Creating a TechZone-based StreamSets Demo Environment

Table of Contents

[Introduction 2](#_Toc174544799)

[Provision a TechZone-based StreamSets Data Plane Instance 2](#_Toc174544800)

[Choose your deployment method 3](#_Toc174544801)

[Log in to StreamSets Platform 6](#_Toc174544802)

[1. If you already have an account on StreamSets Platform 6](#_Toc174544803)

[2. If you do not yet have an account on StreamSets Platform 6](#_Toc174544804)

[Deploy a StreamSets Engine 8](#_Toc174544805)

[1. Clone the Hands On Lab Deployment 8](#_Toc174544806)

[2. Set Permissions on the Deployment 10](#_Toc174544807)

[3. Start the Deployment 12](#_Toc174544808)

[4. Generate an Engine Deployment Script 13](#_Toc174544809)

[5. Run the Engine Deployment Script 14](#_Toc174544810)

[6. Confirm the Engine is Healthy 15](#_Toc174544811)

[Run a Test Pipeline 16](#_Toc174544812)

# Introduction

This document describes how to provision and configure a TechZone-based StreamSets Demo Environment.

# Provision a TechZone-based StreamSets Data Plane Instance

Provision an instance of the Data Plane for StreamSets VM in TechZone at this link: <https://techzone.ibm.com/collection/streamsets/environments>

**Dev Data Generator**

Ddd

**Make ID > 0**

dd

**Create account number**

from random import randint

for record in sdc.records:

try:

num\_1 = ''.join(["{}".format(randint(0, 9)) for num in range(0, 3)])

num\_2 = ''.join(["{}".format(randint(0, 9)) for num in range(0, 7)])

account\_number = '000{}-{}'.format(num\_1, num\_2)

record.value['account\_number'] = account\_number

sdc.output.write(record)

except Exception as e:

sdc.error.write(record, str(e))

**Generate amount and type**

import random;

for record in sdc.records:

try:

record.value['amount'] = round(random.uniform(1.5, 12000.0),2)

record.value['account\_type'] = sdc.state['account\_type'][sdc.state['index\_1']]

sdc.state['index\_1'] = (sdc.state['index\_1'] + 1) % len(sdc.state['account\_type'])

record.value['type'] = sdc.state['type'][sdc.state['index\_2']]

sdc.state['index\_2'] = (sdc.state['index\_2'] + 1) % len(sdc.state['type'])

sdc.output.write(record)

except Exception as e:

sdc.error.write(record, str(e))

**Add Timestamp**

Output Field:

/timestamp

Field Expression:

${time:extractStringFromDate(time:now(), "yyyy-MM-dd HH:mm:ss")}

A screenshot of a computer

AI-generated content may be incorrect.

**Kafka**

A screenshot of a computer

AI-generated content may be incorrect.

**Generate random order**

ddd

**Generate customer id**

Jyton

from random import randint

# Process records

for record in sdc.records:

try:

num = str(randint(1, 10000))

customer\_id = '000{}'.format(num)

record.value['customer\_id'] = customer\_id

sdc.output.write(record)

except Exception as e:

# Send record to error handling

sdc.error.write(record, str(e))

**Generate amount and payment**

Jyton

import random

# Mapping payment type

payment\_type\_map = {

1: 'voucher',

2: 'credit\_card',

3: 'boleto'

}

# Process records

for record in sdc.records:

try:

record.value['payment\_amount'] = round(random.uniform(50, 100000.0), 2)

record.value['payment\_type'] = payment\_type\_map[random.randint(1, 3)]

sdc.output.write(record)

except Exception as e:

# Send record to error handling

sdc.error.write(record, str(e))

**Add timestamp**

Output Field:

/purchase\_datetime

Field Expression:

${time:extractStringFromDate(time:now(), "yyyy-MM-dd HH:mm:ss")}

**Transaction records**

A screenshot of a computer screen

AI-generated content may be incorrect.

Order

ddd

ddd

ddd

ddd

ddd

ddd

ddd

ddd

ddd

ddd

ddd