



Created : 29 November 2017

Last Updated :

Date Submitted : 29 November 2017

Version Number : 1.0

Author : Logan Data Team

Recommendation Report – Boston University Cloud Data Integration Project

Contents

[1 INTRODUCTION & BACKGROUND 2](#_Toc500162486)

[2 OBJECTIVES 2](#_Toc500162487)

[3 RECOMMENDATIONS 2](#_Toc500162488)

[3.1 SNAPLOGIC INFRASTRUCTURE 2](#_Toc500162489)

[3.2 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) 6](#_Toc500162490)

[3.3 OPERATIONS MONITORING 6](#_Toc500162491)

[3.4 MISCELLANEOUS 7](#_Toc500162492)

## 1 INTRODUCTION & BACKGROUND

Logan Data Inc., located at 2 Lan Dr., Ste. 200, Westford MA, provides Data Consulting Services. Logan Data had worked previously on several Data Integrations Projects for Fresenius Medical Care, SnapLogic, Insulet and various other clients. Logan Data has Master Service agreement with Boston University. The company has experienced employees with varied and diversified skillsets who are SME’s and Certified Professionals in various consulting spheres.

## 2 OBJECTIVES

This Recommendation Report is based on the initial discussions that Logan Data team had with Boston University Integrations Team and is based on our initial study done with BU Team.

## 3 RECOMMENDATIONS

Initial Recommendations are proposed under 2 major areas: viz. SnapLogic Infrastructure, SDLC and Miscellaneous.

### 3.1 SNAPLOGIC INFRASTRUCTURE

**Recommendation# 1**

Limit Admin Access/Privileges

* Have 2/3 users who are designated as admins. Only their login ids will have the privileges to do admin activities
* Have a admin login set up, say bu\_slp\_admin@bu.edu. Only this user will have the rights to do admin changes. Its credentials can be shared between 2 or 3 individuals for logistical ease.

**Recommendation# 2**

User Groups and Guest Developers

* We should think about setting up groups by roles and functions. For example, a group named like Developer can be set up. Individuals who have the role/function of development can be added to this group. Once a project is initiated then the group Developer can be permitted access on the project, say when “Integration Performance Reporting” project is initiated, Developer group with appropriate access like Read Only, Read & Write, Read & Execute, Full Access, etc.
* If there are end-users who need to execute a certain project/pipeline they can be set up in the equivalent manner too
* In case Guest Developers like Contractors are assigned to projects then they can be put under a Guest Group which is granted access to the project OR they can be assigned specific project access based on need.
* For higher environments like QA and PROD only admin user will have the privileges. All others should have read-only privileges

**Recommendation# 3**

Service Accounts

* Accounts which are used to access different end-points or different applications store credentials within them. Putting them under Shared project should be evaluated on a case-by-case basis and project need. For example, for Integration Performance Reporting if access is needed to Aurora on AWS then rather than putting it under admin-integration-services/shared; put it under “Accounts” in admin-integration-services/Integration Performance Reporting

**Recommendation# 4**

SnapLogic Environment Structure

* Create SnapLogic Project Spaces by the 6 Service Areas in BU and their sub-areas. This will allow development teams to instinctively know where to find pipelines, projects and other details for the domain or area that they are working.
* Use an intelligent naming convention for projects.

Following table describe the naming convention for various SnapLogic Project Spaces. Second table lists out a few projects and the way they are named.

|  |  |
| --- | --- |
| **PROJECT SPACE** | **Description** |
| **• Administrative Systems & Reporting** | |
| admin-appdev | Application Development |
| admin-dar | Development & Alumni Systems |
| admin-facilities-management-services | Facilities Management Systems |
| admin-fin-hr-proc | Finance/HR/Procurement Systems |
| admin-integration-services | Integration Services |
| admin-research-systems | Research Administration Systems |
| admin-the-links | The Links |
| admin-auxiliary-systems | Auxiliary Systems |
| admin-document-management | Document Imaging & Management |
| admin-faculty-systems | Faculty Systems |
| admin-galaxy | GALAXY |
| admin-report-analytics | Reporting & Analytics |
| admin-student-systems | Student Systems |
| **• Research Computing** | |
| research-computation | Research Computation |
| research-visualization | Research Visualization |
| research-consulting-training | Research Consulting & Training |
| **• Information Security & Business Continuity** | |
| security-cyber-security | Cyber Security Incident Response |
| security-device | Device, Data, & Computer Security |
| security-network | Network Security Services |
| security-policy | Security Policy & Consulting Services |
| security-data-protection | Data Protection & Encryption |
| security-iam | Identity & Access Management (IAM) |
| security-education | Security Education & Awareness |
| security-server | Server Security Services |
| **• Client Computing & Collaboration Services** | |
| cccs-av | Audio/Visual Services |
| cccs-collaboration | Collaboration & Social Media |
| cccs-desktop | Desktop Computing |
| cccs-mobile | Mobile Services |
| cccs-remote | Remote Access Services |
| cccs-service-management | Service Management Systems |
| cccs-television | Television |
| cccs-campus-safety | Campus Safety Services |
| cccs-conf | Conferencing |
| cccs-email | Email & Calendaring |
| cccs-printing | Printing |
| cccs-research-sites | Research Sites |
| cccs-phone | Telephone Services |
| cccs-websites | Websites |
| **• Infrastructure** | |
| infrastructure-hosting | Hosting Services & Technical Administration |
| infrastructure-storage backup | Storage & Backup Services |
| infrastructure-networks | Network Services |
| **• Teaching & Learning Technologies** | |
| teaching-assessment | Assessment Systems |
| teaching-e-portfolio-sites | e-Portfolio Sites |
| teaching-lms | Learning Management Systems |
| teaching-classroom | Classroom Technology & Support |
| teaching-instructional-video | Instructional Video Services |

|  |  |
| --- | --- |
| **Sample Project Names** | **Remarks** |
| admin-integration-services/Integration-Performance-Reporting | Project has data sourced entirely in Integration Services Domain |
| admin-research-systems/RIMS-Load-BioRAFT-Environmental-Compliance-Data | Project has data sourced in RIMS (Research Information Management Systems) and loads BioRAFT. |
| admin-research-systems/BioRAFT-Load-lab-and-researcher-profiles | Project has data sources in BioRAFT Domain |
| admin-research-systems/RIMS-Load-Compliance-Data-To-AWSAurora | This project takes data from RIMS under "Research Administration Systems" domain and uploads it to Cloud on Aurora in AWS which is a component of "Integration Services" under "Administrative Systems & Reporting" area. But it will be resident under admin-research-systems as it is the data source |

### 3.2 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

**Recommendation# 1**

Create Developer Checklist

Every development process should maintain a dev checklist. Common items to include on the checklist:

* Are users and accounts needed for project created as per recommended guidelines?
* Is the project named as per recommended guidelines?
* Are correct names on snaps given; comments on the snaps given and as per recommended guidelines? etc.

Only when a developer can tick off the checklist it should go on for Peer Code Review phase

**Recommendation# 2**

Code Review/Peer Code Review

* Code Review and Peer Code Review should be strictly enforced as a 'must-do' step in SDLC. Code Review should be done not only for syntax, best practices, semantics but performance considerations also should be considered.

**Recommendation# 3**

Unit Testing and SIT

* Performance testing should be made part of this on a case-by -case basis. This would include making production-like data available for testing pipelines and/or spinning off copies of sources and targets to Dev which would mimic those in production.

### 3.3 OPERATIONS MONITORING

**Recommendation# 1**

Setup Ops or ProdSupport or DevOps Team

* A dedicated Ops or ProdSupport or DevOps Team should be set up. This is important as Operational Monitoring is needed only on and set up only on Production Org and the whole team should not have access to Prod Environment. There should be checks and controls in place.
* Effort is already underway to set up a common standard Error Handler/Monitoring Pipeline which sends an error e-mail whenever a pipeline fails. Production pipelines should ideally be setup with error handler e-mails being directed to Ops Team.

**Recommendation# 2**

Triage of Production Errors

* Level 1 Support should come from Ops Team. Thus, any production error that comes out should be triaged and resolved by Ops Team. Level 2 support will be provided by Dev Team and SMEs. They will be engaged by L1 Team as needed.
* As the process matures, mechanism to create Incident Tickets in ServiceNow and/or other suitable system can be thought of. This will allow to maintain a trail of the of the production issue and its resolution process.

**Recommendation# 3**

Error Resolution Process

* Ops Team should be trained in SnapLogic Error Resolution Process especially in the usage of SnapLogic Dashboard. SnapLogic Dashboard is a very useful tool to debug pipeline errors as well as many other details like overall Health of SL system, graphical measures for Snaplex, etc.
* When using features like Execute Pipeline, the error may actually happen in the child pipeline. Parent pipeline will error out but to see the exact error message and other debug information, team would need to go to the child pipeline and pull up its details in dashboard.

### 3.4 MISCELLANEOUS

**Recommendation# 1**

Git Hub Revisioning For Source Control

* Strengthen automation around GitHub so that project changes can be captured and uploaded to GitHub seamlessly.
* Will need some brainstorming between the team to come up with a final design like including branches or not, allow multiple developers to work on the same code or not. If this is done, then how best to achieve collaboration on merging the changes to main branch.
* In GitHub Integration Pipelines, one thought is to use different GitHub Accounts for REST GET and REST PUT to avoid hitting request rate limit which is 5000 per hour. This may also help in the performance of REST GET/PUT Snaps.
* Thought needs to be given to components that are checked in to GitHub like

1. Confidentiality of information in the file
2. Size of the files as pipelines may generate huge files during execution which we don’t necessarily want to version control
3. In case we decide not to put everything in GitHub, we will need to come up with a file naming convention so that qualifying files can be committed into GitHub