Import Intake Form

In main:

```
    print("")
    print("****************************")
    print("*Login to Migration factory*")
    print("****************************")
    token = mfcommon.Factorylogin()

    print("****************************")
    print("*Reading intake form List*")
    print("****************************")
    data = get_reader(args.Intakeform)
    print("Intake form data loaded for processing....")
    print("")

    print("***********************************************")
    print("*Creating resources in the migration factory*")
    print("***********************************************")

    r = uploading_data(data,token,UserHOST)
```

What does Factorylogin() do?

- if UserPoolId and Region are defined in the FactoryEndpoints.json
  - then try to read the username and password from secretsmanager
  - if that fails, prompt for username and password
- else
  - if 'DefaultUser' in mf_config then only prompt for password
  - else prompt for username and password
- Post the username and password to the LoginApiUrl endpoint
- if 200, then return token.

```
def Factorylogin():
    username = ""
    password = ""
    using_secret = False
    if 'UserPoolId' in mf_config and 'Region' in mf_config:
        try:
            secretsmanager_client = boto3.client('secretsmanager', mf_config['Region'])
            # mf_service_account_details = secretsmanager_client.describe_secret(SecretId=
            mf_service_account = secretsmanager_client.get_secret_value(SecretId='MFServi
            #username = mf_service_account_details['Description']
            mfauth = json.loads(mf_service_account['SecretString'])
            username = mfauth['username']
            password = mfauth['password']
            using_secret = True
        except botocore.exceptions.ClientError as e:
            print(e)
            if e.response['Error']['Code'] == 'ResourceNotFoundException' or e.response['
                print("Service Account doesn't exist or access is denied to Secret, pleas
                if 'DefaultUser' in mf_config:
                    DefaultUser = mf_config['DefaultUser']
                else:
                    DefaultUser = ''
                username = input("Factory Username [" + DefaultUser + "]: ") or DefaultUs
                password = getpass.getpass('Factory Password: ')
    else:
        if 'DefaultUser' in mf_config:
            DefaultUser = mf_config['DefaultUser']
        else:
            DefaultUser = ""
        username = input("Factory Username [" + DefaultUser + "]: ") or DefaultUser
        password = getpass.getpass('Factory Password: ')
    login_data = {'username': username, 'password': password}
    try:
        r = requests.post(mf_config['LoginApiUrl'] + '/prod/login',
                  data=json.dumps(login_data))
        if r.status_code == 200:
            print("Migration Factory : You have successfully logged in")
            print("")
            token = str(json.loads(r.text))
            return token
        if r.status_code == 502 or r.status_code == 400:
            if using_secret:
                print("ERROR: Incorrect username or password stored in Secrets Manager [MF
            else:
                print("ERROR: Incorrect username or password....")
            sys.exit()
        else:
            print(r.text)
            sys.exit()
    except requests.ConnectionError as e:
```

```
            raise SystemExit("ERROR: Connecting to the Login API failed, please check Logi
                             "If the API endpoint is correct, please close cmd and open a
```

What does get_reader() do?

- reads the file using `csv. DictReader`
- for each row in the file, append to ordered_dict_list

```
def get_reader(file):
    ordered_dict_list = []
    input_file = csv.DictReader(open(file))
    for row in input_file:
        ordered_dict_list.append(row)
    # return input_file
    return ordered_dict_list
```

What does uploading_data() do?

- Gets the current list of waves and apps via the API Gateway
- Calls data_validation() function
- For each row of data passed into uploading_data()
  - Get Unique new Wave Id, add to the list if Wave Id doesn't exist in the factory
  - Get Unique new App Name, add to the resource list if App Name doesn't exist in the factory
  - Get Unique server names, add to the resource list if Server Name doesn't exist in the factory
- Converts dictionaries into json structures and posts them to the Wave, App, and Server endpoints

```
def uploading_data(data, token, UserHOST):
    auth = {"Authorization": token}
    try:
        waves = json.loads(requests.get(UserHOST + waveendpoint, headers=auth).text)
        apps = json.loads(requests.get(UserHOST + appendpoint, headers=auth).text)
    except requests.exceptions.ConnectionError as e:
            raise SystemExit("ERROR: Connecting to User API failed, please check User API
                             "If the API endpoint is correct, please close cmd and open a
    wave_list_csv = []
    app_list_csv = []
    wave_ids = []
    app_list = []
    server_list = []

    for row in data:
        if row['wave_id'].strip() not in wave_list_csv:
            wave_list_csv.append(str(row['wave_id']).strip())
        match = False
        for app in app_list_csv:
            if row['app_name'].lower().strip() == app['app_name'].lower().strip():
                match = True
        if (match == False):
            app_item = {}
            app_item['app_name'] = row['app_name'].strip()
            app_item['wave_name'] = "Wave " + row['wave_id'].strip()
            app_item['aws_region'] = row['aws_region'].strip()
            if len(row['aws_accountid']) == 12 and row['aws_accountid'].isdigit():
               app_item['aws_accountid'] = row['aws_accountid'].strip()
            else:
               print("Error: Invalid AWS Account Id for app: " + row['app_name'].strip())
               sys.exit()

            app_list_csv.append(app_item)

    data_validation(data, app_list_csv)

    # Get Unique new Wave Id, add to the list if Wave Id doesn't exist in the factory
    for wave_id in wave_list_csv:
        match = False
        for wave in waves:
            wave_name = "Wave " + wave_id
            if str(wave_name) == str(wave['wave_name']):
                match = True
        if (match == False):
            wave_ids.append(wave_id)
    if len(wave_ids) != 0:
        wave_ids.sort()
        print("New Waves: ")
        print("")
        for wave in wave_ids:
```

```
        print("Wave " + wave)
    print("")
    # Creating new Waves in the migration factory
    for wave in wave_ids:
        wave_name = {}
        wave_name['wave_name'] = "Wave " + wave
        try:
            r = requests.post(UserHOST + waveendpoint, headers=auth, data=json.dumps(
            if r.status_code == 200:
                print("Wave " + wave + " created in the migration factory")
            else:
                print("Wave " + wave + " failed : " + r.text + ".......")
                sys.exit()
        except requests.exceptions.ConnectionError as e:
               raise SystemExit("ERROR: Connecting to User API failed, please check Us
                                "If the API endpoint is correct, please close cmd and
    print("")
    print("----------------------------------------")
    print("")

# Get Unique new App Name, add to the resource list if App Name doesn't exist in the
for app_csv in app_list_csv:
    try:
        new_waves = json.loads(requests.get(UserHOST + waveendpoint, headers=auth).tex
    except requests.exceptions.ConnectionError as e:
            raise SystemExit("ERROR: Connecting to User API failed, please check User
                             "If the API endpoint is correct, please close cmd and oper
    for wave in new_waves:
        if app_csv['wave_name'] == str(wave['wave_name']):
            app_csv['wave_id'] = wave['wave_id']
            del app_csv['wave_name']
            break
    match = False
    for app in apps:
        if app_csv['app_name'].lower().strip() == app['app_name'].lower().strip():
            match = True
            if app_csv['wave_id'] != app['wave_id']:
                print("Error: Wave_id for app " + app_csv['app_name'] + " doesn't mat
                sys.exit(2)
            if app_csv['aws_region'] != app['aws_region']:
                print("Error: aws_region for app " + app_csv['app_name'] + " doesn't
                sys.exit(3)
            if app_csv['aws_accountid'] != app['aws_accountid']:
                print("Error: aws_accountid for app " + app_csv['app_name'] + " doesn
                sys.exit(4)
    if (match == False):
        app_list.append(app_csv)
if len(app_list) != 0:
    print("New Apps: ")
    print("")
    for app in app_list:
```

```
        print(app["app_name"])
    print("")
    # Creating new Apps in the migration factory
    for app in app_list:
        try:
            r = requests.post(UserHOST + appendpoint, headers=auth, data=json.dumps(a
            if r.status_code == 200:
                print("App " + app['app_name'] + " created in the migration factory")
            else:
                print("App " + app['app_name'] + " failed : " + r.text + ".......")
                sys.exit()
        except requests.exceptions.ConnectionError as e:
               raise SystemExit("ERROR: Connecting to User API failed, please check U
                                "If the API endpoint is correct, please close cmd and
    print("")
    print("----------------------------------------")
    print("")
# Get Unique server names, add to the resource list if Server Name doesn't exist in t
try:
    newapps = json.loads(requests.get(UserHOST + appendpoint, headers=auth).text)
except requests.exceptions.ConnectionError as e:
        raise SystemExit("ERROR: Connecting to User API failed, please check User API
                         "If the API endpoint is correct, please close cmd and open a
for row in data:
    for app in newapps:
        if row['app_name'].lower().strip() == app['app_name'].lower().strip():
           row['app_id'] = app['app_id']
    tags = []
    tag = {}
    server_item = {}
    server_item['server_name'] = row['server_name'].strip()
    tag['key'] = 'Name'
    tag['value'] = row['server_name'].strip()
    tags.append(tag)
    server_item['tags'] = tags
    server_item['app_id'] = row['app_id'].strip()
    server_item['server_os_family'] = row['server_os_family'].strip()
    server_item['server_os_version'] = row['server_os_version'].strip()
    server_item['server_fqdn'] = row['server_fqdn'].strip()
    server_item['server_tier'] = row['server_tier'].strip()
    server_item['server_environment'] = row['server_environment'].strip()
    server_item['subnet_IDs'] = convert_string_to_list(row['subnet_IDs'].strip())
    server_item['securitygroup_IDs'] = convert_string_to_list(row['securitygroup_IDs'
    server_item['subnet_IDs_test'] = convert_string_to_list(row['subnet_IDs_test'].st
    server_item['securitygroup_IDs_test'] = convert_string_to_list(row['securitygroup_
    if 'instanceType' in row:
        server_item['instanceType'] = row['instanceType'].strip()
    if 'iamRole' in row:
        server_item['iamRole'] = row['iamRole'].strip()
    if 'private_ip' in row:
        server_item['private_ip'] = row['private_ip'].strip()
```

```
        if 'tenancy' in row:
            server_item['tenancy'] = row['tenancy'].strip()
        server_list.append(server_item)
    if len(server_list) != 0:
        print("New Servers: ")
        print("")
        for server in server_list:
            print(server['server_name'])
        print("")
        # Creating new Apps in the migration factory
        for server in server_list:
            try:
                r = requests.post(UserHOST + serverendpoint, headers=auth, data=json.dumps
                if r.status_code == 200:
                    print("Server " + server['server_name'] + " created in the migration 
                else:
                    print("ERROR: " + server['server_name'] + " failed : " + r.text + ".."
            except requests.exceptions.ConnectionError as e:
                   raise SystemExit("ERROR: Connecting to User API failed, please check Us
                                    "If the API endpoint is correct, please close cmd and
```