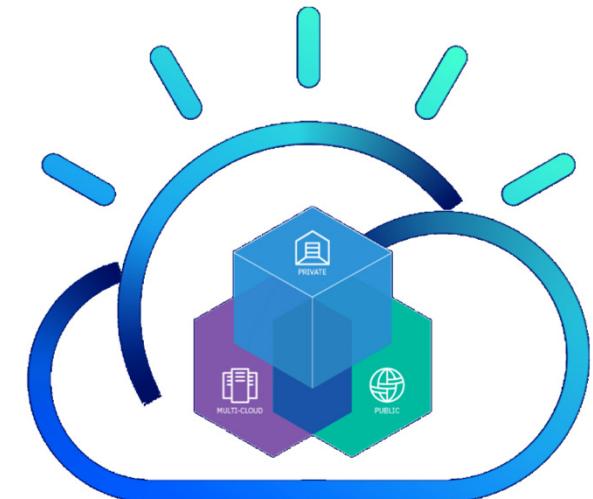
# IBM Watson Studio Local (WSL) – Security

## Usage Report

- Every day at 1AM (machine time) WSL creates and stores a report of users signing into the system.
- To retrieve these usage report files, run the script “/wdp/utils/get\_usage\_reports.sh” on any one of the master nodes.
- The script saves zipped usage reports in the ws\_usage\_reports directory.
- Each usage report is a comma delimited CSV file with the column: Username, Login timestamp.
- Example report:

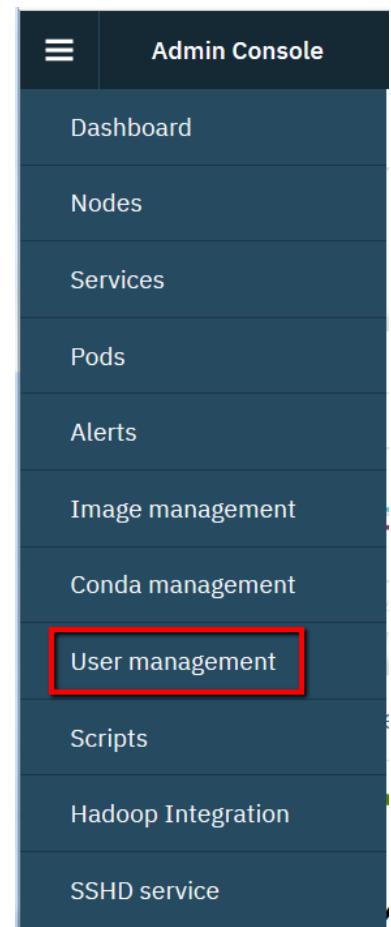
```
test_user_1_36, 2018-12-12T18:04:23+00:00
test_user_1_29, 2018-12-12T18:04:23+00:00
test_user_1_62, 2018-12-12T18:04:22+00:00
test_user_1_2, 2018-12-12T18:04:22+00:00
test_user_1_83, 2018-12-12T18:04:21+00:00
test_user_1_3, 2018-12-12T18:04:21+00:00
test_user_1_46, 2018-12-12T18:04:20+00:00
```

# Watson Studio Local Cluster Management Users



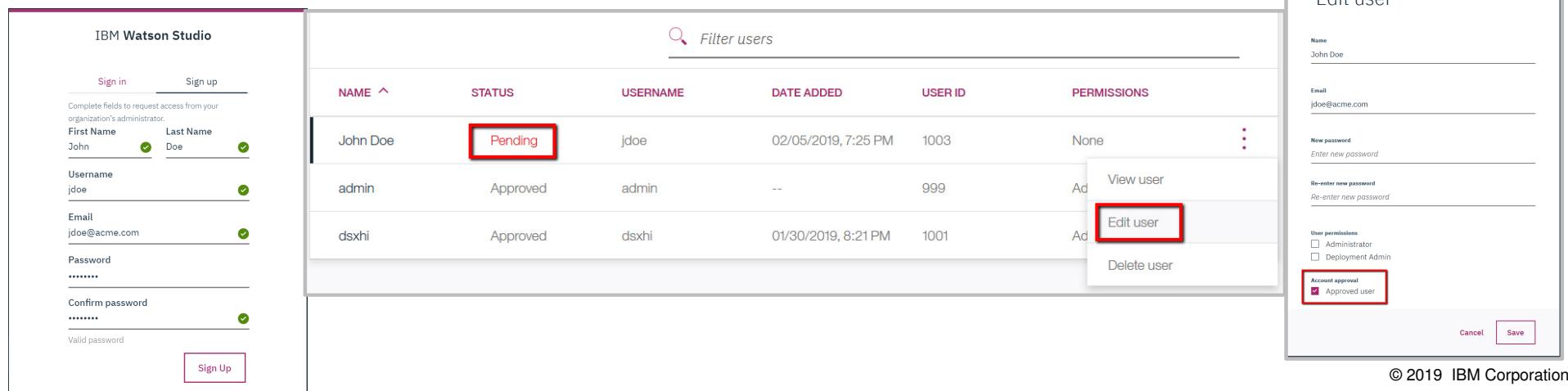
## Manage Users

- WSL users can be managed from either an external LDAP server (recommended) or an internal repository database
- In the Admin Console, click the menu icon (≡) and click User Management to approve sign-up requests, add users, filter them, edit them, assign permissions to them, or delete them



## Manage Users

- The user permissions are as follows:
  - Admin:** Can sign in to both the admin dashboard and the WSL client
  - Deployment Admin:** Can create project releases in IBM Watson Machine Learning
  - User:** Can sign in to the WSL client only. This is the default permission if neither Admin check box is selected
- If new users request an account, you must approve them by editing them and selecting Approved User
- Unapproved users have a status of pending

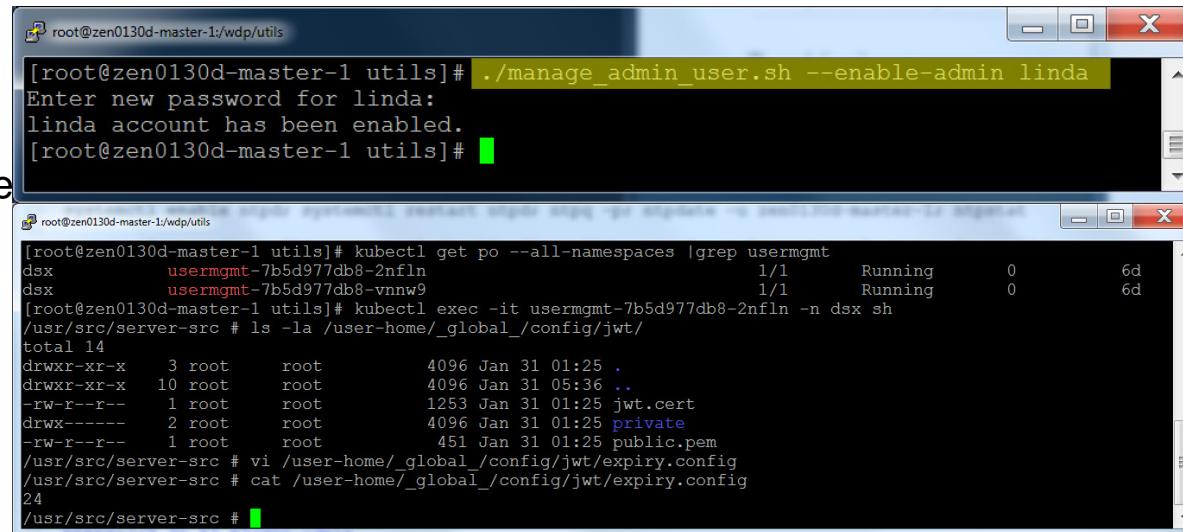


The screenshot shows the IBM Watson Studio User Management interface. On the left, there is a sign-up form for a new user named 'John Doe' with fields for First Name, Last Name, Username, Email, Password, and Confirm password. The 'Status' field is set to 'Pending'. In the center, a table lists existing users: 'John Doe' (Status: Pending), 'admin' (Status: Approved), and 'dsxhi' (Status: Approved). A context menu is open over the 'John Doe' row, with options 'View user', 'Edit user' (which is highlighted with a red box), and 'Delete user'. On the right, a detailed 'Edit user' dialog is displayed for 'John Doe'. It includes fields for Name (John Doe), Email (jdoe@acme.com), New password, Re-enter new password, User permissions (Administrator and Deployment Admin checkboxes), and Account approval (Approved user checkbox, which is checked and highlighted with a red box). The dialog also has 'Cancel' and 'Save' buttons.

NAME	STATUS	USERNAME	DATE ADDED	USER ID	PERMISSIONS
John Doe	Pending	jdoe	02/05/2019, 7:25 PM	1003	None
admin	Approved	admin	--	999	Administrator
dsxhi	Approved	dsxhi	01/30/2019, 8:21PM	1001	Administrator

## Manage Users & Others

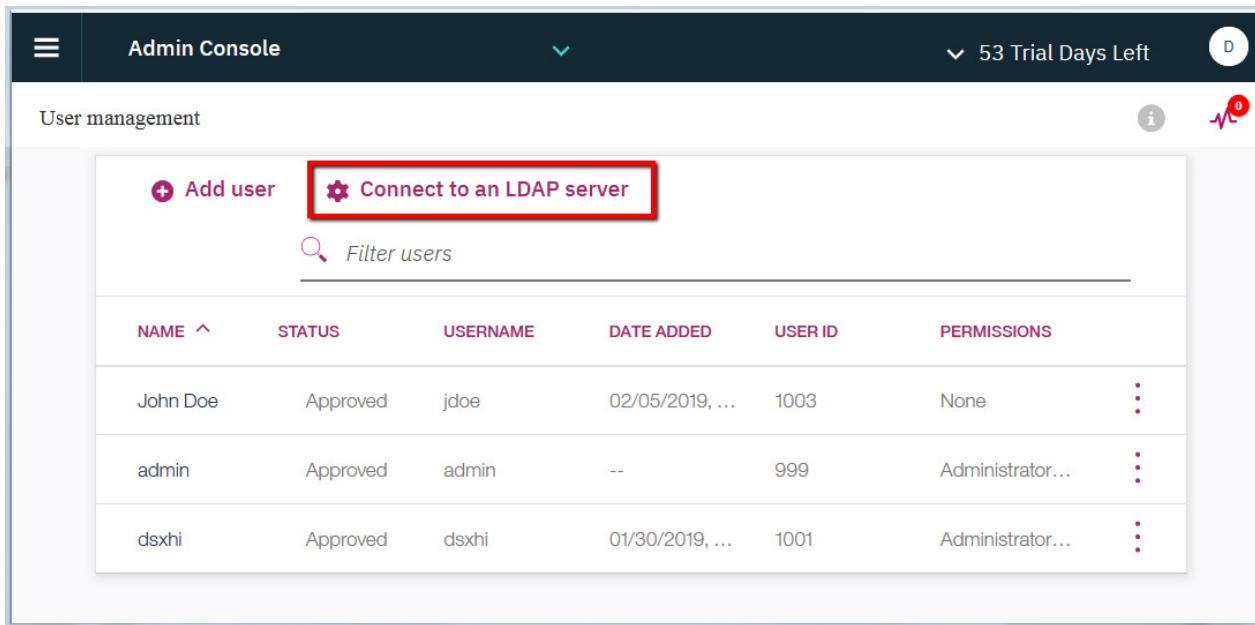
- Enable and set a new password for the WSL admin, enter the following command:
  - ./wdp/utils/manage\_admin\_user.sh --enable-admin <userid>
- Disable a WSL admin, enter the following command:
  - ./wdp/utils/manage\_admin\_user.sh --disable-admin <userid>
- Modify the session expiration time
  1. SSH to WSL cluster node
  2. Run kubectl exec to a usermgmt pod.
  3. Edit or create the following file to specify the number of hours. (default is 12 hours)  
- \$cat /user-home/\_global\_/config/jwt/expiry.config
  4. Restart usermgmt pod by deleting all of the current usermgmt pods.



```
root@zen0130d-master-1:/wdp/utils# ./manage_admin_user.sh --enable-admin linda
Enter new password for linda:
linda account has been enabled.
[root@zen0130d-master-1:/wdp/utils# ./manage_admin_user.sh --enable-admin linda
Enter new password for linda:
linda account has been enabled.
[root@zen0130d-master-1:/wdp/utils# kubectl get po --all-namespaces |grep usermgmt
dsx      usermgmt-7b5d977db8-2nfln          1/1     Running   0      6d
dsx      usermgmt-7b5d977db8-vnnw9          1/1     Running   0      6d
[root@zen0130d-master-1:/wdp/utils# kubectl exec -it usermgmt-7b5d977db8-2nfln -n dsx sh
/usr/src/server-src # ls -la /user-home/_global_/config/jwt/
total 14
drwxr-xr-x  3 root    root        4096 Jan 31 01:25 .
drwxr-xr-x  10 root   root        4096 Jan 31 05:36 ..
-rw-r--r--  1 root   root       1253 Jan 31 01:25 jwt.cert
drwx-----  2 root   root        4096 Jan 31 01:25 private
-rw-r--r--  1 root   root       451 Jan 31 01:25 public.pem
/usr/src/server-src # vi /user-home/_global_/config/jwt/expiry.config
/usr/src/server-src # cat /user-home/_global_/config/jwt/expiry.config
24
/usr/src/server-src #
```

## Manage Users – LDAP Server

- By default, WSL user records are stored in its internal repository database
- Alternatively, you can use your own external LDAP server instead. To set up your own LDAP server, click Connect to an LDAP server



The screenshot shows the Admin Console User management interface. At the top, there are buttons for 'Add user' and 'Connect to an LDAP server'. The 'Connect to an LDAP server' button is highlighted with a red box. Below these buttons is a search bar labeled 'Filter users'. The main area displays a table of user records:

NAME ^	STATUS	USERNAME	DATE ADDED	USER ID	PERMISSIONS
John Doe	Approved	jdoe	02/05/2019, ...	1003	None
admin	Approved	admin	--	999	Administrator...
dsxhi	Approved	dsxhi	01/30/2019, ...	1001	Administrator...

## Manage Users – LDAP Server

- In the LDAP host field, use the *ldap://* prefix for a non-secure port and the *ldaps://* prefix for a secure port
- If you opt to authenticate LDAP with search, then specify the domain search user, password, and base. If you opt to authenticate by distinguished name without search, ensure the **LDAP Prefix** and **LDAP Suffix** fields match the distinguished name exactly. For example, uid for the prefix and ou = dsxusers, dc = ibm, dc = com for suffix for the setup to succeed
- When finished, click the **Set up LDAP** button. If the LDAP setup succeeds, WSL no longer displays password fields whenever you sign up a new user in the Admin Console.

## Manage Users – LDAP Server

- Because the WSL user records are stored in the external LDAP server, only the LDAP administrator can perform user management tasks like password resets and changes
- Note that after LDAP is enabled, both local and LDAP users can sign in, but only LDAP users can be added.
- **Tip:** To save time from approving LDAP users, you can select **Auto-Signup** to automatically approve all LDAP user sign-up requests.

## Manage Users – LDAP Server

- Example “with search” configuration
- Notes:
  - Domain search user and password are optional for testing connectivity
  - Username and Password for testing must be valid LDAP user

### Connect LDAP

- with search
- without search

**LDAP host**

ldaps://bluepages.ibm.com

**LDAP port**

636

**Domain search user***The LDAP user that performs user lookups***Domain search password***The password for the search user***Domain base**

ou=bluepages,o=ibm.com

**User Search Field**

emailAddress

 Auto-Signup (Automatically approve all sign-up requests from users)**Username for testing**

&lt;valid LDAP user&gt;@ibm.com

**Password for testing**

\*\*\*\*\*

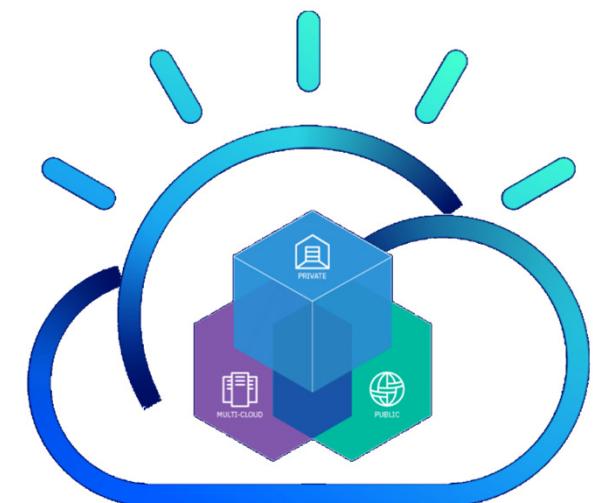
[Cancel](#)[Test LDAP](#)[Set up LDAP](#)

IBM Corporation

## Manage Users – LDAP Server

- To verify that your LDAP connection works, type in an existing LDAP user in the Username for testing and Password for testing fields, then click the Test LDAP button
- When finished, click the Set up LDAP button
- If the LDAP setup succeeds, WSL no longer displays password fields whenever you sign up a new user in the admin dashboard
- Because the WSL user credentials are stored in the external LDAP server, only the LDAP administrator can perform LDAP user password resets and changes

# Watson Studio Local Administration Manage Nodes



## Monitor Cluster Nodes

- Your IBM Watson Studio Local deployment consists of up to three types of nodes
  - *Control/Storage/Compute Nodes (required)*
  - *Additional Compute Nodes (optional)*
  - *Additional Deployment Nodes (optional)*
- The cluster requires a minimum of three machines—set up as Control/Storage/Compute “master” nodes in the minimum three machine configuration—for ensuring platform high availability
- Storage is always part of the master nodes. In current versions of WSL, there are no independent storage nodes
- Additional node machines can be added to provide greater compute and deployment capability

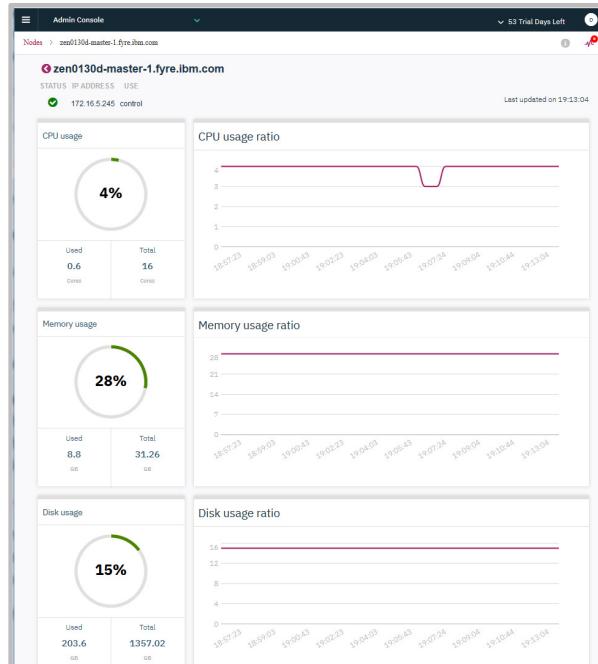
## Manage Cluster Nodes

- You can check whether cluster nodes are running from the Nodes page
- You can access the Nodes page from the menu icon (
- For each node in the cluster, you can see:
  - The IP address of the node
  - Whether the node is running or down
  - How many CPU cores the node is using
  - The GB of memory the node is using
  - The GB of storage (disk) the node is using

Control plane nodes						Last updated on 00:57:26
NAME	IP ADDRESS	STATUS	CPU USED (CORES)	MEMORY USED (GB)	DISK USED (GB)	
> 3nodeautomated-master-1. 			0.5 / 16 (3%)	10.45 / 31.26 (33%)	42.73 / 245.87 (17%)	
> 3nodeautomated-master-2. 			0.7 / 16 (4%)	8.92 / 31.26 (29%)	42.56 / 245.87 (17%)	
> 3nodeautomated-master-3. 			1.2 / 16 (8%)	9.77 / 31.26 (31%)	52.98 / 245.87 (22%)	

## Manage Cluster Nodes

- What if you need more detailed information?
- If you click the node name, you can see the CPU, storage, and memory consumption for the node over the last 20 minutes
  - For example, if we would have clicked on the a specific node from the previous screen this is what we would have seen



## Manage Cluster Nodes

- Lets “zoom out” and go back to our high level view where you can see all the nodes
- If you expand the node name (ie clicking the arrow to the left of the node), you can see the operating system partition and disk partition where the node is running.
  - For example, in the below image you can see the size of the partition and how much of the partition is in use

Control plane nodes						Last updated on 19:15:04
NAME	IP ADDRESS	STATUS	CPU USED (CORES)	MEMORY USED (GB)	DISK USED (GB)	
PARTITION	USAGE		CAPACITY	RATIO		
zen0130d-master-1.fyre.ibm.com	172.16.5.245	✓	0.7 / 16 (4%)	8.77 / 31.26 (28%)	203.65 / 1357.02 (15%)	
/dev/mapper/rhel-root	2.05 GB		240.88 GB	1%		
/dev/vdal	207.69 MB		1014 MB	20%		
/dev/vdb1	152.76 GB		599.71 GB	25%		
/dev/vdc1	48.63 GB		499.75 GB	10%		
shm	0 Byte		64 MB	0%		
tmpfs	0 Byte		15.68 GB	0%		
zen0130d-master-2.fyre.ibm.com	172.16.11.13	✓	0.6 / 16 (4%)	8.36 / 31.26 (27%)	131.02 / 1356.96 (10%)	
zen0130d-master-3.fyre.ibm.com	172.16.11.14	✓	0.6 / 16 (4%)	9.12 / 31.26 (29%)	82.52 / 1356.96 (6%)	

## Manage Cluster Nodes

- Kubernetes is designed to handle node outages, so you can still use WSL even if some of your nodes are down
- However, if a node is down, you must attempt to diagnose and resolve the problem to avoid outages if other nodes fail
- Contact IBM Software Support if you cannot determine why the node failed or how to restart the node
- Also, down the road if you want to expand the usage / resources for WSL you can add additional compute or deployment nodes
  - **Restriction:** WSL does not support adding additional nodes with GPU

# Monitor Cluster Nodes

- Your WSL deployment consists of three types of nodes:

## 1. Control Nodes:

- Nodes that manage your Kubernetes cluster and your DSX Local deployment
- By default, the cluster has three control nodes. If you notice that a node is down, attempt to restore it to prevent outages.
- The cluster can continue to run if one node fails. However, if two nodes fail, your cluster fails.

## Monitor Cluster Nodes

### 2. Storage Nodes: (Combine with Control node in latest version)

- Nodes where WSL metadata and any data that you load into WSL is stored
- By default, the cluster has three storage nodes. The data on these nodes is replicated across each node, so that if a node fails, you can still access the data. WSL can continue to run if two nodes fail
- **Tip:** If you run out of space on your storage nodes, add XFS-formatted disks to each node and extend the Logical Volume Management (LVM) partition to include the disks. If possible, ensure that the disks are the same size.

## Monitor Cluster Nodes

### 3. Compute Nodes:

- Nodes where WSL services, such as Spark, run
- Unlike storage nodes, compute nodes are not replicated. When a new process starts, Kubernetes determines which node has sufficient capacity to run the process.
- WSL can continue to run when multiple compute nodes fail. However, you might notice that performance decreases when multiple nodes are down
- Additionally, if a node fails, Kubernetes attempts to bring any active processes up on another node. While Kubernetes attempts to bring up the processes, you might experience an outage
  - **If Kubernetes cannot bring the processes up on another node and the outage continues, contact IBM Software Support**

# Monitor Cluster Nodes

## 4. Deployment Nodes: (Optional\*\*)

- Deployment nodes are production versions of the compute nodes
- Asset deployment requires at least one deployment node.
- If the deployment nodes fail all the deployments will go offline and remain offline until nodes are repaired or replaced.

**Recommended  
configuration examples**

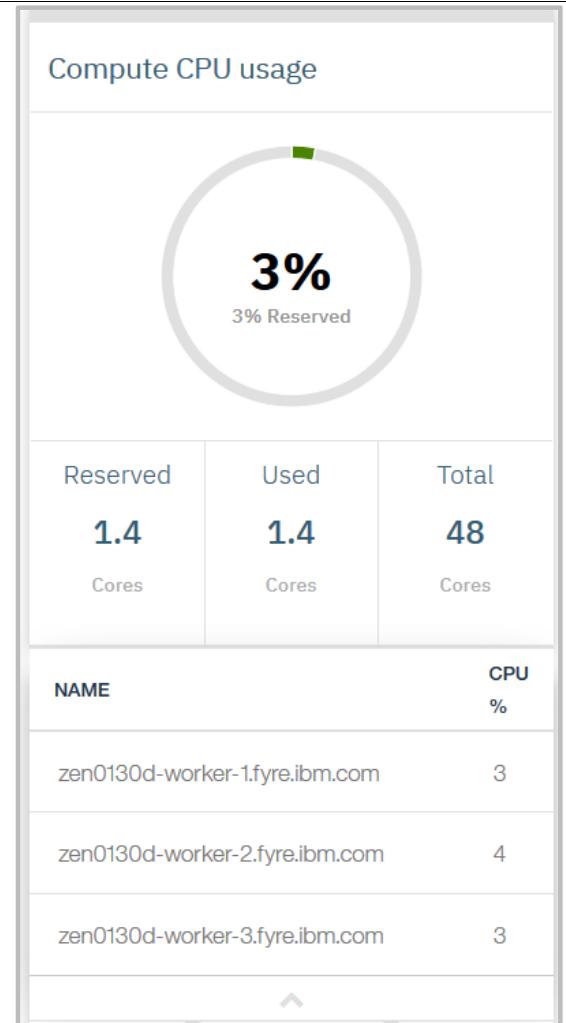
Cluster type	# node breakdown	Notes
3 nodes	3 shared control/compute	Unable to deploy assets.
4 nodes	3 shared control/compute + 1 deploy	
5 nodes	3 shared control/compute + 2 deploy	
7 nodes	3 control + 3 compute + 1 deploy	
8 nodes	3 control + 3 compute + 2 deploy	
11 nodes	3 control + 6 compute + 2 deploy	

## Node Health

- If you want a high-level overview of the status of your cluster, you can monitor the health of your cluster nodes from the Dashboard page
- You can access the Dashboard page from the menu icon (☰)
- Specifically, you can monitor:
  - CPU Usage
  - Memory Usage
  - Disk Usage

## Node Health

- For compute nodes, the usage of CPU and memory is measured against the CPU and memory that the WSL users reserved
- Each card on the Dashboard page shows the average usage across all of the nodes.
- However, you can expand the cards to see the specific usage for each node.

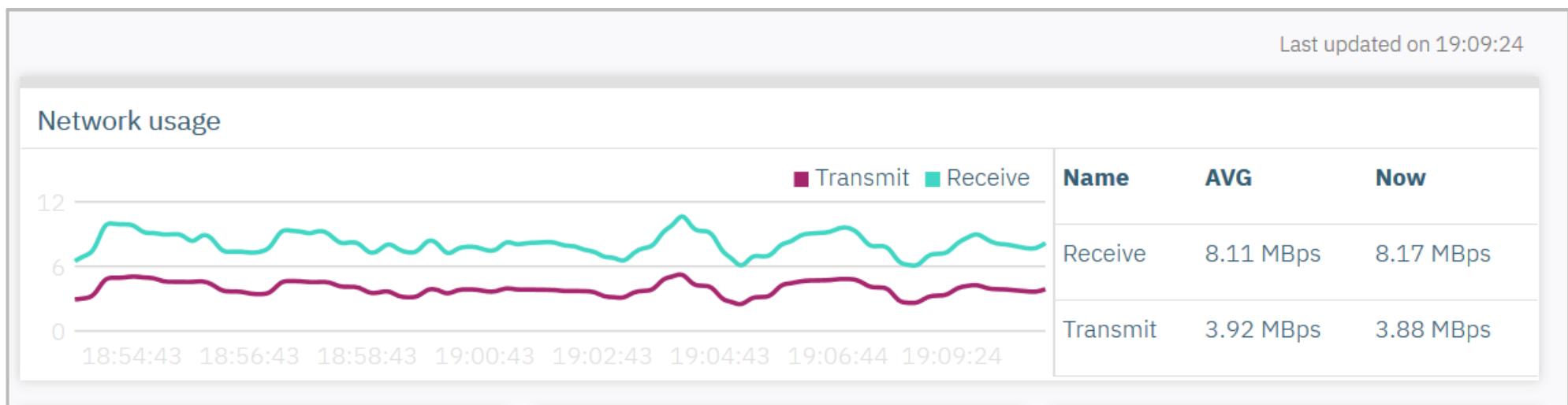


## Node Health

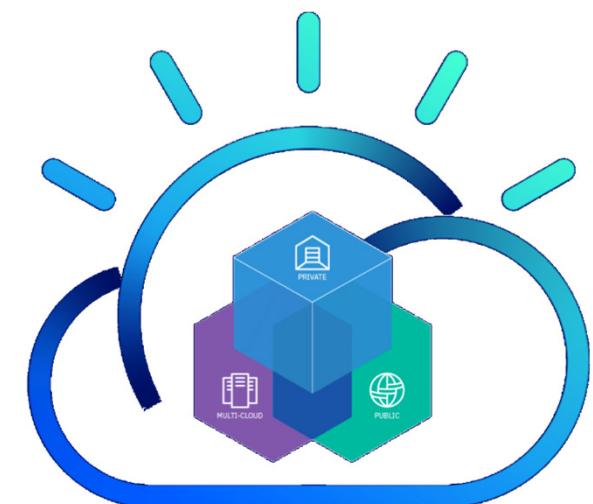
- Data is refreshed every 10 seconds
- By default, Kubernetes attempts to balance the load across servers
- Contact IBM Software Support if you notice that all of the nodes in a group are overloaded for extended time
  - Nodes are overloaded when they run above 90% usage
- Nodes can become overloaded when:
  - You have more users than your cluster configuration can handle
    - For example, your cluster doesn't have sufficient CPU, memory, or storage
  - A node fails and other nodes need to handle requests that would normally be handled by that node.

## Monitor Network Usage

- If you encounter an issue with WSL, you can view the recent network traffic in your cluster on the Dashboard page
- You can view the number of megabytes that were sent and received across the nodes of the cluster over the last 20 minutes



# Watson Studio Local Administration Manage Services



## Manage Services

- Watson Studio Local is composed of multiple processes that run in a set of distributed pods.
- Watson Studio Local uses services to enable communication between pods.
- You can view the list of services that are used in your WSL deployment from the **Services page**
- You can access the Services page from the menu icon ()

# Manage Services

- **Click the service name to see:**

- The list of pods that the service communicates with
- The status of each pod
- The nodes where each pod is deployed

Services				
NAME	DESCRIPTION	STATUS	PODS	IMAGE
Access load balancer	The load balancer for the Watson Studio client.		3	privatecloud-nginx-repo:v3.13.74-x86_64
Admin console APIs	The REST APIs for retrieving the data that is displayed in the administration console.		3	dashboard-backend:1.8.7-x86_64
Admin console UI	The user interface for the administration console.		3	dashboard-frontend:1.8.7-x86_64
Admin console metrics	Aggregates and calculates the metrics that are displayed in the administration console.		1	dashboard-metric-calc:1.1.7-x86_64
Cloudant DB	The distributed database for Watson Studio. This database contains user ID and the metadata for Watson Studio services.		1	privatecloud-cloudant-repo:v3.13.141-x86_64

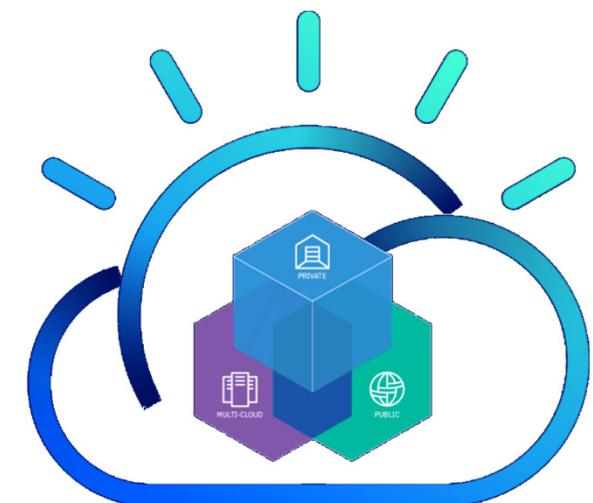
## Manage Services

- You can also click each pod name to get more details about each pod

Services		
Name	Access load balancer	Kubernetes service name ibm-nginx-svc
Description	The load balancer for the Watson Studio client.	
NAME		
ibm-nginx-dfb988cbf-5bm9h	✓	zen0130d-master-3.fyre.ibm.com
ibm-nginx-dfb988cbf-c9pln	✓	zen0130d-master-2.fyre.ibm.com
ibm-nginx-dfb988cbf-qjf7l	✓	zen0130d-master-1.fyre.ibm.com

- If you encounter an issue with any of the services in WSL, contact IBM Software Support for assistance with resolving the issue

# Watson Studio Local Administration Manage Pods



## Manage Pods

- In Kubernetes, a *pod* is a process that is running on your Kubernetes cluster
- A pod is a runnable unit of work, which can be either a stand-alone application or a microservice
- In Watson Studio Local, if a process needs to run on multiple nodes, more pods are created and deployed for extra users or extra notebook servers and IDEs.
  - For example, Spark is deployed as a cluster with at least one pod in each compute node.

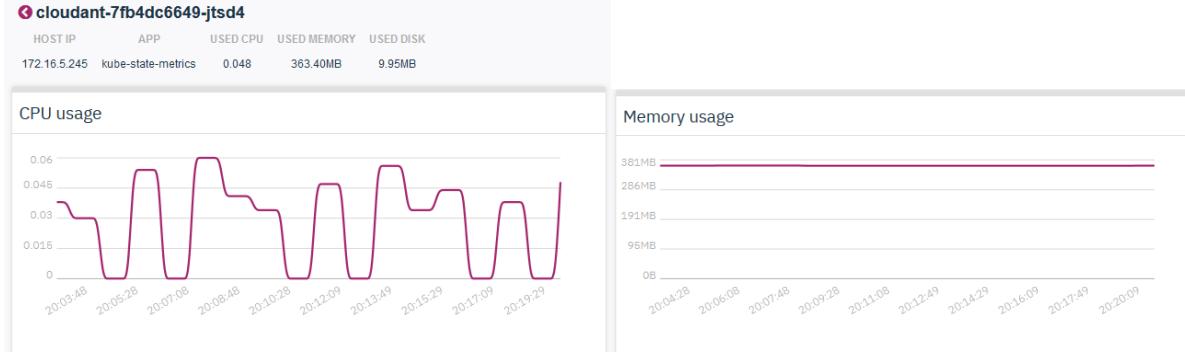
## Manage Pods

- You can view the list of pods that are used in your Watson Studio Local deployment from the **Pods** page
- You can access the **Pods** page from the Menu icon (≡)
- From the **Pods** page you can see:
  - The names of the pods
  - The status of the pods
  - The node where each pod is deployed
  - The CPU, memory, and disk usage for each pod

Pod list						
POD NAME	STATUS	NODE NAME	USED CPU	USED MEMORY	USED DISK	REDEPLOY
cloudant-7fb4dc6649-jtsd4	✓	zen0130d-master-1.fyre.ibm.com	0.00	363MB	10MB	
dash-back-deploy-795b65d87c-f4mmd	✓	zen0130d-worker-1.fyre.ibm.com	0.00	111MB	3MB	

## Manage Pods

- Additionally, you can click on the name of each pod to see:
  - The IP address of the host where the pod is deployed
  - The name of the service that is running on the pod
  - The amount of CPU the pod is using and a graph of the CPU usage over the last 15 minutes
  - The amount of memory that the pod is using and a graph of the memory usage over the last 15 minutes
  - The amount of disk space that the pod is using and a graph of the disk space usage over the last 15 minutes



## Pod Status

- Typically, all of the pods in your environment are running
- However, you might notice that the pods in your environment are in one of the following states:
  - **Pending** (  ): The Kubernetes cluster is creating the Container images that are included in the pod
  - **Succeeded** (  ): All of the Containers in the pod stopped successfully. The Containers are not restarted
  - **Failed** (  ): All of the Containers in the pod stopped successfully, but at least one Container stopped with an error or was stopped by Kubernetes
  - **Unknown** (  ): The state of the pod cannot be determined, which typically occurs when there is a communication issue with the node where the pod is deployed
  - **Running** (  ): The pod is deployed on a node. At least one Container is running or is in the process of starting or restarting.

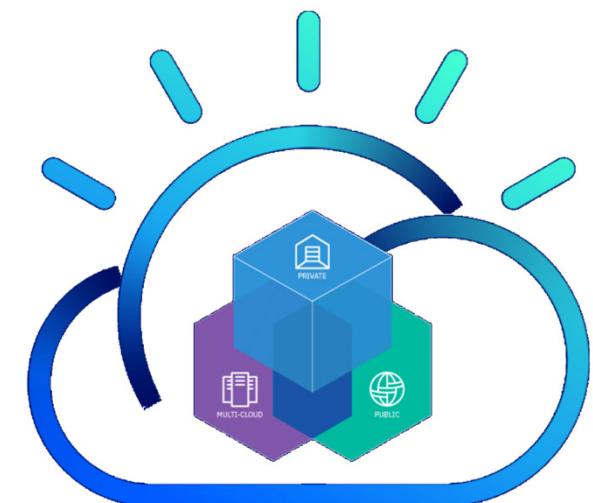
## Pod Status

- If you encounter an issue with a service, you might need to redeploy the pod
- You can redeploy the pod by clicking the redeploy button
- If you cannot successfully redeploy a pod from the Pods page, contact IBM Software Support for assistance with resolving the issue

Pod List						
POD NAME	STATUS	NODE NAME	USED CPU	USED MEMORY	USED DISK	REDEPLOY
cdsx-data-api-1789792843-s2rpd		skeleton-storage-1. 	0.00	949MB	56MB	
cloudant-repo-3127452009-w94hw		skeleton-storage-2. 	0.08	1GB	633MB	
dap-notebooks-ui-347150342-3wct2		skeleton-storage-3. 	0.00	1GB	82MB	

# Watson Studio Local Administration

## View Alerts



## View Alerts

- IBM Watson Studio Local automatically notifies you when a node or pod goes down or when you’re at risk of overloading a resource.
- By default, WSL issues alerts when:
  - CPU usage on a node goes above 90%
  - Memory usage on a node goes above 90%
  - Disk usage on a node goes above 90%
  - A node in the cluster goes down
  - A pod fails
  - A pod is not running or is in an unknown state for more than 5 minutes
- If you want to change the threshold at which alerts are issued, you can configure them on the [Settings page](#).

## View Alerts

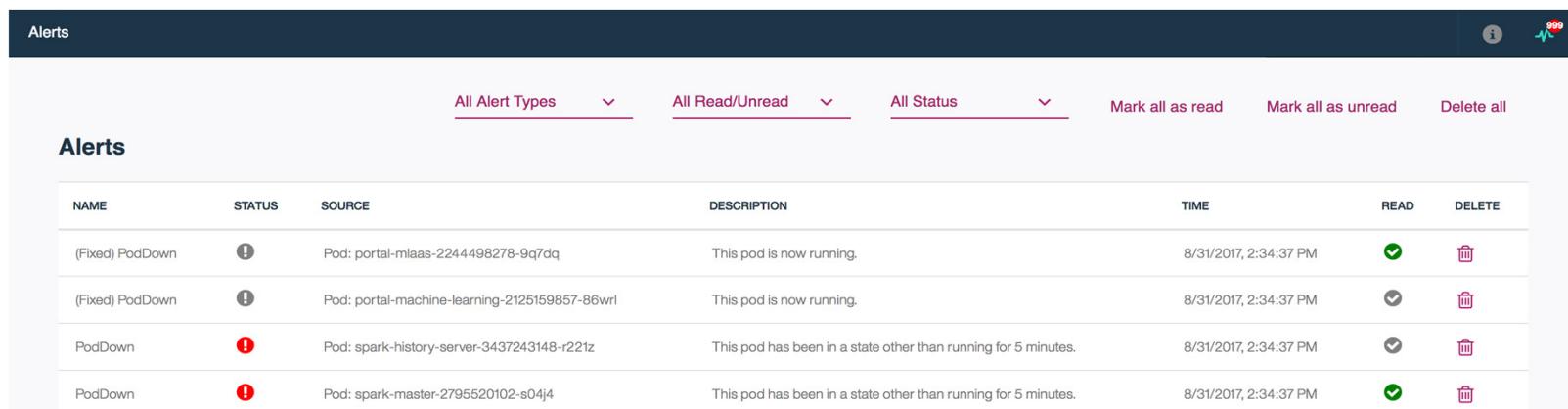
- When you have alerts, the alert icon displays the number of unread alerts in your queue:



- You can access alerts in either of the following ways:
  - If you want a quick peek at your alerts, click the alert icon in the menu bar (  )
  - If you want to manage your alerts, you can access the Alerts page from the menu icon (  )
- **Tip:** When you delete an alert, you can't access it again. Make sure that you don't need it before you delete it

## View Alerts

- From the Alerts page you can:
  - Filter alerts by type
  - Filter alerts based on whether they were read or not
  - Filter alerts by status
  - Mark alerts as read or unread
  - Delete alerts



The screenshot shows the 'Alerts' page with the following interface elements:

- Header:** 'Alerts' tab, three small icons (info, chart, 99+), and a search bar.
- Filtering:** Three dropdown menus: 'All Alert Types', 'All Read/Unread', and 'All Status'. Below them are three buttons: 'Mark all as read', 'Mark all as unread', and 'Delete all'.
- Section Header:** 'Alerts'.
- Table:** A grid displaying five alert entries. The columns are: NAME, STATUS, SOURCE, DESCRIPTION, TIME, READ, and DELETE.
- Data:** The table rows are:

NAME	STATUS	SOURCE	DESCRIPTION	TIME	READ	DELETE
(Fixed) PodDown		Pod: portal-mlaas-2244498278-9q7dq	This pod is now running.	8/31/2017, 2:34:37 PM		
(Fixed) PodDown		Pod: portal-machine-learning-2125159857-86wrl	This pod is now running.	8/31/2017, 2:34:37 PM		
PodDown		Pod: spark-history-server-3437243148-r221z	This pod has been in a state other than running for 5 minutes.	8/31/2017, 2:34:37 PM		
PodDown		Pod: spark-master-2795520102-s04j4	This pod has been in a state other than running for 5 minutes.	8/31/2017, 2:34:37 PM		

## Change alert threshold

- In the Watson Studio Local client, an administrator can optionally adjust when alerts are generated, how long log files and metrics are stored, and how frequently the metrics on the dashboard are refreshed.
- To configure refresh and retention settings:
  1. From the Admin user profile icon, select Settings
  2. In the Refresh and alert settings, adjust the appropriate settings

**Log retention (days)** – the number of days to keep logs before they are automatically deleted. **Default is 10 days.**

**Metrics retention (days)** – the number of days to keep metrics history. **Default is 1 day.**

**Dashboard refresh (seconds)** – the frequency with which the data in the dashboard is refreshed. **Default is 10 seconds.**

**Alert threshold (%)** – the usage threshold at which an alert is triggered. **Default is 90%.**

**Alert warning threshold (%)** – the usage threshold at which a warning is triggered and the node color changes to yellow.

**Default is 70%.**

3. Click Save

# Change alert threshold

## Refresh and alert settings

Log retention (days)

10

Metric retention (days)

1

CPU alert threshold (%)

95

CPU alert warning threshold (%)

75

Memory alert threshold (%)

95

Memory alert warning threshold (%)

75

Disk alert threshold (%)

95

Disk alert warning threshold (%)

75

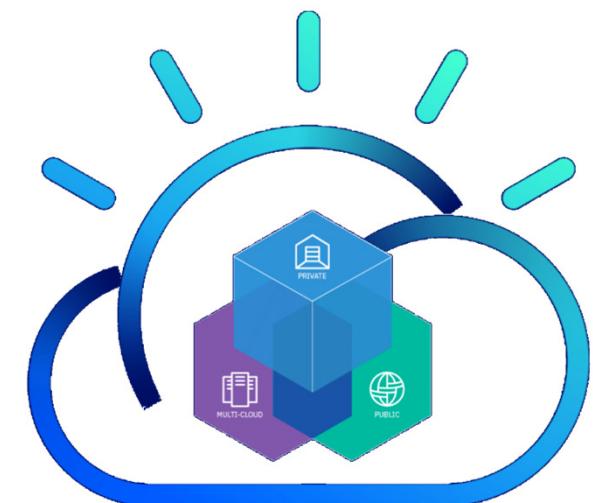
Alert length threshold (minutes)

5

Dashboard refresh (seconds)

10

# Watson Studio Local Administration Manage Library & Package



## Install Custom or 3rd Party Library – Notebooks

- You can add processing capabilities to Apache Spark in WSL by installing external or third-party libraries
- WSL includes many pre-installed libraries
- Before you install a library, check the list of preinstalled libraries. Run the appropriate command from a notebook cell:

- Python: `!pip list --isolated`
- R: `installed.packages()`

- If the library you want to use is not listed, then you can use the commands on the following slides to install it

## Install Local Python Library – Notebooks

- Use the Python pip package installer command to install Python libraries to your notebook
  - For example, using the command “*!pip install prettyplotlib*” installs the prettyplotlib library into a pod’s conda and removes the package once the pod is terminated
  - Or using the command “*!pip install –user prettyplotlib*” installs the prettyplotlib library in your user home and the package exists even after the pod is terminated
- Use the Python import command to import the library components
  - Going off our example above, you would then use the command “*import prettyplotlib as ppl*”
- Restart the kernel

## Install Global Library & Package

- A WSL administrator can install Python or R packages in global directories. These packages are then made available to all users
- To install a global Python library:
  - Log in to WSLL as admin and create a Python notebook
  - Use the Python pip package installer command to install Python libraries to your notebook for example see the image below:

```
!pip install --target  
/user-home/_global_/python-2.7 prettyplotlib
```

- The installed packages can be used by all notebook users that use the same Python version in the Spark service
- Notebook users can now use the Python import command to import the library components.
  - For example, users can run the following command in a code cell:

## Install Global Library & Package

- To install a global R package:
  - Log in to WSL as admin and create an R notebook
  - Use the R `install.packages()` function to install new R packages. For example, running the following command would install the `ggplot2` package

```
install.packages("ggplot2")
```

- The imported package can be used by all R notebooks that are running in the Spark service
- Now, users can use the R `library()` function to load the installed package
  - For example, a user can run the following command in a code cell:

```
library("ggplot2")
```

# Manage Packages

## ▪ Administrator

A WSL administrator can install Python or R packages in global directories. These packages are available to all users on the cluster

- **Add a global Spark JAR file**

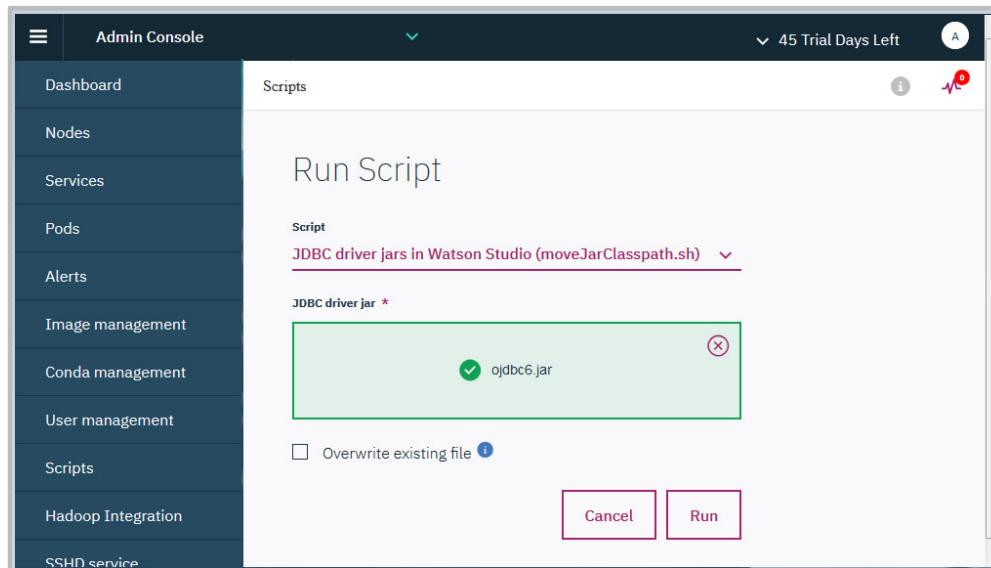
- In the Scripts panel of the Admin console, an administrator can select “Add Spark jars in Watson Studio (moveJarClasspath.sh) to upload JAR files to the /user-home/\_global\_/spark/jars/ directory for use with Spark.

# Manage Packages

## ▪ Administrator

- Add a JDBC driver

- In the Admin Console, click the menu icon (≡) and click Scripts.
- In the Scripts pull-down menu, select JDBC driver jars in Watson Studio (moveJarClasspath.sh), and click add jar.



# Manage Packages

## ▪ Administrator

- **Install a global Python package**

- Log in to WSL as the default administrator (ie. Admin) and create a Python notebook.
- Use the Python pip package installer command to install Python libraries in the Python notebook.  
**Example:**  
`!pip install --target /user-home/_global_/python-2.7 prettyplotlib`
- Once the package is installed, it can be used by all notebook users. Example from above installation, users can then use the library simply do: Ex: import prettyplotlib as ppl

# Manage Packages

## ▪ WSL Local User

- Local users can add additional packages to a base image, which can then be reused by other users.
- WSL includes many preinstalled libraries. Check the [list](#) before you install a library. (or type command: “pip list” for Python, “installed.packages()” for R.)

### • Install a Python library

- Use the Python conda install. Ex: !conda install prettyplotlib

- Use custom functions/libraries in Python scripts

- **Example:**

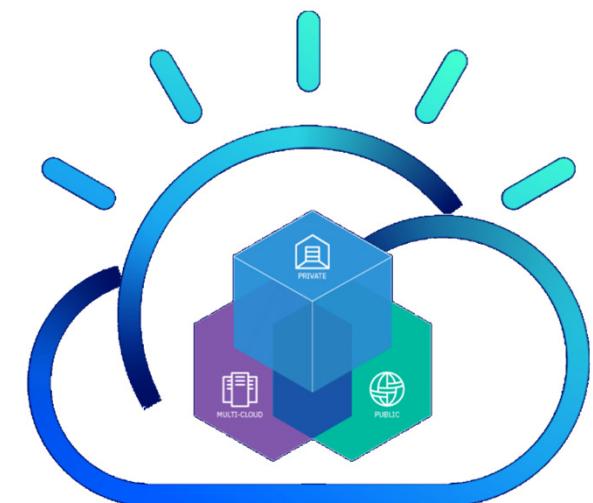
```
sys.path.insert(0, ‘..scripts/’)
from your_custom_script import *
```

### • Install a R library

- Example: install.packages(“ggplot2”)

# Watson Studio Local

## Administration Hadoop Integration



## Objective Overview

- Hadoop infrastructure is a huge investment but the tools in the ecosystem are not intuitive for business analyst and users. Most of your data is probably available within your HDFS along with your huge compute capacity but access to it is difficult.
- Build machine learning and analytics on Watson Studio that provides variety of open source libraries and frameworks such as Spark, R, Keras, TensorFlow, Scikit-learn as a service within a kubernetes based platform behind your firewall. Along with cloud agility the platform provides end-end data management and analytics capability with collaboration and lifecycle management functions.
- Hadoop connector provides seamless access to compute power available along with easy access to hdfs data.
- Accelerate your machine learning deployments in hours as oppose to months. Build quick to market business decision making Artificial Intelligence systems easily and quickly.

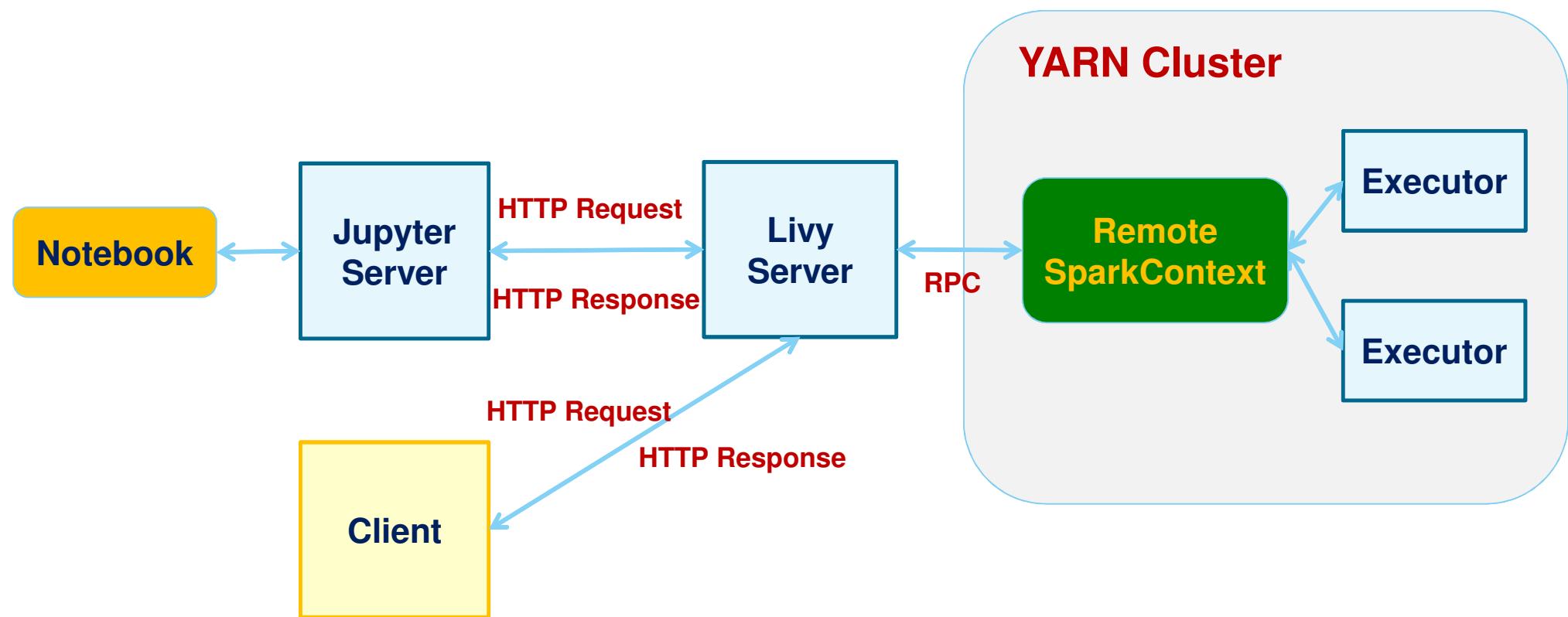
## Hadoop Integration – Overview

- The Watson Studio Local Hadoop Integration Service is a registration service that can be installed on a Hadoop edge node.
- WSL Version 1.2 or later clusters can securely access data residing on the Hadoop cluster, submit interactive Spark jobs, build models, and schedule jobs that run as a YARN application.

**Important:** Your Hadoop admin must install the Hadoop registration service on a Hadoop cluster, add the Watson Studio Local cluster to the Hadoop registration to be managed, and provide the Hadoop registration service user ID and the URLs.

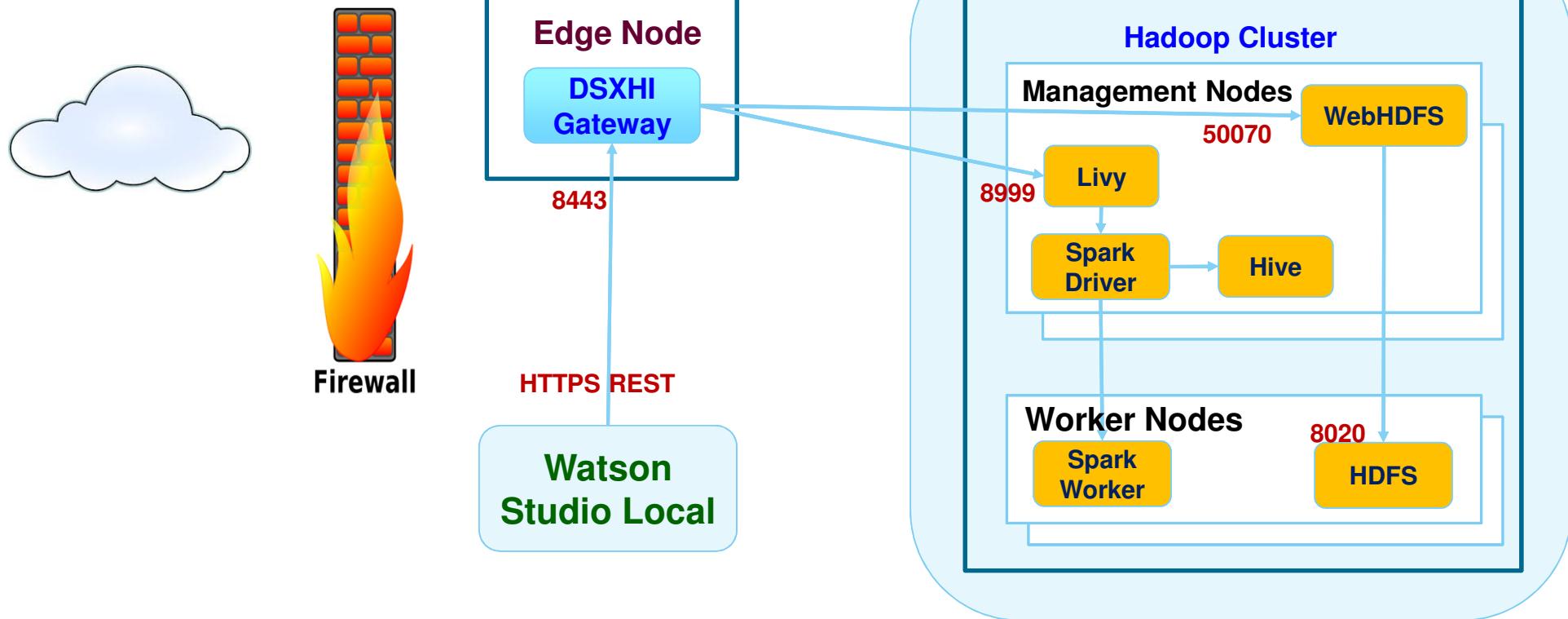
# Livy Interface with Spark

## Architecture I



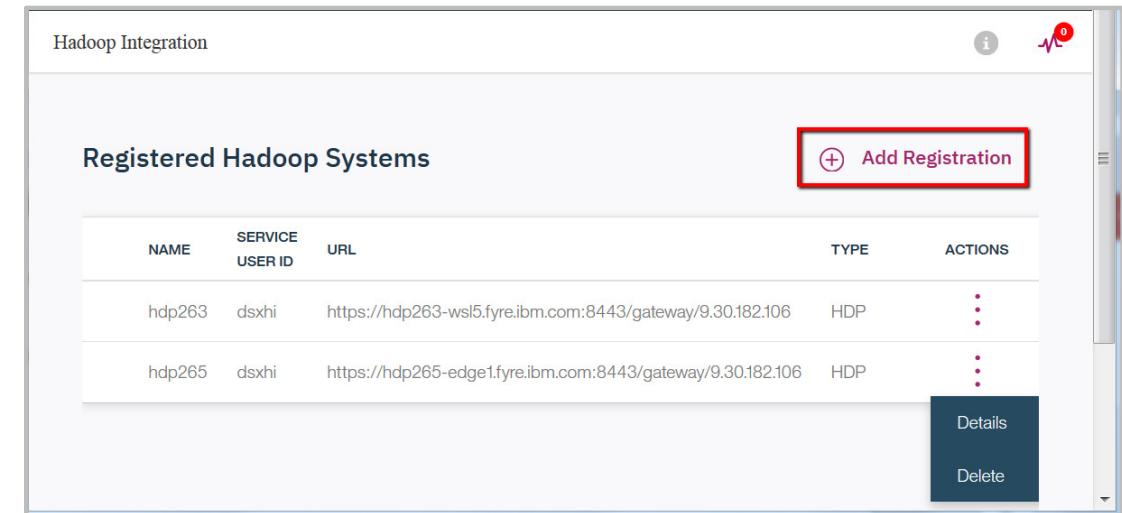
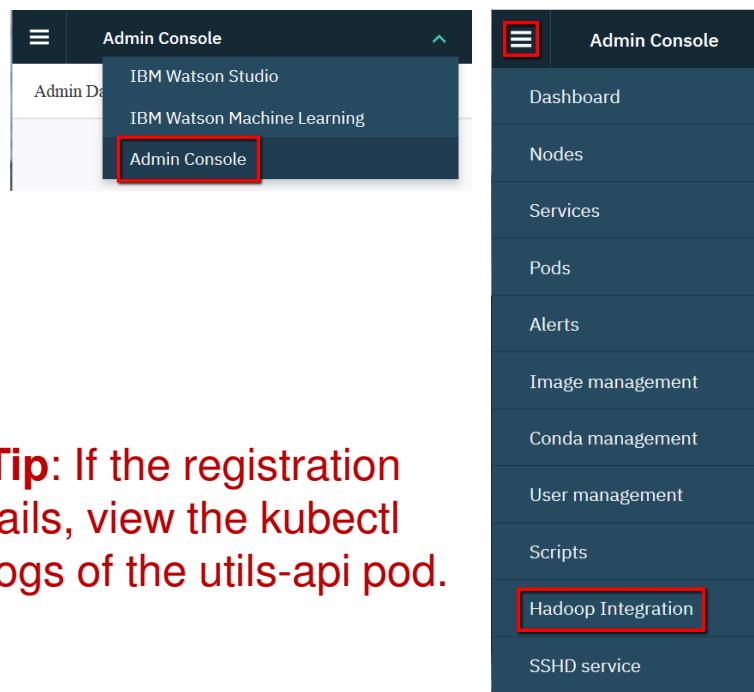
# Hadoop Connector Installation & Setup

## Architecture



# Hadoop Integration – Register a Hadoop Cluster

- In the Admin Console, click the menu icon (☰) and click **Hadoop Integration**.
- HDFS and Hive data sources are automatically created when a Hadoop registration service is registered in WSL admin console.



The image shows the 'Registered Hadoop Systems' page in the Admin Console. It lists two registered systems: 'hdp263' and 'hdp265'. Each entry includes columns for NAME, SERVICE USER ID, URL, TYPE, and ACTIONS. The 'Actions' column for each entry has a 'Details' button and a 'Delete' button. A red box highlights the 'Add Registration' button in the top right corner of the table header.

NAME	SERVICE USER ID	URL	TYPE	ACTIONS
hdp263	dsxhi	https://hdp263-wsl5.fyre.ibm.com:8443/gateway/9.30.182.106	HDP	<span>⋮</span>  <span>Details</span> <span>Delete</span>
hdp265	dsxhi	https://hdp265-edge1.fyre.ibm.com:8443/gateway/9.30.182.106	HDP	<span>⋮</span>  <span>Details</span> <span>Delete</span>

**Tip:** If the registration fails, view the kubectl logs of the utils-api pod.

## Hadoop Integration – View details about the registered Hadoop cluster

If Livy for Spark and/or Livy for Spark2 services are exposed, WSL users can list these endpoints through `dsx_core_utils` and `dsxCoreUtilsR`, and use them as defined Livy endpoint in Jupyter, RStudio and Zeppelin notebooks.

### Python syntax:

```
%python
import dsx_core_utils;
dsx_core_utils.list_dsxhi_livy_endpoints();
```

### R syntax:

```
library('dsxCoreUtilsR');
dsxCoreUtilsR::listDSXHILivyEndpoints()
```

## Hadoop Integration

FYI. When a Watson Studio Local administrator pushes a custom image to a registered Hadoop system, a notebook can initialize a remote Spark Livy session which runs within an environment associated with that remote custom image:

```
import dsx_core_utils
%load_ext sparkmagic.magics

# Retrieve a list of registered Hadoop Integration systems.
DSXHI_SYSTEMS = dsx_core_utils.get_dsxhi_info(showSummary=True)

Available Hadoop systems:

systemName    LIVYSPARK   LIVYSPARK2  imageId
0      hdp263  livyspark  livyspark2
1      hdp265  livyspark  livyspark2
```

```
session_name = 'spark2_0205'
livy_endpoint = HI_CONFIG['LIVY']
webhdfs_endpoint = HI_CONFIG['WEBHDFS']
%spark add -s $session_name -l python -u $livy_endpoint
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
12	application_1549042673081_0016	pyspark	idle	<a href="#">Link</a>	<a href="#">Link</a>	✓

# Hadoop Integration – Hadoop Gateway Installation & Setup

## Installation

- **Pre-req**

- Install Hadoop clients on an edge node (\*\*Recommended) or a node that will host gateway service
- Create a gateway service id on all Hadoop cluster nodes & edge node
  - ✓ Ex. # useradd -u 3001 -g hdfs dsxhi (all Hadoop cluster nodes\*\*)
- Create HDFS directories
  - ✓ Ex. # su - hdfs -c 'hdfs dfs -mkdir -p /user/dsxhi/environments'
  - ✓ # su - hdfs -c 'hdfs dfs -chown -R dsxhi:hdfs /user/dsxhi'
- Collect port information for HDFS, WebHDFS, Livy\*\* (ex. 8020, 50070, 8999)

# Hadoop Integration – Hadoop Gateway Installation & Setup

## Installation

- **Pre-req**

- Add Hadoop service parameters for gateway service id (listed in below)

- ✓ 1. HDFS custom core-site

hadoop.proxyuser.<service\_id>.groups=\*

hadoop.proxyuser.<service\_id>.hosts=\*

- ✓ 2. Hive custom webhcat-site

webhcat.proxyuser. <service\_id>.hosts=\*

webhcat.proxyuser.<service\_id>.groups=\*

- ✓ 3. Spark2 custom livy2-conf

livy.superusers=<service\_id>

# Hadoop Integration – Hadoop Gateway Installation & Setup

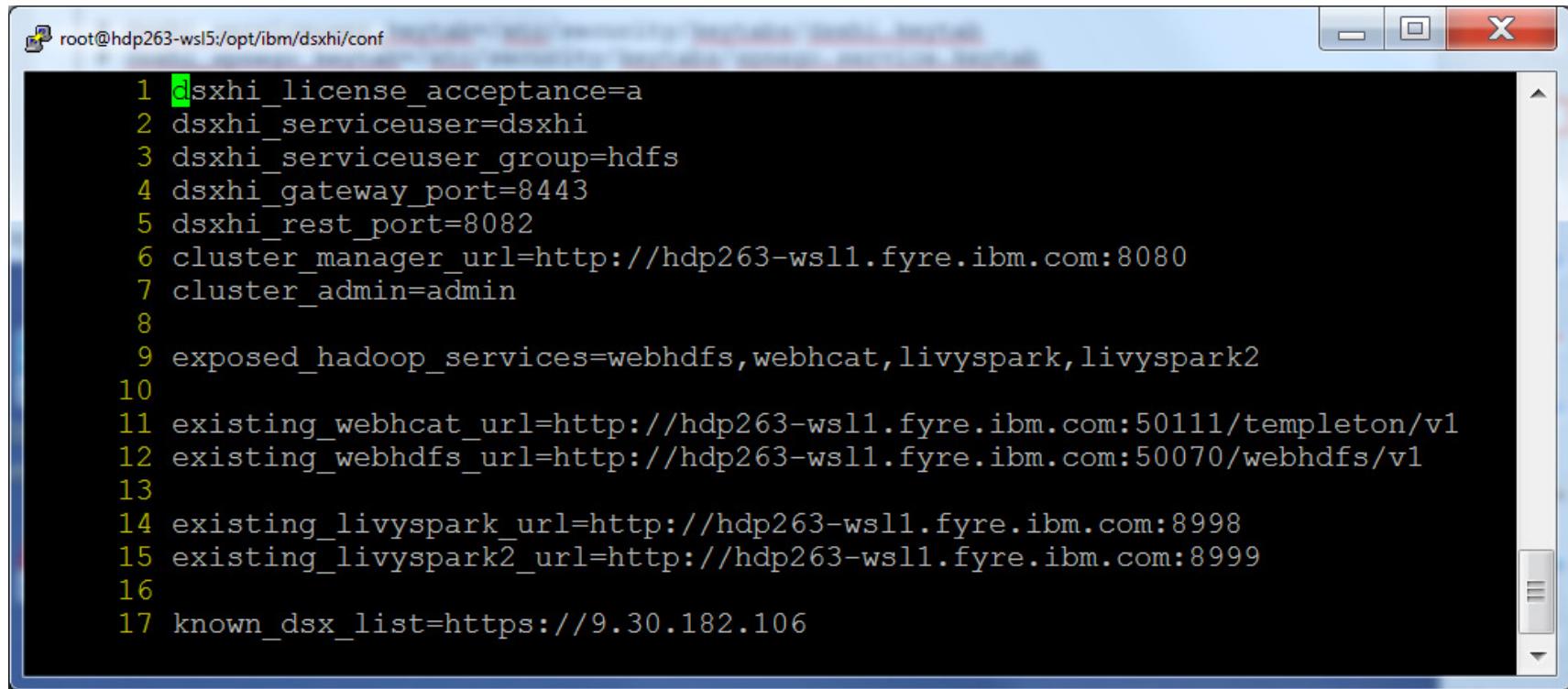
## Installation

- **Install Livy connector (dsxhi) rpm package**
  - Download connector rpm file from Passport Advantage
  - Install rpm file (Ex. rpm -ivh dsxhi-icp4data-dsp-1.1.1.0-64.noarch.rpm)
- **Configure Livy connector**
  - Modify configuration files
    - ✓ # cp /opt/ibm/dsxhi/conf/dsxhi\_install.conf.template.HDP /opt/ibm/dsxhi/conf/dsxhi\_install.conf
    - ✓ # vi /opt/ibm/dsxhi/conf/dsxhi\_install.conf

# Hadoop Integration – Hadoop Gateway Installation & Setup

## Installation

- ✓ Example:



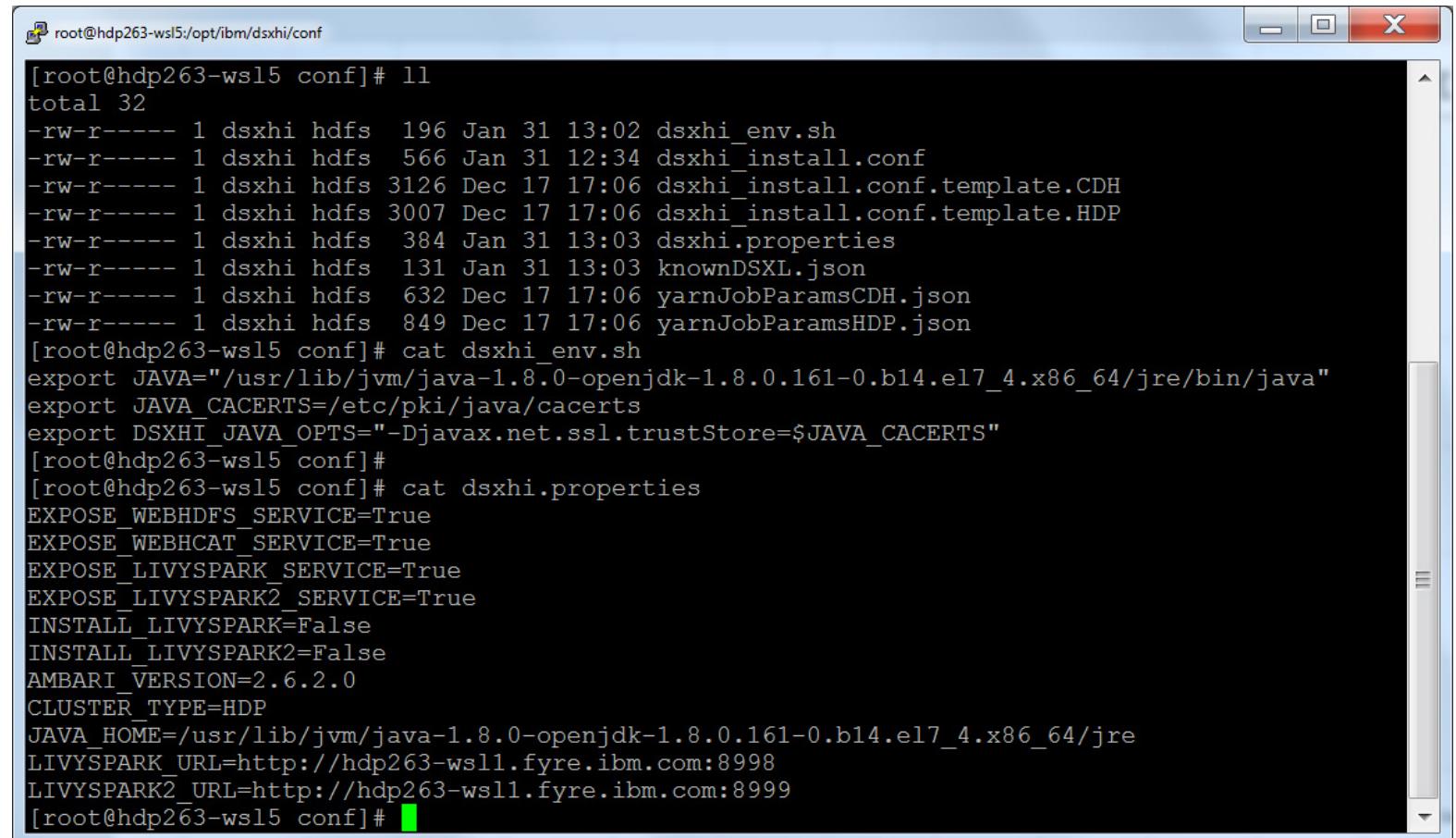
A screenshot of a terminal window titled "root@hdp263-ws15:/opt/ibm/dsxhi/conf". The window displays a configuration file with the following content:

```
1 dsxhi_license_acceptance=a
2 dsxhi_serviceuser=dsxhi
3 dsxhi_serviceuser_group=hdfs
4 dsxhi_gateway_port=8443
5 dsxhi_rest_port=8082
6 cluster_manager_url=http://hdp263-ws11.fyre.ibm.com:8080
7 cluster_admin=admin
8
9 exposed_hadoop_services=webhdfs,webhcatt,livyspark,livyspark2
10
11 existing_webhcatt_url=http://hdp263-ws11.fyre.ibm.com:50111/templeton/v1
12 existing_webhdfs_url=http://hdp263-ws11.fyre.ibm.com:50070/webhdfs/v1
13
14 existing_livyspark_url=http://hdp263-ws11.fyre.ibm.com:8998
15 existing_livyspark2_url=http://hdp263-ws11.fyre.ibm.com:8999
16
17 known_dsx_list=https://9.30.182.106
```

# Hadoop Integration – Hadoop Gateway Installation & Setup

## Installation

✓ Example:



The screenshot shows a terminal window titled "root@hdp263-ws15: /opt/ibm/dsxhi/conf". The window displays several command-line operations:

- The user runs "ls" to list files in the directory, showing 32 total files.
- The user runs "cat dsxhi\_env.sh" to view the contents of the environment script, which includes setting JAVA\_HOME and JAVA\_CACERTS, and defining DSXHI\_JAVA\_OPTS.
- The user runs "cat dsxhi.properties" to view the properties file, which contains settings for various services like WEBHDFS, WEBHCAT, LIVYSPARK, and LIVYSPARK2, along with cluster type and Java home definitions.

# Hadoop Integration – Hadoop Gateway Installation & Setup

## Installation

- **Install Livy connector (dsxhi)**

- Install connector

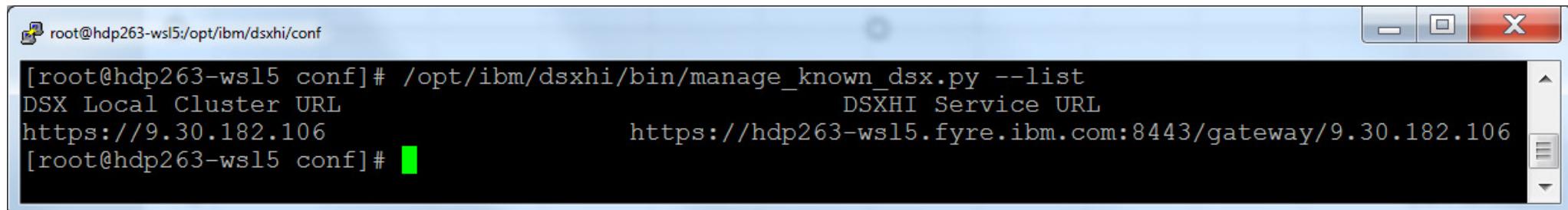
```
# /opt/ibm/dsxhi/bin/install.py --dsxhi_gateway_master_password=<password> --  
password=<password>
```

```
IBM Data Science Experience Hadoop Integration Service  
-----  
Terms and Conditions: http://www14.software.ibm.com/cgi-  
bin/weblap/lap.pl?la_formnum=&li_formnum=L-KLSY-  
AVZW2D&title=IBM+Data+Science+Experience+Hadoop+Integration&l=en  
  
--Determining properties  
--Running the prechecks  
  
--Install Livy for Spark / Spark 2  
--Configure gateway  
--Create template for gateway  
--Setting up known DSX Local clusters  
--Install dsxhi_rest  
--Start all services  
--Install finished. Check status in /var/log/dsxhi/dsxhi.log
```

# Hadoop Integration – Hadoop Gateway Installation & Setup

## Verify Installation

- **Livy connector process status check**
  - /opt/ibm/dsxhi/bin/status.py
- **List registered WSL cluster(s)**
  - /opt/ibm/dsxhi/bin/manage\_known\_dsx.py –list
- **Register/Add new WSL cluster**
  - /opt/ibm/dsxhi/bin/manage\_known\_dsx.py --add https://<WSL\_FQDN>



```
[root@hdp263-ws15:/opt/ibm/dsxhi/conf]# /opt/ibm/dsxhi/bin/manage_known_dsx.py --list
DSX Local Cluster URL                               DSXHI Service URL
https://9.30.182.106                                https://hdp263-ws15.fyre.ibm.com:8443/gateway/9.30.182.106
[root@hdp263-ws15 conf]#
```

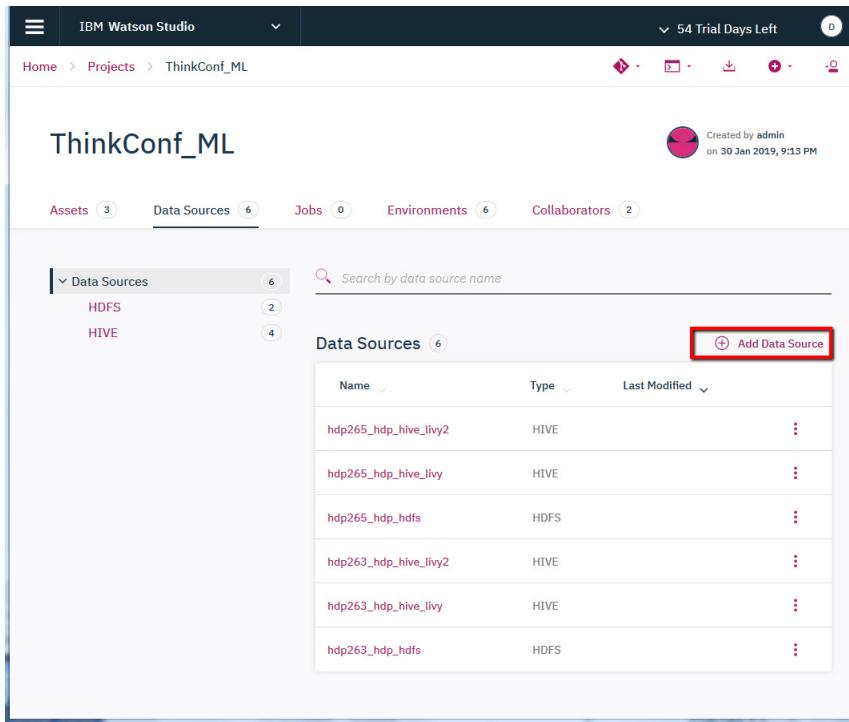
## Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

- Retrieve data from HDFS file system
- Access to Hive database table data
- Push down Spark job/task to the remote Spark2 server on Hadoop cluster

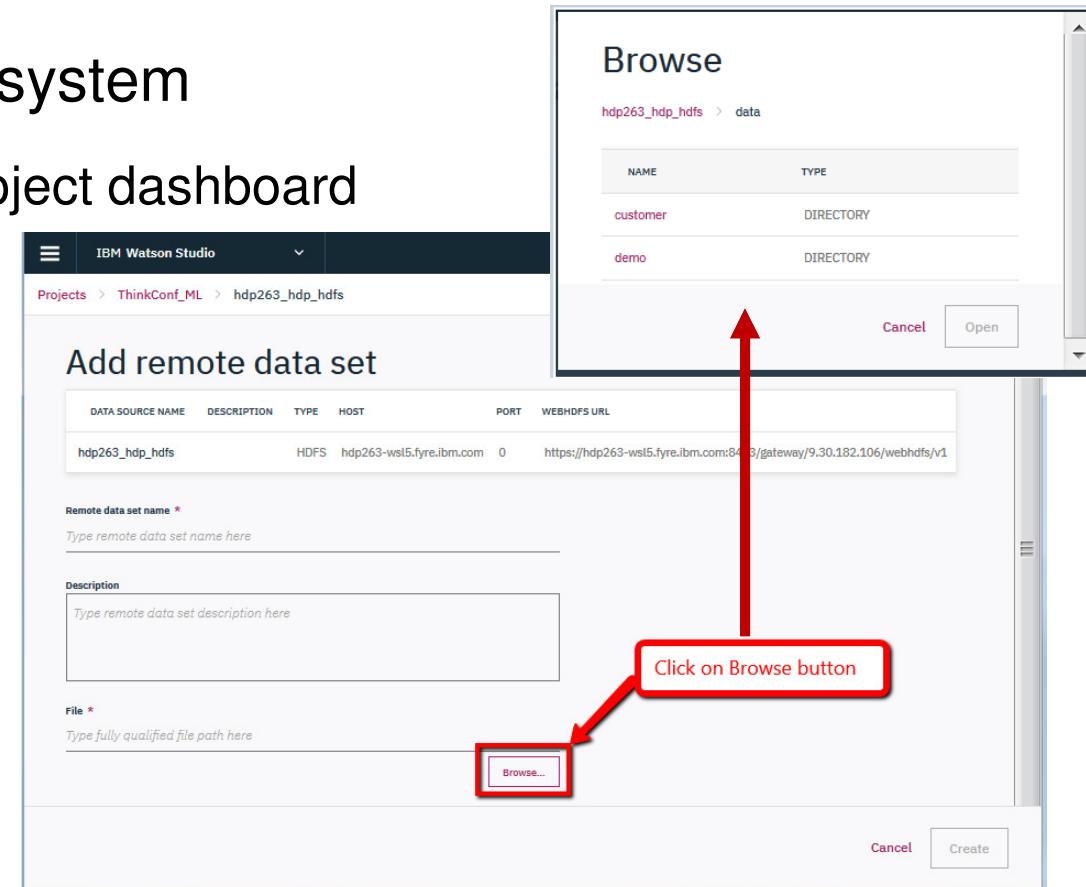
# Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

## 1. Retrieving data from HDFS file system

- Add a remote data source from project dashboard



The screenshot shows the IBM Watson Studio interface for a project named "ThinkConf\_ML". In the top navigation bar, there are tabs for Home, Projects, ThinkConf\_ML, and several status indicators like Trial Days Left (54). Below the navigation, there are sections for Assets, Data Sources (6), Jobs (0), Environments (6), and Collaborators (2). The "Data Sources" section is currently selected. It displays a list of existing data sources: "hdp265\_hdp\_hive\_livy2" (HIVE), "hdp265\_hdp\_hive\_livy" (HIVE), "hdp265\_hdp\_hdfs" (HDFS), "hdp263\_hdp\_hive\_livy2" (HIVE), "hdp263\_hdp\_hive\_livy" (HIVE), and "hdp263\_hdp\_hdfs" (HDFS). A red box highlights the "Add Data Source" button at the bottom right of the list.

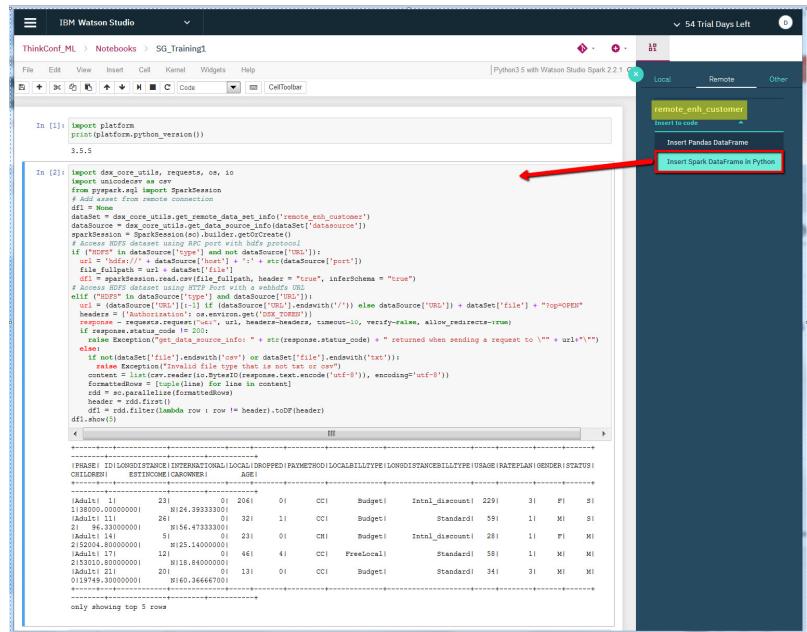


The screenshot shows the "Add remote data set" dialog box. At the top, it displays the configuration for an existing data source: "hdp263\_hdp\_hdfs" (HDFS) with host "hdp263-wsl5.fyre.ibm.com" and port "0". Below this, there are fields for "Remote data set name" (with placeholder "Type remote data set name here"), "Description" (placeholder "Type remote data set description here"), and "File" (placeholder "Type fully qualified file path here"). A red box highlights the "Browse..." button next to the "File" input field. A red arrow points from this highlighted button to a larger callout bubble containing the text "Click on Browse button". In the bottom right corner of the dialog, there are "Cancel" and "Create" buttons.

## Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

## 2. Use added remote HDFS data source in Notebooks

- Click on icon (  ) and select the Remote data source created from previous instruction; then select “Insert Spark DataFrame in Python”.



## Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

### 3. Load HDFS dataset via remote Spark session

- Example code:

```
import dsx_core_utils, requests, jaydebeapi, os, io, sys
import pandas as pd
url = 'https://hdp263-
wsl5.fyre.ibm.com:8443/gateway/9.30.182.106/webhdfs/v1/data/demo/customer/enhanced_customer.csv?OP=OPEN'
headers = {'Authorization': os.environ.get('DSX_TOKEN')}
response = requests.request("GET", url, headers=headers, timeout=10, verify=False, allow_redirects=True)
df = pd.read_csv(io.StringIO(response.text, newline=None), sep=',')
df.head()
```

```
In [6]: import dsx_core_utils, requests, jaydebeapi, os, io, sys
import pandas as pd
url = 'https://hdp263-wsl5.fyre.ibm.com:8443/gateway/9.30.182.106/webhdfs/v1/data/demo/customer/enhanced_customer.csv?OP=OPEN'
headers = {'Authorization': os.environ.get('DSX_TOKEN')}
response = requests.request("GET", url, headers=headers, timeout=10, verify=False, allow_redirects=True)
df = pd.read_csv(io.StringIO(response.text, newline=None), sep=',')
df.head()
```

Out [6]:

	PHASE	ID	LONGDISTANCE	INTERNATIONAL	LOCAL	DROPPED	PAYMETHOD	LOCALBILLTYPE	LONGDISTANCEBILLTYPE	USAGE	RATEPLAN	GENDER
0	Adult	1	23	0	206	0	CC	Budget	Intl_discount	229	3	F
1	Adult	11	26	0	32	1	CC	Budget	Standard	59	1	M
2	Adult	14	5	0	23	0	CH	Budget	Intl_discount	28	1	F
3	Adult	17	12	0	46	4	CC	FreeLocal	Standard	58	1	M
4	Adult	21	20	0	13	0	CC	Budget	Standard	34	3	M

## Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

### 4. Load HDFS dataset via “PyWebHdfs” package

```
import json
from pywebhdfs.webhdfs import PyWebHdfsClient
headers = {'Authorization': os.environ.get('DSX_TOKEN')}
url = 'https://hdp263-wsl5.fyre.ibm.com:8443/gateway/9.30.182.106/webhdfs/v1'
hdfs = PyWebHdfsClient(base_uri_pattern=url, request_extra_opts={'verify': False}, request_extra_headers=headers)
print(json.dumps(hdfs.list_dir('/'), indent=1))
```

```
In [19]: import json
from pywebhdfs.webhdfs import PyWebHdfsClient
headers = {'Authorization': os.environ.get('DSX_TOKEN')}

url = 'https://hdp263-wsl5.fyre.ibm.com:8443/gateway/9.30.182.106/webhdfs/v1'
hdfs = PyWebHdfsClient(base_uri_pattern=url, request_extra_opts={'verify': False}, request_extra_headers=headers)
print(json.dumps(hdfs.list_dir('/'), indent=1))

{
  "FileStatuses": [
    "FileStatus": [
      {
        "group": "hadoop",
        "modificationTime": 1548968993029,
        "permission": "777",
        "pathSuffix": "app-logs",
        "type": "DIRECTORY",
        "fileId": 16396,
        "blockSize": 0,
        "accessTime": 0,
        "storagePolicy": 0,
        "owner": "yarn",
        "length": 0,
        "replication": 0,
        "childrenNum": 4
      },
      {
        "group": "hadoop",
        "modificationTime": 1548968993029,
        "permission": "777",
        "pathSuffix": "app-logs",
        "type": "DIRECTORY",
        "fileId": 16397,
        "blockSize": 0,
        "accessTime": 0,
        "storagePolicy": 0,
        "owner": "yarn",
        "length": 0,
        "replication": 0,
        "childrenNum": 4
      }
    ]
  ]
}
```

# Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

## 5. Access Hive data

- %%spark -s <session\_name> -c sql -o <local\_dataframe>  
select gender, count(gender) from ext\_customer group by gender order by gender  
print(type(df\_ex\_cust\_local))  
print(df\_ex\_cust\_local)

```
In [9]: %%spark -s $session_name -c sql
show tables
```

```
Out[9]:


|   | database | tableName    | isTemporary |
|---|----------|--------------|-------------|
| 0 | default  | del2         | False       |
| 1 | default  | ext_customer | False       |


```

```
In [12]: %%spark -s $session_name -o df_ex_cust_local -c sql
select gender, count(gender) from ext_customer group by gender order by gender
```

```
Out[12]:


|   | gender | count(gender) |
|---|--------|---------------|
| 0 | F      | 1316          |
| 1 | GENDER | 1             |
| 2 | M      | 750           |


```

```
In [15]: print(type(df_ex_cust_local))
print(df_ex_cust_local)
```

```
<class 'pandas.core.frame.DataFrame'>
   gender  count(gender)
0        F          1316
1    GENDER          1
2        M          750
```

# Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

## 6. Push down Spark processing

```
In [25]: import dsx_core_utils
%load_ext sparkmagic.magics
# Retrieve a list of registered Hadoop Integration systems.
DSXHI_SYSTEMS = dsx_core_utils.get_dsxhi_info(showSummary=True)

The sparkmagic.magics extension is already loaded. To reload it, use:
%reload_ext sparkmagic.magics
Available Hadoop systems:

   systemName    LIVYSPARK    LIVYSPARK2 imageId
0      hdp263    livyspark    livyspark2
1      hdp265    livyspark    livyspark2
```

```
In [26]: # Set up sparkmagic to connect to the selected registered HI systemName above.
myConfig={
    "queue": "default",
    "driverMemory": "512M",
    "numExecutors": 1,
    "executorMemory": "512M"
}
HI_CONFIG = dsx_core_utils.setup_livy_sparkmagic(
    system="hdp263",
    livy="livyspark2",
    imageId=None,
    addlConfig=myConfig)
# (Re-)load spark magic to apply the new configs.
%reload_ext sparkmagic.magics
```

sparkmagic has been configured to use <https://hdp263-wsl5.fyre.ibm.com:8443/gateway/9.30.182.106/livy2/v1>  
success configuring sparkmagic livy.

# Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

## 6. Push down Spark processing

```
In [27]: %spark cleanup
session_name = 'spark2_0205a'
livy_endpoint = HI_CONFIG['LIVY']
webhdfs_endpoint = HI_CONFIG['WEBHDFS']
%spark add -s $session_name -l python -u $livy_endpoint
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
4	application_1549042673081_0007	pyspark	idle	<a href="#">Link</a>	<a href="#">Link</a>	✓

SparkSession available as 'spark'.

```
In [28]: %spark info
```

```
Info for running Spark:
Sessions:
  Name: spark2_0205a    Session id: 4    YARN id: application_1549042673081_0007 Kind: pyspark    State: idle
  Spark UI: http://hdp263-ws12.fyre.ibm.com:8088/proxy/application\_1549042673081\_0007/
  Driver Log: http://hdp263-ws15.fyre.ibm.com:8042/node/containerlogs/container\_e02\_1549042673081\_0007\_01\_000001/dsxhi
Session configs:
  {'conf': {'spark.yarn.appMasterEnv.HI_UTILS_PATH': '/user/dsxhi/environments/pythonAddons/hi_core_utils.zip'}, 'proxyUser': 'dsxhi', 'numExecutors': 1, 'queue': 'default', 'driverMemory': '512M', 'executorMemory': '512M'}
```

# Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

## 6. Push down Spark processing

```
In [29]: %%spark -s $session_name
import socket
print("Remote livy session driver: {}".format(socket.gethostname()))

Remote livy session driver: hdp263-wsl5.fyre.ibm.com

In [30]: %%spark -s $session_name
import socket
print("Remote livy session driver: {}".format(socket.gethostname()))

Remote livy session driver: hdp263-wsl5.fyre.ibm.com

In [32]: %%spark

file_customer = "hdfs:///data/demo/customer/enhanced_customer.csv"
df_cust = spark.read.format("org.apache.spark.sql.execution.datasources.csv.CSVFileFormat").option("header", "true").option("inferSchema", "true").load(file_customer)
df_cust.printSchema()

print("Total number of customer dataset row count: {}\n".format(df_cust.count()))

root
|-- PHASE: string (nullable = true)
|-- ID: integer (nullable = true)
|-- LONGDISTANCE: integer (nullable = true)
|-- INTERNATIONAL: integer (nullable = true)
|-- LOCAL: integer (nullable = true)
|-- DROPPED: integer (nullable = true)
|-- PAYMETHOD: string (nullable = true)
|-- LOCALBILLTYPE: string (nullable = true)
|-- LONGDISTANCEBILLTYPE: string (nullable = true)
|-- USAGE: integer (nullable = true)
|-- RATEPLAN: integer (nullable = true)
|-- GENDER: string (nullable = true)
|-- STATUS: string (nullable = true)
|-- CHILDREN: integer (nullable = true)
|-- ESTINCOME: double (nullable = true)
|-- CAROWNER: string (nullable = true)
|-- AGE: double (nullable = true)

Total number of customer dataset row count: 2066
```

## Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

### 6. Push down Spark processing – Review the remote Spark application

**loop**

**FINISHED Applications**

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	
20	0	0	20	0	0 B	10 GB	0 B	0	12	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocat
Capacity Scheduler	[MEMORY]	<memory:1536, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State
application_1549042673081_0015	dsxhi	DSXHI	YARN	default	0	Thu Feb 7 08:44:24 -0800 2019	Thu Feb 7 08:44:31 -0800 2019	FINISHED
application_1549042673081_0014	dsxhi	batch-score-a8f0d693-460b-40a4-bf2c-f65b9da79847	SPARK	default	0	Thu Feb 7 08:10:42 -0800 2019	Thu Feb 7 08:12:06 -0800 2019	FINISHED
application_1549042673081_0013	dsxhi	livy-session-10	SPARK	default	0	Wed Feb 6 21:55:08 -0800 2019	Wed Feb 6 22:14:26 -0800 2019	FINISHED
application_1549042673081_0012	dsxhi	batch-score-789e5bc2-21dd-43ec-9667-d0be6488c7aa	SPARK	default	0	Wed Feb 6 21:43:47 -0800 2019	Wed Feb 6 21:50:54 -0800 2019	FINISHED

## Hadoop Integration – Hadoop Gateway Testing & Troubleshoot

Delete the remote Spark session\*\*

- %spark delete -s <session\_name>

```
In [38]: %spark delete -s $session_name
```

```
In [23]: %spark cleanup
```

```
In [*]: %%javascript  
Jupyter.notebook.session.delete();
```

# Watson Studio Local Administration Best Practices



## Best Practices

- **Access large amount of data – with relational databases**

When you load gigabytes of data from tables, no amount of memory you reserve will be adequate. For better practice, consider following

- You can load and process by batches. Instead of `fetchall()` use `batch` similar to the following example:

```
cur.execute('select * from table1')  
for row in cur:  
    print row  
    #perform processing...
```

## Best Practices

- **Access large amount of data – with relational databases**

- Or add criteria in your query filter. Example

```
cur = conn.cursor()  
  
cur.execute("SELECT * FROM large_table WHERE geo = 'NA'")
```

- Using SparkSQL and DataFrame to provide faster data access compared to other database JDBC clients.
  - Consider letting RDBMS process data as much as possible, and extract only the subset.
  - Define stored procedures that can perform most of the filtering and processing on your relational database.

## Best Practices

- **Access large amount of data – with Spark and Hadoop**
  - As a best practice, processing should be close to your data. If you need to use large amount of data in HDFS or Hive, It's best to use the Spark running on Hadoop cluster through Livy.
  - Take advantage of data localization, and avoids data transfer to speed up your processing.
  - All runtime environments in WSL (Jupyter, Zeppelin and RStudio) support accessing Hadoop Spark through Livy. More details refer to: [remote spark documentation](#)

# Best Practices

## ▪ Performance Tuning

- Generally, WSL reads and writes to relational databases using JDBC and modules JayDeBeApi and RJDBC/SparkR.
- Using database vendor specific clients such as cx\_Oracle and pymssql can provide comparable performance to SparkSQL.
- WSL writes database specific data frames using SQLAlchemy and database vendor specific adapters such as ibm\_db\_sa.
- WSL does NOT support ODBC as the moment.

## Best Practices

### ▪ Performance Tuning – Tune Zeppelin notebook

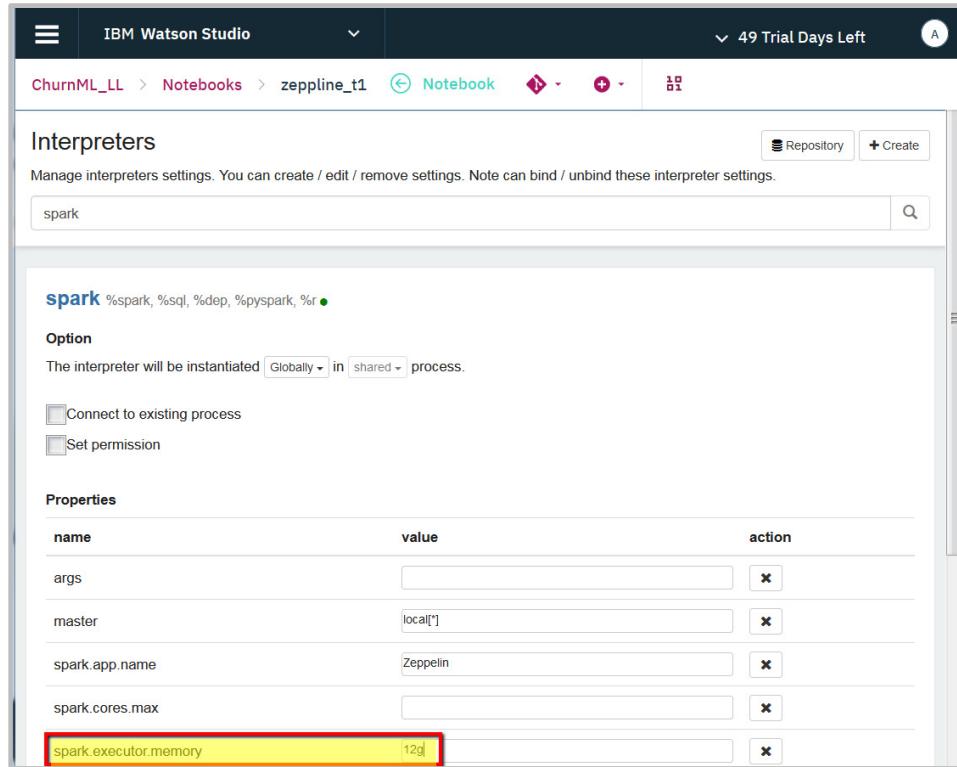
Set Zeppelin interpreter property and increase memory

- Go to the Zeppelin interpreters by clicking on the wheel at the right top of the Zeppelin notebook window.
- Click on the Interpreter link to open the Interpreter settings window.
- Search for spark and select edit on the upper right.
- Set spark.executor.memory to 12g for example. You might have to try increasing this as much possible to test.

# Best Practices

## ▪ Performance Tuning – Tune Zeppelin notebook

Set Zeppelin interpreter property and increase memory



The screenshot shows the 'Interpreters' settings page in IBM Watson Studio. A search bar at the top contains the text 'spark'. Below it, a section titled 'spark' lists supported interpreters: %spark, %sql, %dep, %pyspark, %r. Under the 'Option' section, it says the interpreter will be instantiated 'Globally' in a 'shared' process. There are two unchecked checkboxes: 'Connect to existing process' and 'Set permission'. The 'Properties' section contains a table with the following rows:

name	value	action
args		x
master	local[*]	x
spark.app.name	Zeppelin	x
spark.cores.max		x
spark.executor.memory	12g	x

# Best Practices

- **Performance Tuning – Tune Spark applications**
  - **Spark executor configuration – Change the resources for the Spark application**
    - Stop the pre-created spark context, and then create a new spark context with the proper resource configuration. For example:

```
sc.stop()
from pyspark import SparkConf, SparkContext
conf = (SparkConf()
         .set("spark.cores.max", "15")
         .set("spark.dynamicAllocation.initialExecutors", "3")
         .set("spark.executor.cores", "5")
         .set("spark.executor.memory", "6g"))
sc=SparkContext(conf=conf)
```

## Best Practices

- **Performance Tuning – Tune Spark applications**
  - **Spark executor configuration – Change the resources for the Spark application**
    - Verify the new settings by running the following command in a cell using the new Spark context

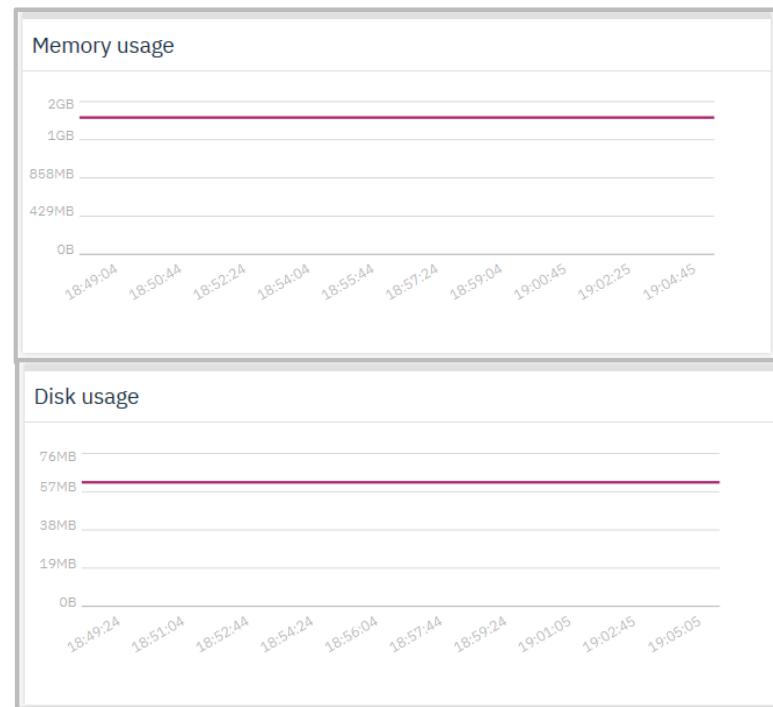
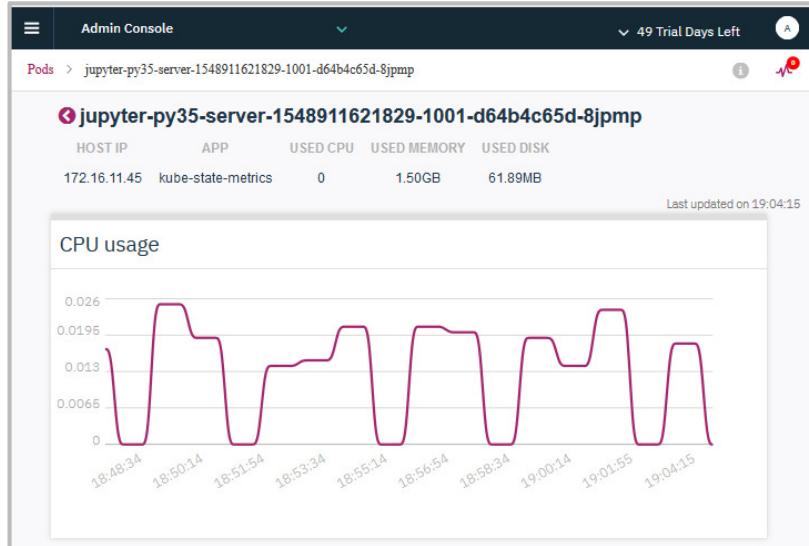
```
for item in sorted(sc._conf.getAll()): print(item)
```

# Best Practices

## ▪ Performance Tuning – Tune Spark applications

### – Monitoring

- You can monitor the per-pod CPU/memory/disk usage from the dashboard in the **Admin Console**.

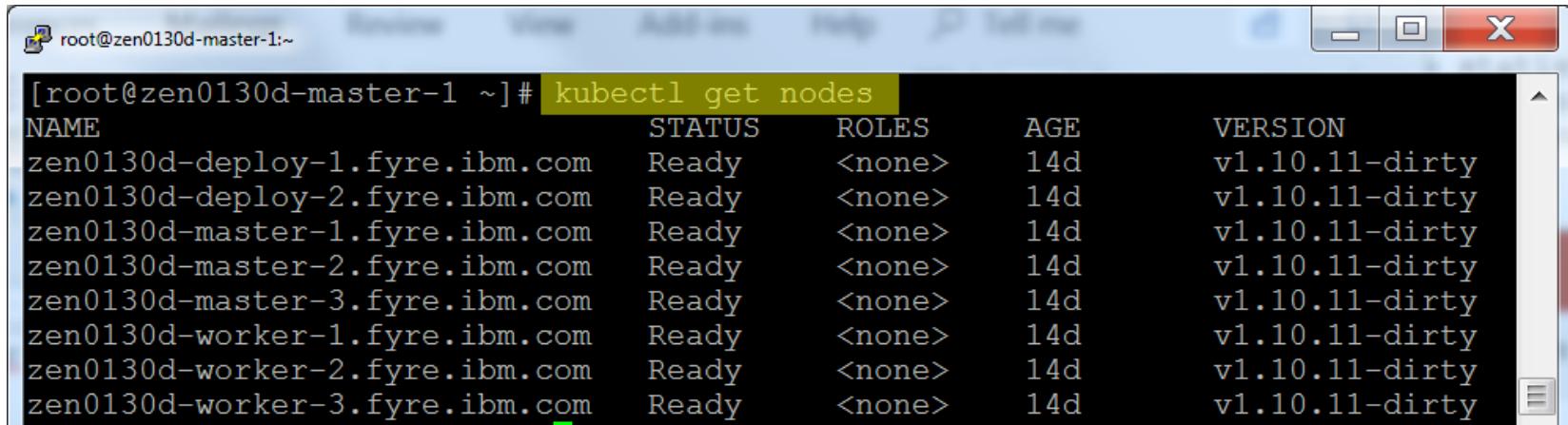


# Watson Studio Local Administration Health Check



## Health Check

- Services need to be up and running on all cluster nodes
  - docker, kubelet, glusterd (ex. # systemctl enable docker; systemctl start docker)
- Verify gluster peer status (make sure communication among cluster nodes is working)
  - # gluster peer status
- Verify all cluster nodes status
  - # kubectl get nodes (FYI. Cluster has issues If any cluster node status is not shown “Ready”)



```
[root@zen0130d-master-1 ~]# kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
zen0130d-deploy-1.fyre.ibm.com   Ready    <none>    14d   v1.10.11-dirty
zen0130d-deploy-2.fyre.ibm.com   Ready    <none>    14d   v1.10.11-dirty
zen0130d-master-1.fyre.ibm.com   Ready    <none>    14d   v1.10.11-dirty
zen0130d-master-2.fyre.ibm.com   Ready    <none>    14d   v1.10.11-dirty
zen0130d-master-3.fyre.ibm.com   Ready    <none>    14d   v1.10.11-dirty
zen0130d-worker-1.fyre.ibm.com  Ready    <none>    14d   v1.10.11-dirty
zen0130d-worker-2.fyre.ibm.com  Ready    <none>    14d   v1.10.11-dirty
zen0130d-worker-3.fyre.ibm.com  Ready    <none>    14d   v1.10.11-dirty
```

# Health Check

- Verify if Pods are all up and running
  - Verify from Admin Console GUI
  - OR via command line: # kubectl get [pods | po] --all-namespaces

**Tip:** If a pod is not “Running”, you can:

- Delete the Pod (it will auto create a new instance).

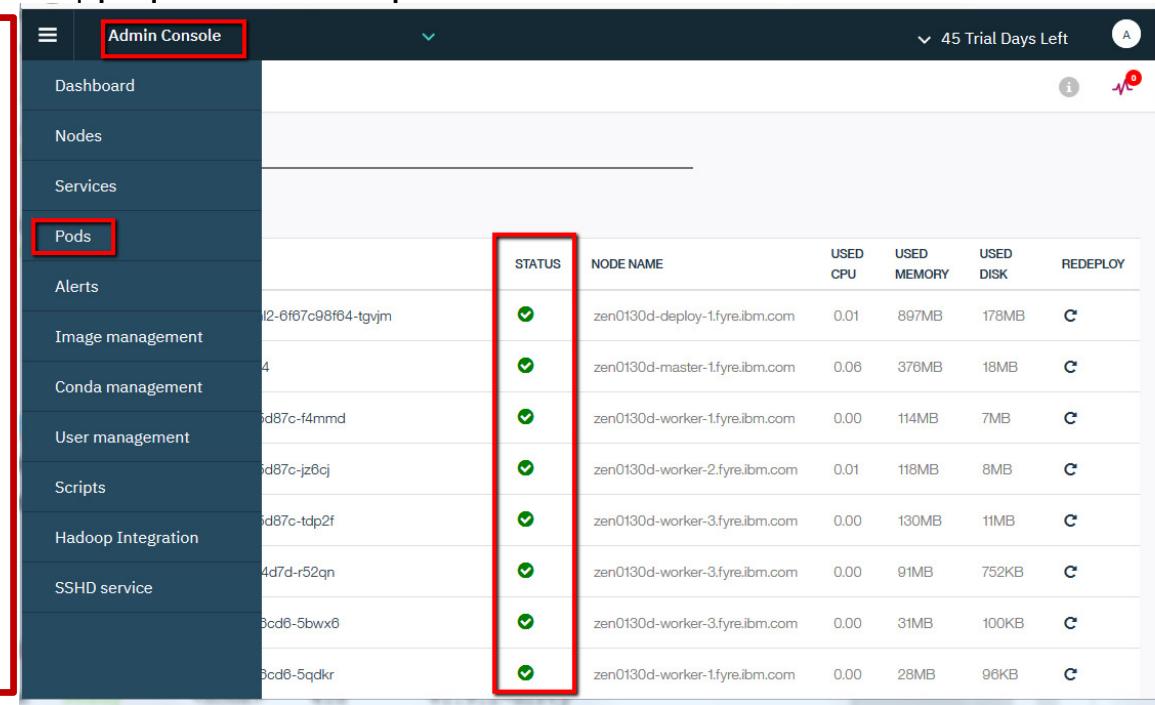
Ex. kubectl delete po [--force] -n <namespace>  
<pod>

- Run describe command to review more detail

Ex. kubectl describe po <pod> -n <namespace>

- Review Pod logs to investigate errors.

Ex. kubectl logs <pod> -n <namespace>



STATUS	NODE NAME	USED CPU	USED MEMORY	USED DISK	REDEPLOY
Running	zen0130d-deploy-1.fyre.ibm.com	0.01	897MB	178MB	C
Running	zen0130d-master-1.fyre.ibm.com	0.06	376MB	18MB	C
Running	zen0130d-worker-1.fyre.ibm.com	0.00	114MB	7MB	C
Running	zen0130d-worker-2.fyre.ibm.com	0.01	118MB	8MB	C
Running	zen0130d-worker-3.fyre.ibm.com	0.00	130MB	11MB	C
Running	zen0130d-worker-3.fyre.ibm.com	0.00	91MB	752KB	C
Running	zen0130d-worker-3.fyre.ibm.com	0.00	31MB	100KB	C
Running	zen0130d-worker-1.fyre.ibm.com	0.00	28MB	96KB	C
Running	zen0130d-worker-1.fyre.ibm.com	0.00	28MB	96KB	C

# Health Check

- Verify if Pods are all up and running
  - Verify from Admin Console GUI
  - OR via command line: # kubectl get [pods | po] --all-namespaces

**Tip:** If a pod is not “Running”, you can:

- Delete the Pod (it will auto create a new instance).

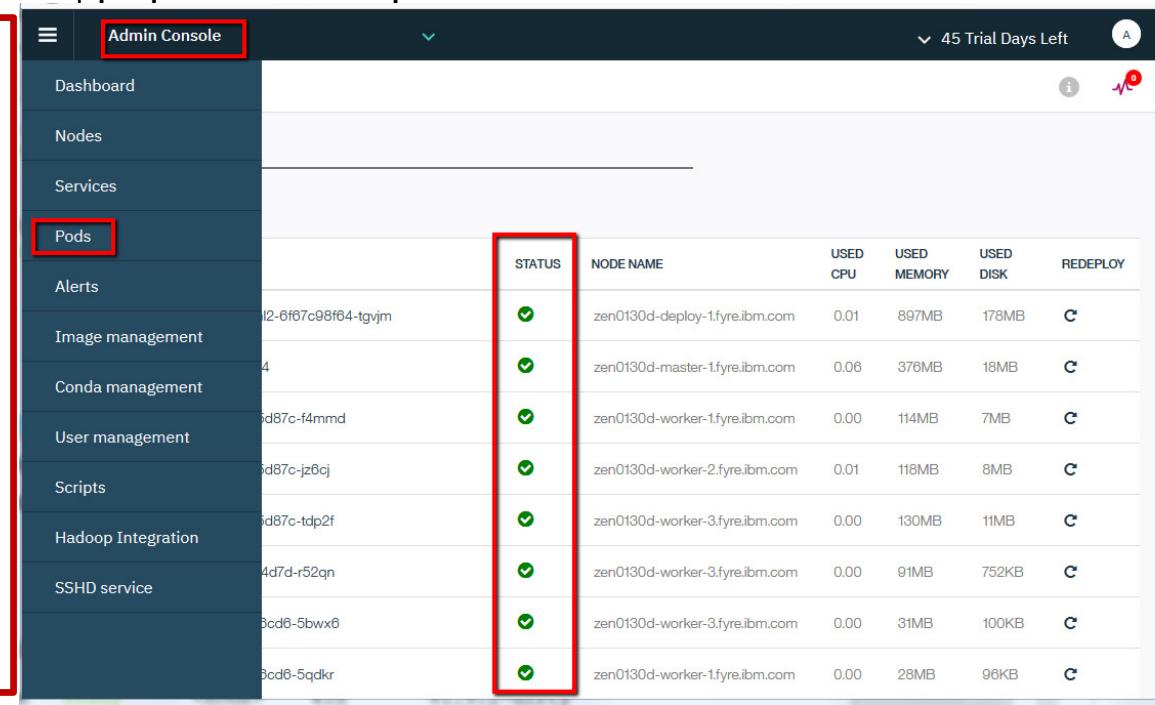
Ex. kubectl delete po [--force] -n <namespace>  
<pod>

- Run describe command to review more detail

Ex. kubectl describe po <pod> -n <namespace>

- Review Pod logs to investigate errors.

Ex. kubectl logs <pod> -n <namespace>



STATUS	NODE NAME	USED CPU	USED MEMORY	USED DISK	REDEPLOY
Running	zen0130d-deploy-1.fyre.ibm.com	0.01	897MB	178MB	C
Running	zen0130d-master-1.fyre.ibm.com	0.06	376MB	18MB	C
Running	zen0130d-worker-1.fyre.ibm.com	0.00	114MB	7MB	C
Running	zen0130d-worker-2.fyre.ibm.com	0.01	118MB	8MB	C
Running	zen0130d-worker-3.fyre.ibm.com	0.00	130MB	11MB	C
Running	zen0130d-worker-3.fyre.ibm.com	0.00	91MB	752KB	C
Running	zen0130d-worker-3.fyre.ibm.com	0.00	31MB	100KB	C
Running	zen0130d-worker-1.fyre.ibm.com	0.00	28MB	96KB	C
Running	zen0130d-worker-1.fyre.ibm.com	0.00	28MB	96KB	C

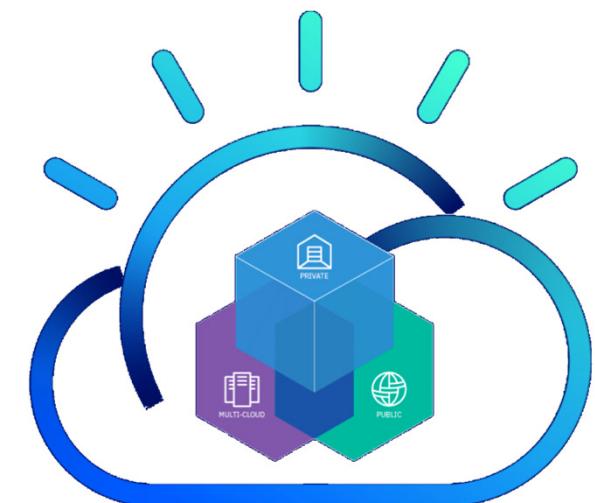
# Health Check

## ▪ Admin utility scripts

- Automatically troubleshoot and repair common problems in your Watson Studio Local cluster:
  - `/wdp/utils/cluster_repair_utility.sh [-h]`
- Troubleshoot and print out results in a zipped tar file:
  - `/wdp/utils/admin-utils.sh --user <username> --password --port <port> --key-pair <ssh_key_pair_file>`

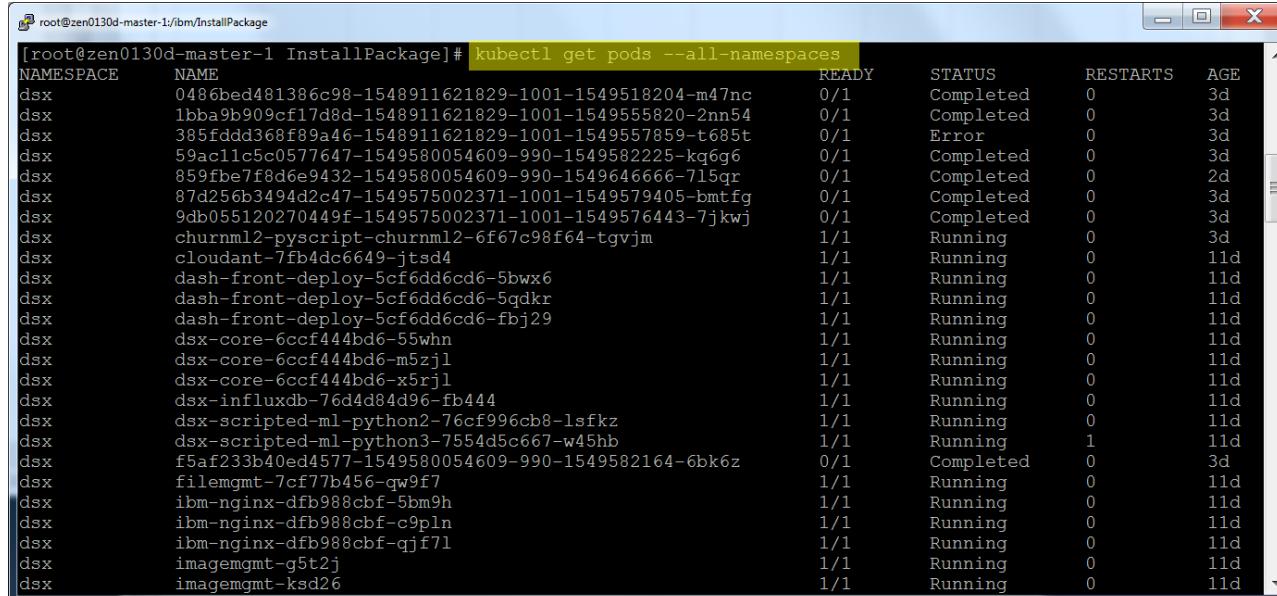
`Ex. /wdp/utils/admin-utils.sh --user root --password --key-pair /root/.ssh/id_rsa`

# Watson Studio Local Administration Troubleshoots



# Access Pods

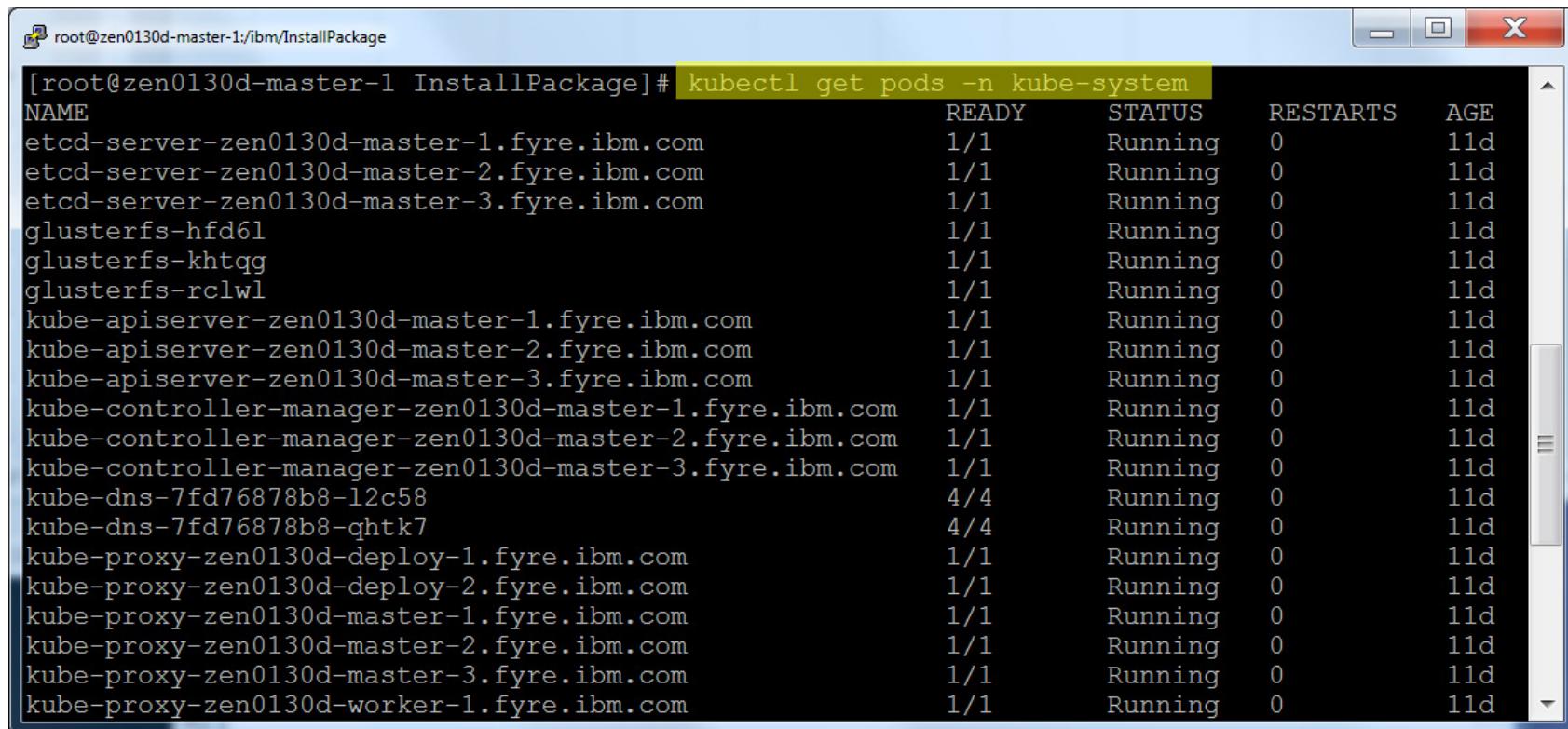
- List all Pods in the cluster
  - \$ kubectl get pods --all-namespaces



NAME	READY	STATUS	RESTARTS	AGE
0486bed481386c98-1548911621829-1001-1549518204-m47nc	0/1	Completed	0	3d
1bba9b909cf17d8d-1548911621829-1001-1549555820-2nn54	0/1	Completed	0	3d
385fddd368f89a46-1548911621829-1001-1549557859-t685t	0/1	Error	0	3d
59ac11c5c0577647-1549580054609-990-1549582225-kq6g6	0/1	Completed	0	3d
859fbe7f8d6e9432-1549580054609-990-1549646666-7l5qr	0/1	Completed	0	2d
87d256b3494d2c47-1549575002371-1001-1549579405-bmtfg	0/1	Completed	0	3d
9db055120270449f-1549575002371-1001-1549576443-7jkwj	0/1	Completed	0	3d
churnml2-pyscript-churnml2-6f67c98f64-tgvjm	1/1	Running	0	3d
cloudant-7fb4dc6649-jtsd4	1/1	Running	0	11d
dash-front-deploy-5cf6dd6cd6-5bwx6	1/1	Running	0	11d
dash-front-deploy-5cf6dd6cd6-5qdkr	1/1	Running	0	11d
dash-front-deploy-5cf6dd6cd6-fbj29	1/1	Running	0	11d
dsx-core-6ccf444bd6-55whn	1/1	Running	0	11d
dsx-core-6ccf444bd6-m5zjl	1/1	Running	0	11d
dsx-core-6ccf444bd6-x5rjl	1/1	Running	0	11d
dsx-influxdb-76d4d84d96-fb444	1/1	Running	0	11d
dsx-scripted-ml-python2-76cf996cb8-lsfkz	1/1	Running	0	11d
dsx-scripted-ml-python3-7554d5c667-w45hb	1/1	Running	1	11d
f5af233b40ed4577-1549580054609-990-1549582164-6bk6z	0/1	Completed	0	3d
filemgmt-7cf77b456-qw9f7	1/1	Running	0	11d
ibm-nginx-dfb988cbf-5bm9h	1/1	Running	0	11d
ibm-nginx-dfb988cbf-c9pln	1/1	Running	0	11d
ibm-nginx-dfb988cbf-qjf71	1/1	Running	0	11d
imagemgmt-g5t2j	1/1	Running	0	11d
imagemgmt-ksd26	1/1	Running	0	11d

## Access Pods

- List all Pods in the cluster based on namespace
  - \$ kubectl get pods -n kube-system



```
[root@zen0130d-master-1 InstallPackage]# kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS   AGE
etcd-server-zen0130d-master-1.fyre.ibm.com   1/1     Running   0          11d
etcd-server-zen0130d-master-2.fyre.ibm.com   1/1     Running   0          11d
etcd-server-zen0130d-master-3.fyre.ibm.com   1/1     Running   0          11d
glusterfs-hfd61                         1/1     Running   0          11d
glusterfs-khtqg                         1/1     Running   0          11d
glusterfs-rclwl                         1/1     Running   0          11d
kube-apiserver-zen0130d-master-1.fyre.ibm.com 1/1     Running   0          11d
kube-apiserver-zen0130d-master-2.fyre.ibm.com 1/1     Running   0          11d
kube-apiserver-zen0130d-master-3.fyre.ibm.com 1/1     Running   0          11d
kube-controller-manager-zen0130d-master-1.fyre.ibm.com 1/1     Running   0          11d
kube-controller-manager-zen0130d-master-2.fyre.ibm.com 1/1     Running   0          11d
kube-controller-manager-zen0130d-master-3.fyre.ibm.com 1/1     Running   0          11d
kube-dns-7fd76878b8-12c58                  4/4     Running   0          11d
kube-dns-7fd76878b8-qhtk7                  4/4     Running   0          11d
kube-proxy-zen0130d-deploy-1.fyre.ibm.com   1/1     Running   0          11d
kube-proxy-zen0130d-deploy-2.fyre.ibm.com   1/1     Running   0          11d
kube-proxy-zen0130d-master-1.fyre.ibm.com   1/1     Running   0          11d
kube-proxy-zen0130d-master-2.fyre.ibm.com   1/1     Running   0          11d
kube-proxy-zen0130d-master-3.fyre.ibm.com   1/1     Running   0          11d
kube-proxy-zen0130d-worker-1.fyre.ibm.com   1/1     Running   0          11d
```

## Access Pods

- Log in to a Pod with shell

- \$ kubectl exec -it -n <namespace> <pod\_name> [bash|sh]

**Example: \$ kubectl exec -it -n kube-system glusterfs-hfd6l bash**

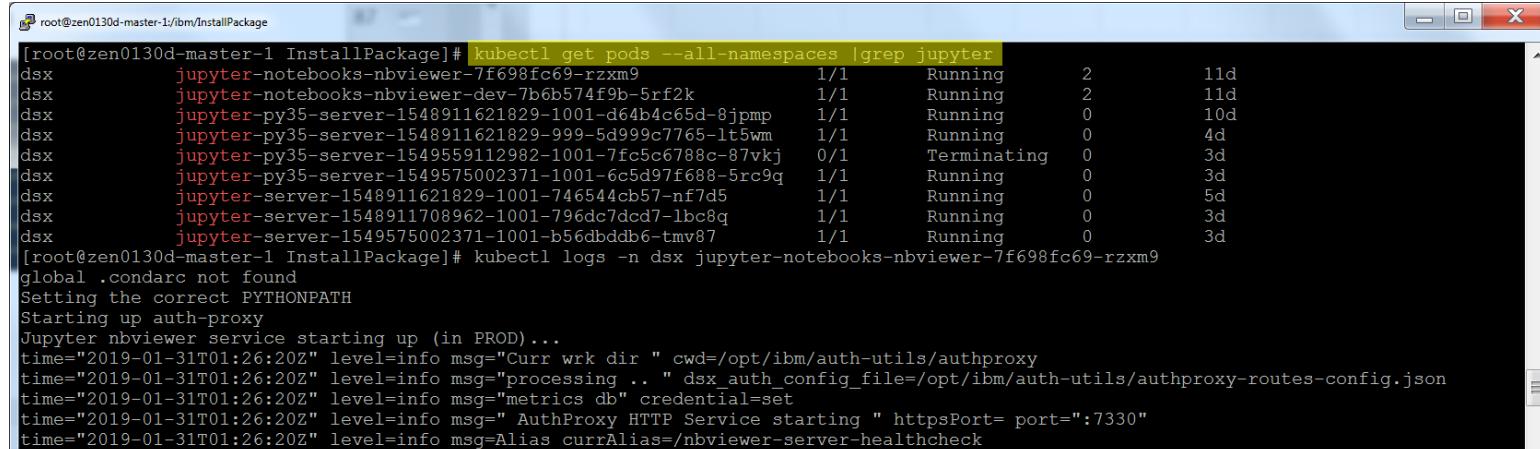
- Access Pod's logs

- \$ kubectl logs -n <namespace> <pod-name>

- Restart a Pod

- \$ kubectl delete pod -n <namespace> <pod\_name>

**Example: \$ kubectl delete pod -n dsx jupyter-notebooks-nbviewer-7f698fc69-rzxm9**



The screenshot shows a terminal window with two command outputs. The top part shows a list of pods across namespaces:

```
[root@zen0130d-master-1 InstallPackage]# kubectl get pods --all-namespaces |grep jupyter
dsx      jupyter-notebooks-nbviewer-7f698fc69-rzxm9      1/1     Running   2      11d
dsx      jupyter-notebooks-nbviewer-dev-7b6b574f9b-5rf2k    1/1     Running   2      11d
dsx      jupyter-py35-server-1548911621829-1001-d64b4c65d-8jmpm  1/1     Running   0      10d
dsx      jupyter-py35-server-1548911621829-9999-5d999c7765-1t5wm  1/1     Running   0      4d
dsx      jupyter-py35-server-1549559112982-1001-7fc5c6788c-87vkj  0/1     Terminating 0      3d
dsx      jupyter-py35-server-1549575002371-1001-6c5d97f688-5rc9q  1/1     Running   0      3d
dsx      jupyter-server-1548911621829-1001-746544cb57-nf7d5    1/1     Running   0      5d
dsx      jupyter-server-1548911708962-1001-796dc7ddc7-lbc8q    1/1     Running   0      3d
dsx      jupyter-server-1549575002371-1001-b56dbddb6-tmv87    1/1     Running   0      3d
```

The bottom part shows the logs for the 'jupyter-notebooks-nbviewer-7f698fc69-rzxm9' pod:

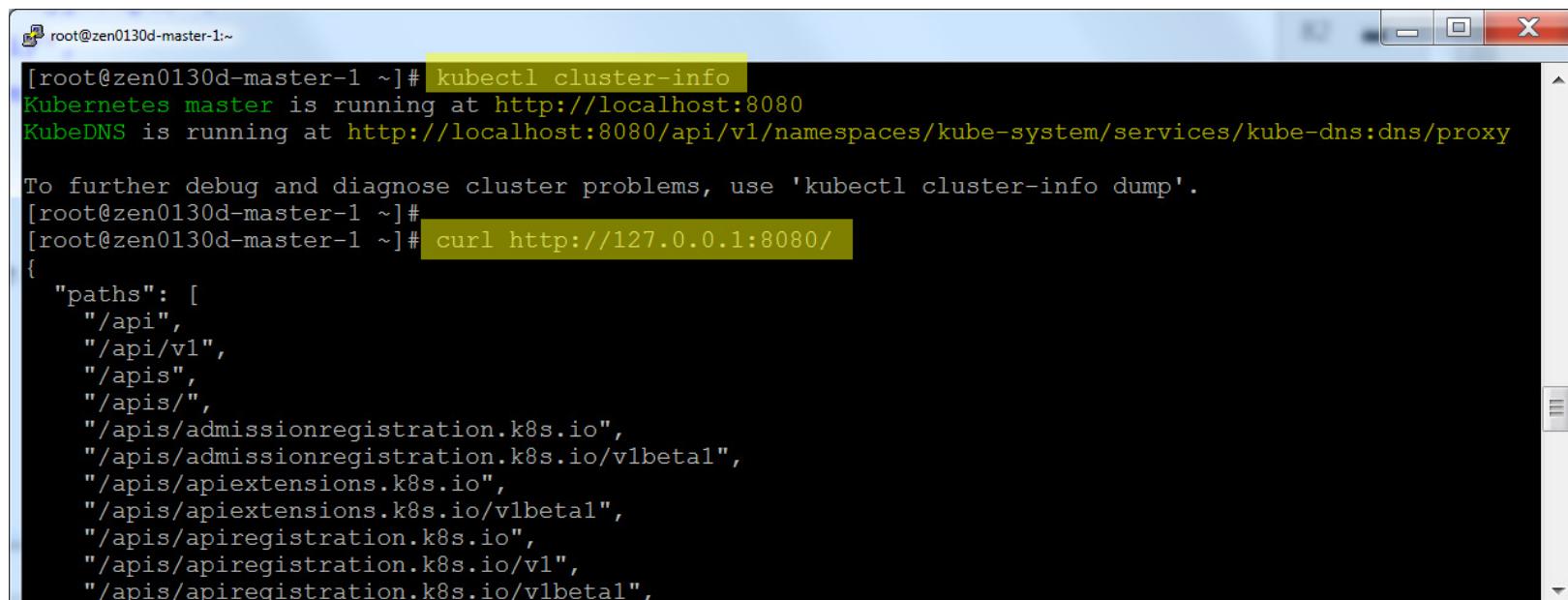
```
[root@zen0130d-master-1 InstallPackage]# kubectl logs -n dsx jupyter-notebooks-nbviewer-7f698fc69-rzxm9
global .condarc not found
Setting the correct PYTHONPATH
Starting up auth-proxy
Jupyter nbviewer service starting up (in PROD)...
time="2019-01-31T01:26:20Z" level=info msg="Curr wrk dir " cwd=/opt/ibm/auth-utils/authproxy
time="2019-01-31T01:26:20Z" level=info msg="processing .. " dsx_auth_config_file=/opt/ibm/auth-utils/authproxy-routes-config.json
time="2019-01-31T01:26:20Z" level=info msg="metrics db" credential=set
time="2019-01-31T01:26:20Z" level=info msg=" AuthProxy HTTP Service starting " httpsPort= port=":7330"
time="2019-01-31T01:26:20Z" level=info msg=Alias currAlias=/nbviewer-server-healthcheck
```

# Docker 101

- Docker image vs container
  - A docker container is the runtime instance of an image
- Useful Docker commands
  - \$ docker images → list all images that are locally stored with the Docker
  - \$ docker rmi <image\_name> → delete an image from the local image store
  - \$ docker kill <container> → kill a running container
  - \$ docker ps → list all “running” docker containers
  - \$ docker ps -a → list all containers (running and not running)
  - \$ docker inspect <container\_id> → list detailed info about a docker container
  - \$ docker logs <container\_id> → show logs of a given container
  - \$ docker build -t <image\_name> . → build an image from the Dockerfile in the current directory and tag the image
  - \$ docker [start | stop | rm] <container> → start: Launch a container previously stopped.
  - \$ docker run <image> → create a new container of an image, and execute the container.
  - \$ docker exec -it <container> [command] → ex. docker exec -it jupyter-notebooks-nbviewer-

# Cluster information

- Show cluster information
  - \$ kubectl cluster-info
- List all API endpoint
  - \$ curl http://localhost:8080/



```
root@zen0130d-master-1:~ [root@zen0130d-master-1 ~]# kubectl cluster-info
Kubernetes master is running at http://localhost:8080
KubeDNS is running at http://localhost:8080/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
[root@zen0130d-master-1 ~]#
[root@zen0130d-master-1 ~]# curl http://127.0.0.1:8080/
{
  "paths": [
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
    "/apis/admissionregistration.k8s.io/v1beta1",
    "/apis/apiextensions.k8s.io",
    "/apis/apiextensions.k8s.io/v1beta1",
    "/apis/apiregistration.k8s.io",
    "/apis/apiregistration.k8s.io/v1",
    "/apis/apiregistration.k8s.io/v1beta1",
```

## Cluster / Pod information

- List all namespaces in the cluster
  - \$ kubectl get namespaces
  
- Look at a Pod's details
  - \$ kubectl describe pod <pod\_name> -n <namespace>

```
root@zen0130d-master-1:~]# kubectl get namespaces
NAME        STATUS   AGE
default     Active  11d
dsx         Active  11d
kube-public Active  11d
kube-system Active  11d
sysibm-adm  Active  11d
[root@zen0130d-master-1 ~]#
```

```
root@zen0130d-master-1:~]# kubectl get pods --all-namespaces | grep spark
dsx      spark-history-server-8db797c7c-6vhpc    1/1    Running   0      11d
dsx      spark-master-855585f76-vplz6           1/1    Running   0      11d
dsx      spark-master221-6f99c4fbc7-gnqf4       1/1    Running   0      11d
dsx      spark-worker-bqn2l                      1/1    Running   0      11d
dsx      spark-worker-g6nv2                      1/1    Running   0      11d
dsx      spark-worker-gv5zq                      1/1    Running   0      11d
dsx      spark-worker221-csgdj                   1/1    Running   0      11d
dsx      spark-worker221-fwtm8                   1/1    Running   0      11d
dsx      spark-worker221-r8b5s                   1/1    Running   0      11d
dsx      sparkaas-api-deploy-644f997f-51vnj    1/1    Running   0      11d
dsx      sparkaas-api-deploy-644f997f-1hs8r    1/1    Running   0      11d
dsx      sparkaas-api-deploy-644f997f-vngp4    1/1    Running   0      11d
dsx      sparkgxm-764bcf8f8-thkp2              3/3    Running   0      11d
[root@zen0130d-master-1 ~]#
[root@zen0130d-master-1 ~]# kubectl describe pod spark-master-855585f76-vplz6 -n dsx
Name:           spark-master-855585f76-vplz6
Namespace:      dsx
Node:           zen0130d-worker-2.fyre.ibm.com/172.16.11.42
Start Time:     Wed, 30 Jan 2019 17:25:49 -0800
Labels:          app=spark
                  chart=ibm-dsx-prod
                  component=spark-master
                  heritage=tiller
                  pod-template-hash=411141932
                  release=dsx
                  run=spark-master-deployment-pod
Annotations:    productID=IBMWatsonStudio_123_perpetual_00000
                  productName=IBM Watson Studio
```

## Troubleshooting

- Watson Studio Local provides various methods, scripts, and support for troubleshooting your cluster
- The areas we are going to cover include:
  - Troubleshoot common problems
  - Troubleshoot your system performance
  - Troubleshoot your system with administration utilities

# Troubleshoot Common Problems

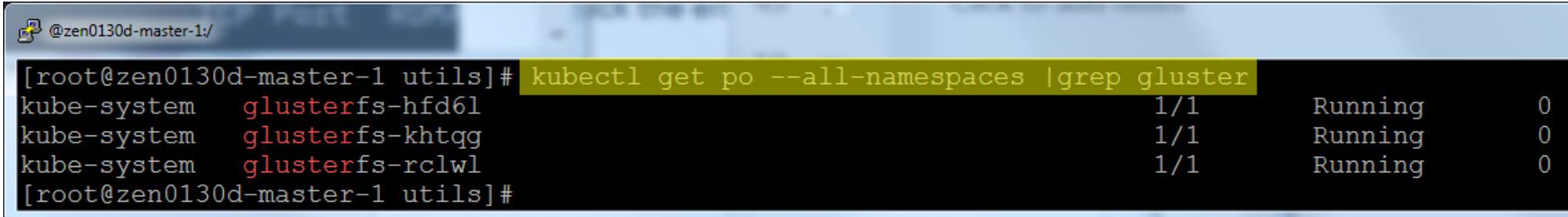
- **Symptom:**

- Watson Studio Local does not start, pods stuck creating or crashing, due to glusterfs in read-only or offline

- **Diagnostic Procedure:**

- 1. Verify glusters bricks are online by using the following command:

```
kubectl get po --all-namespaces | grep gluster
```



```
[root@zen0130d-master-1:/] [root@zen0130d-master-1 utils]# kubectl get po --all-namespaces |grep gluster
kube-system    glusterfs-hfd61                           1/1        Running      0
kube-system    glusterfs-khtqg                           1/1        Running      0
kube-system    glusterfs-rclwl                           1/1        Running      0
[root@zen0130d-master-1 utils]#
```

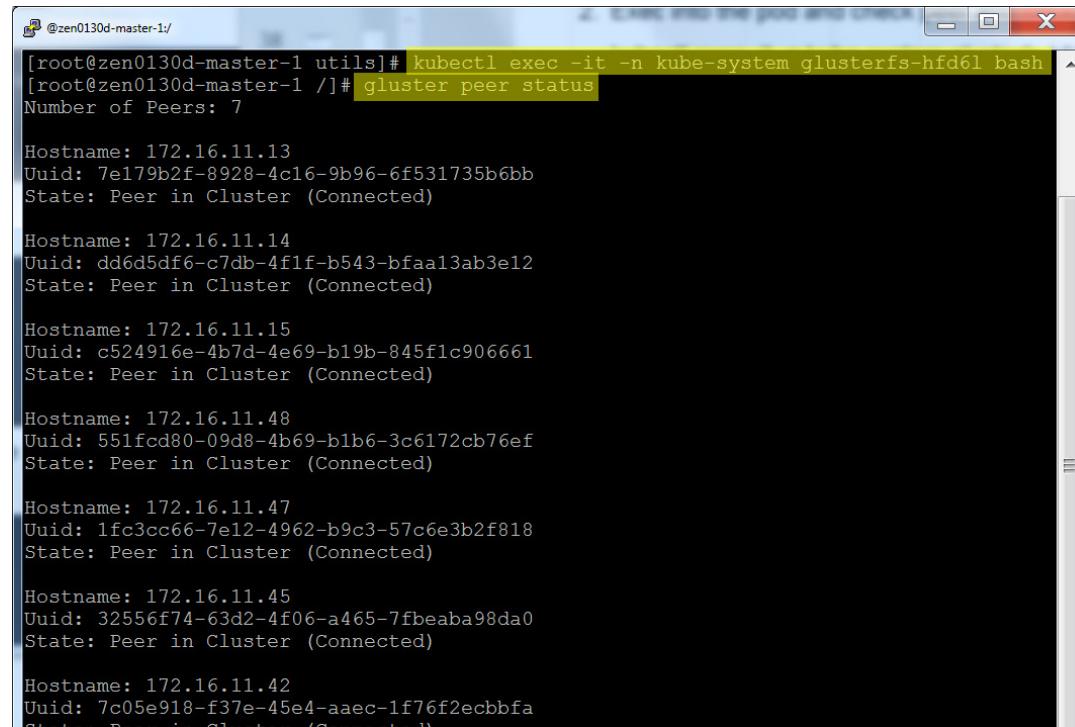
# Troubleshoot Common Problems

- **Diagnostic Procedure (cont):**

2. Exec into the pod and check peer and brick status using the following command:

```
kubectl exec -it -n kube-system glusterfs-hfd6l bash
```

```
gluster peer status
```



```
[root@zen0130d-master-1 utils]# kubectl exec -it -n kube-system glusterfs-hfd6l bash
[root@zen0130d-master-1 /]# gluster peer status
Number of Peers: 7

Hostname: 172.16.11.13
Uuid: 7e179b2f-8928-4c16-9b96-6f531735b6bb
State: Peer in Cluster (Connected)

Hostname: 172.16.11.14
Uuid: dd6d5df6-c7db-4f1f-b543-bfaa13ab3e12
State: Peer in Cluster (Connected)

Hostname: 172.16.11.15
Uuid: c524916e-4b7d-4e69-b19b-845f1c906661
State: Peer in Cluster (Connected)

Hostname: 172.16.11.48
Uuid: 551fcfd80-09d8-4b69-b1b6-3c6172cb76ef
State: Peer in Cluster (Connected)

Hostname: 172.16.11.47
Uuid: 1fc3cc66-7e12-4962-b9c3-57c6e3b2f818
State: Peer in Cluster (Connected)

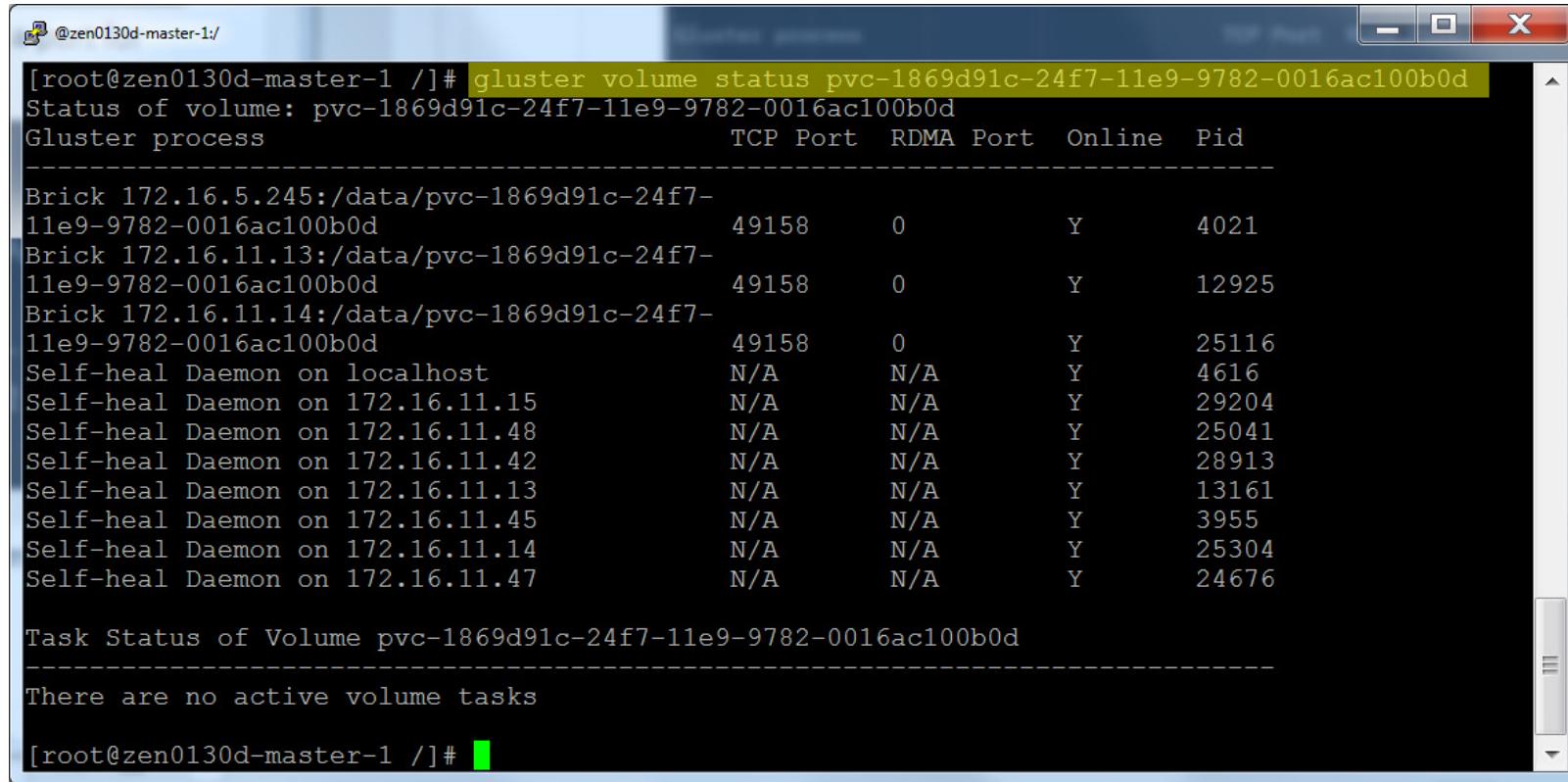
Hostname: 172.16.11.45
Uuid: 32556f74-63d2-4f06-a465-7fbeaba98da0
State: Peer in Cluster (Connected)

Hostname: 172.16.11.42
Uuid: 7c05e918-f37e-45e4-aaec-1f76f2ecbbfa
State: Peer in Cluster (Connected)
```

# Troubleshoot Common Problems

- **Diagnostic Procedure (cont):**

3. Enter gluster volume status. Verify that you see 3 bricks and show a port for the brick that is online:



```
[root@zen0130d-master-1 /]# gluster volume status pvc-1869d91c-24f7-11e9-9782-0016ac100b0d
Status of volume: pvc-1869d91c-24f7-11e9-9782-0016ac100b0d
Gluster process          TCP Port  RDMA Port  Online  Pid
-----
Brick 172.16.5.245:/data/pvc-1869d91c-24f7-11e9-9782-0016ac100b0d      49158     0       Y      4021
Brick 172.16.11.13:/data/pvc-1869d91c-24f7-11e9-9782-0016ac100b0d      49158     0       Y      12925
Brick 172.16.11.14:/data/pvc-1869d91c-24f7-11e9-9782-0016ac100b0d      49158     0       Y      25116
Self-heal Daemon on localhost           N/A      N/A       Y      4616
Self-heal Daemon on 172.16.11.15      N/A      N/A       Y      29204
Self-heal Daemon on 172.16.11.48      N/A      N/A       Y      25041
Self-heal Daemon on 172.16.11.42      N/A      N/A       Y      28913
Self-heal Daemon on 172.16.11.13      N/A      N/A       Y      13161
Self-heal Daemon on 172.16.11.45      N/A      N/A       Y      3955
Self-heal Daemon on 172.16.11.14      N/A      N/A       Y      25304
Self-heal Daemon on 172.16.11.47      N/A      N/A       Y      24676

Task Status of Volume pvc-1869d91c-24f7-11e9-9782-0016ac100b0d
-----
There are no active volume tasks

[root@zen0130d-master-1 /]#
```

## Troubleshoot Common Problems

- **Diagnostic Procedure (cont):**

4. If you do not, then run the following to attempt to connect the volumes that are not online:

```
gluster volume stop <volume name>
```

5. Select Y to stop the volume

6. Enter gluster volume start <volume name>

7. Verify with gluster volume status, correct for all the volumes and then WSL should recover

# Troubleshoot Common Problems

- **Symptom:**

- Kubectl get no returns the following ‘The connection to the server localhost:8080 was refused – did you specify the right host or port?’

- **Diagnostic Procedure:**

1. Verify that at least two master nodes are online
2. Check the status of docker:
3. Check the status of kubelet:
4. Check that the kube-apiserver is running: docker ps | grep kube-apiserver
5. If not shown: docker ps -a | grep kube-apiserver, then check the logs: docker logs <api server container>
6. If the logs look fine, check to verify that there is adequate space in the installer partition: df -lh
7. Check logs for etcd container, and verify that you don't see messages about the time being out of sync

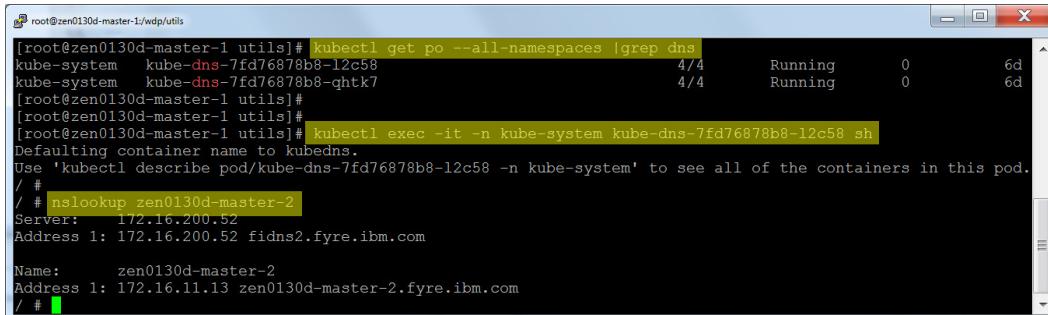
# Troubleshoot Common Problems

- **Symptom:**

- ibm-nginx pods CrashLoopBack/Error

- **Diagnostic Procedure:**

1. Check logs for kubedns.kube-system.svc.cluster.local or usermgmt-svc.dsx.svc.cluster.local
2. Check logs of kubedns pods / restart kubedns
3. On the host, check the kubedns: nslookup <cluster\_node>
4. Exec into dsx-core pod, and check dns resolution in the pod:



```
[root@zen0130d-master-1:~/wdp/utils]# kubectl get po --all-namespaces |grep dns
kube-system kube-dns-7fd76878b8-12c58 4/4 Running 0 6d
kube-system kube-dns-7fd76878b8-qhtk7 4/4 Running 0 6d
[root@zen0130d-master-1 utils]#
[root@zen0130d-master-1 utils]#
[root@zen0130d-master-1 utils]# kubectl exec -it -n kube-system kube-dns-7fd76878b8-12c58 sh
Defaulting container name to kubedns.
Use 'kubectl describe pod/kube-dns-7fd76878b8-12c58 -n kube-system' to see all of the containers in this pod.
/ #
/ # nslookup zen0130d-master-2
Server: 172.16.200.52
Address 1: 172.16.200.52 fidns2.fyre.ibm.com

Name: zen0130d-master-2
Address 1: 172.16.11.13 zen0130d-master-2.fyre.ibm.com
/ #
```

5. If resolving names doesn't work, check /etc/resolv.conf setting; and confirm that outside DNS lookups work: nslookup <hostname on network>

# Troubleshoot Common Problems

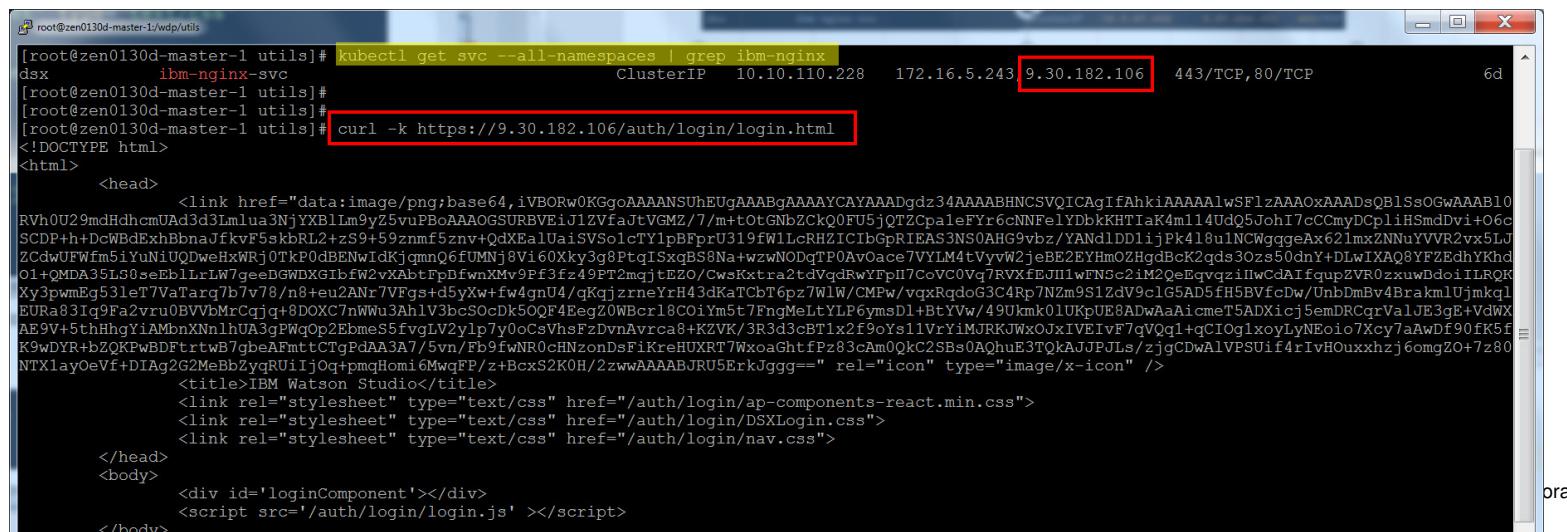
## ▪ Symptom:

- Connecting to WSL from browser fails but ALL pods show as running

## ▪ Diagnostic Procedure:

1. Verify that you can ping the proxy IP or hostname
2. Connect to one of the nodes and attempt to connect to the site internally

## 3. Verify site IP:



```

root@zen0130d-master-1:~# curl -k https://9.30.182.106/auth/login/login.html
<!DOCTYPE html>
<html>
  <head>
    <link href="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABqAAAAACAYAAADgdz34AAAAABHNC SVQICAgI fAhkiAAAAAlwSFlzAAAoxAAADsQB1ssOGwAAAB1oR
    Vh0U29mdHdcmUAd3d3LmLua3NjYXBB1Lm9yZ5vuPBoAAAOGSURBVEiJ1ZVfaJtVGMZ//7/m+t0tGnbZCkQ0FU5jQTZCpaleFYr6cNNFe1YDbkKHTiaK4m14UdQ5Johi7cCCmyDCpliHSmdDvi+O6c
    SCDF+h+DcWBdExhBbnaJfkvF5skbRL2+zS9+59znmf5zvn+QdXEa1UaiSVSo1cTY1pBFprU319fW1LcRHZICIB6PRIEAS3NS0AHG9vbz/YANd1DD1ijPk418u1NCWgggeAx62lmxZNnuYVVR2vx5LJ
    ZCdwUFwMnDkQdWeXwRjOTkP0dBNwldKjgmn6fUMNj8Vi60Xky3g3PtqISxqBS8Na+wzwNODqTP0Av0ace7YLM4tVvv2jeBE2EYHm0ZHgdbck2qds30z50dnY+DlwIXAq8YFZEdhYKhd
    O1+QMda35LS0seEb1LrLw7geeBGwdxG1bfw2vxAbtFpEfwnXMv9PF3fz49PT2mcjtEZo/CwsKxtra2tdVqdRwYpII7CoVc0Vq7RVxfEJH1wFN5c2iM2QeEgvqzilwcd4IfqupZVR0zxuwBdoiiLRQK
    Xy3pwEg53leT7VaTaRq7b7v78/n8+eu2ANr7VFgs+d5yXw+fw4gnU4/qKqjzrneYrH43dKaTCb76pz7wlW/CMPw/vqxRqdoG3C4Rp7Nzm951Zdv9c1G5AD5fH5BVfcDw/UnbdmBv4BrakmlUjmql
    EURa831q9Fa2vru0BVVbMrCqjq+8DOXC7nWwU3Ah1V3bcSOCdk50QF4EegZOBcr1CoIym5t7FnqMeLtYLP6ymsDl+BtYWW/49Ukmk01UKpUE8ADwhaaAicmeT5ADXicj5emDRcqrValJE3g+EvdWX
    AE9V+5thlhqYiAMbnXNnlhuA3gPWqOp2EbmesS5fvqLV2ylp7yooCsVhsFzDvnAvrcab+KZVK/3R3d3cBT1x2f9oys11VrYiMURKJWxOjxIVEIvF7qVqg1+qCIog1xoyLyNEcioi7xcytaAwDf90fk5f
    K9wDYR+bZQKfwBDfttwB7gbeAFmttCTqPdAA3A7/5vn/Fb9fwNR0cHNzonDsFIKreHUXRT7WxoahGtfPz83cAm0QkC2SBs0Aqhnb3TqKAJJPLs/zjgCDwAlVPSUif4rIvHoUxxhzj6omgZO+7z80
    NTIX1ayOeVf+DIAg2G2MeBbZyqRUiIjoq+pmqHomie6MwqFP/z+BcxS2K0H/2zwAAAABJRU5ErkJgg==" rel="icon" type="image/x-icon" />
    <title>IBM Watson Studio</title>
    <link rel="stylesheet" type="text/css" href="/auth/login/ap-components-react.min.css">
    <link rel="stylesheet" type="text/css" href="/auth/login/DSXLLogin.css">
    <link rel="stylesheet" type="text/css" href="/auth/login/nav.css">
  </head>
  <body>
    <div id='loginComponent'></div>
    <script src='/auth/login/login.js' ></script>
  </body>

```

## Troubleshoot Common Problems

- **Diagnostic Procedure (cont):**

If the page comes up as previous screenshot, Watson Studio Local works internally.

4. Check if firewalls are enabled:

`systemctl status firewalld`

`Systemctl status iptables`

Verify that there are no firewalls, proxies, or blocked 443 port from the system.

## Troubleshoot System Performance

- If performance slows for computing jobs, or stalls to the point where no new projects or assets can be created anymore, reserve more CPU and memory for your runtime in the Environments page
- Additionally, you can stop the runtime environment for idle notebooks to free up more resource

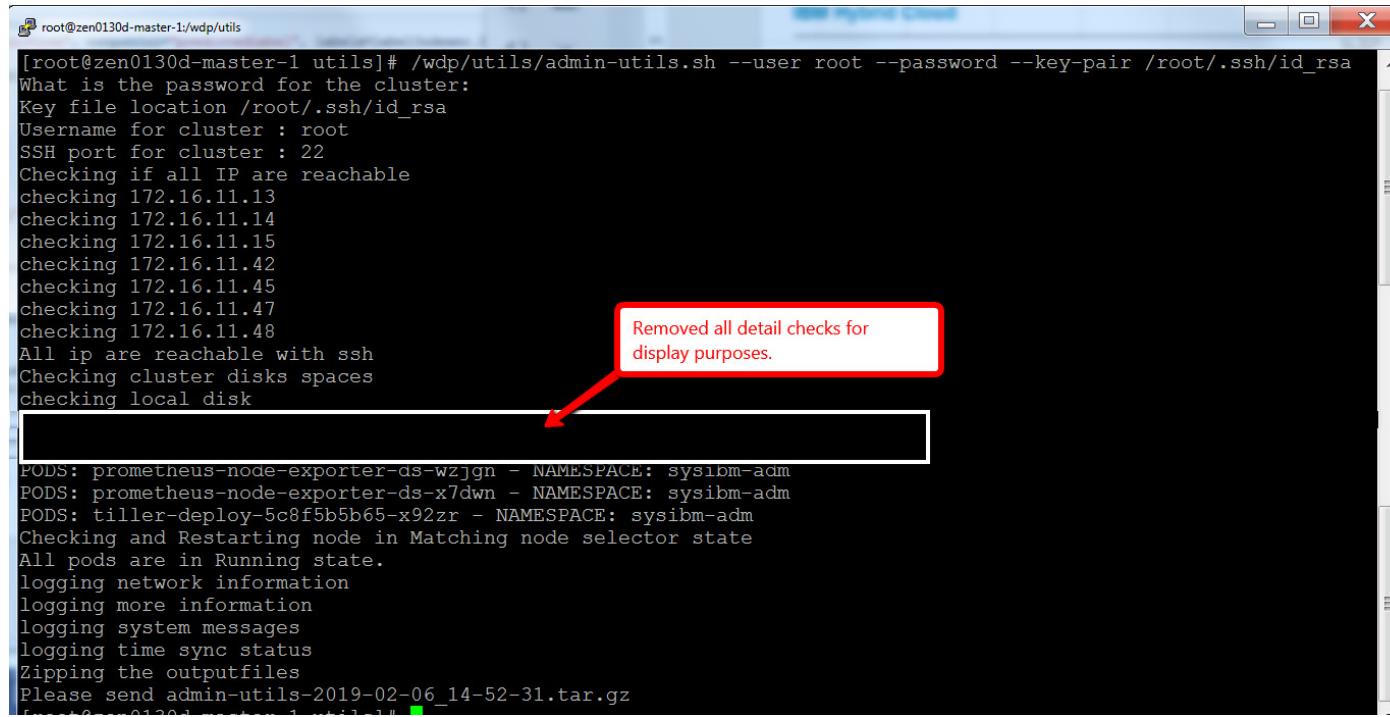
```
In [10]: %spark cleanup
```

```
In [ ]: %%javascript  
Jupyter.notebook.session.delete();
```

# Troubleshoot using Administration Utilities

- To automatically troubleshoot your entire WSL cluster, run the `/wdp/utils/admin-utils.sh` script by entering the following command:

```
/wdp/utils/admin-utils.sh --user <userid> --password <--port port_number> --key-pair <ssh_key_file>
```



```
[root@zen0130d-master-1 wdp/utils]# /wdp/utils/admin-utils.sh --user root --password --key-pair /root/.ssh/id_rsa
What is the password for the cluster:
Key file location /root/.ssh/id_rsa
Username for cluster : root
SSH port for cluster : 22
Checking if all IP are reachable
checking 172.16.11.13
checking 172.16.11.14
checking 172.16.11.15
checking 172.16.11.42
checking 172.16.11.45
checking 172.16.11.47
checking 172.16.11.48
All ip are reachable with ssh
Checking cluster disks spaces
checking local disk
PODS: prometheus-node-exporter-ds-wzjgn - NAMESPACE: sysibm-adm
PODS: prometheus-node-exporter-ds-x7dwn - NAMESPACE: sysibm-adm
PODS: tiller-deploy-5c8f5b5b65-x92zr - NAMESPACE: sysibm-adm
Checking and Restarting node in Matching node selector state
All pods are in Running state.
logging network information
logging more information
logging system messages
logging time sync status
Zipping the outputfiles
Please send admin-utils-2019-02-06_14-52-31.tar.gz
```

Removed all detail checks for display purposes.

## Troubleshoot using Administration Utilities

- The command performs the following tasks:
  - Checks whether all nodes are up and accessible with SSH
  - Checks free disk space on all nodes
  - Checks whether the Docker and Kubelet are running
  - Checks whether Gluster is installed and the volumes are up
- If all of the checks pass, then the script automatically logs everything that is not in the state ‘Ready’ or ‘Running’ into the following compressed file:
  - `/wdp/utils/admin-utils-<timestamp>.tar.gz`
- You can then send this file to IBM Support to diagnose or you can take a look for yourself