Configuration Manual

MSc Research Project

MSc in Cloud Computing

Ravindra Ardhapure
Student ID: x21189838

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ravindra Ardhapure
Student ID:	x21189838
Programme:	Masters in cloud comuting **Year:** 2022-2023
Module:	Msc in cloud computing
Lecturer:	Vikas Sahni
Submission Due Date:	14/08/2023
Project Title:	LSTM based Predictive Network for Video Anomaly Detection
Word Count:	**866 Page Count: 11**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ravindra Ardhapure

Date: 10/08/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□

You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ravindra Ardhapure
Student ID: x21189838

1 Introduction

All the requirements for replicating the study and its results on local and Azure ML cloud environment is provided in this Configuration Manual. The system specifications for local run, dataset source, Python ML packages, configuring the cloud environment using Azure SDK v1 and the Azure pipeline execution model for the project are detailed in this manual.

2 System Specifications

Hardware Configuration for the local run:

- Processor: Intel 11th Gen Core i5-1135G7 @2.4 GHz
- RAM: 16 GB DDR4 RAM 3200MHz
- Storage (SSD): 512GB
- Operating System: Windows 10, 64-bit

Software Packages for the local run:

- Python 3.6.4
- Anaconda Navigator 2.3.2
- PyCharm IDE Community Edition 2021.3
- Jupyter Notebook

3 ML Packages

The following ML packages were installed for the code development both locally and on Azure ML cloud. The project environment requirements were compiled in .yml file for easy environment setup along with packages installation.

On a local computer, use the following command in the windows terminal,

conda env create -f config/environment.yml

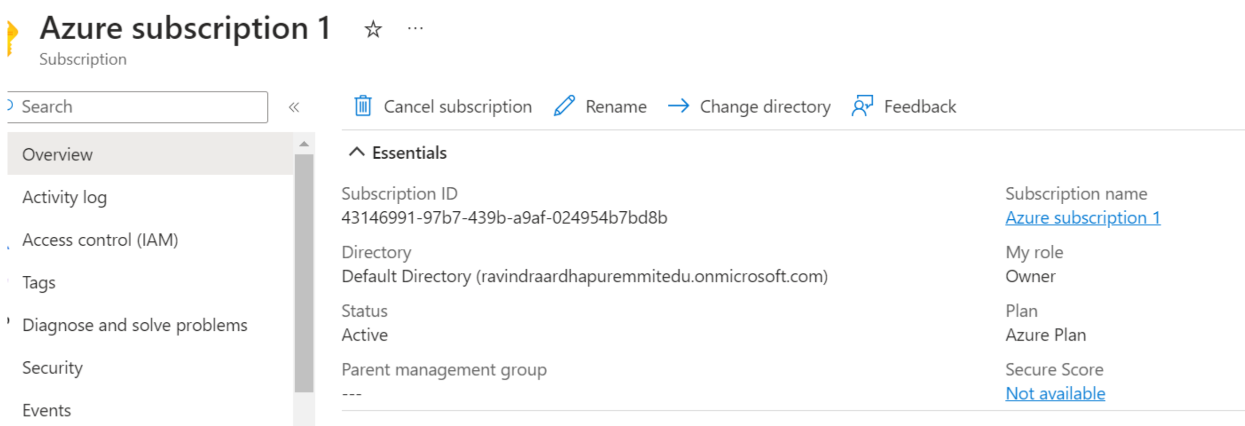
```
%%writefile conda_dependencies.yml

dependencies:
- python=3.6
- cudatoolkit=10.0
- pip:
  - azureml-sdk==1.47.0
  - azure-storage-blob==2.1.0
  - tensorflow-gpu==1.15
  - keras==2.1.6
  - h5py==2.10.0
  - scikit-learn
  - pandas
  - numpy
  - matplotlib
  - pillow==6.0.0
  - seaborn
  - hickle
  - requests==2.21.0
```

Figure 1: Conda Dependencies Installation for Azure ML environment Setup

4 Dataset

The dataset used in this work is UCSD anomaly detection dataset collected from the source, http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm

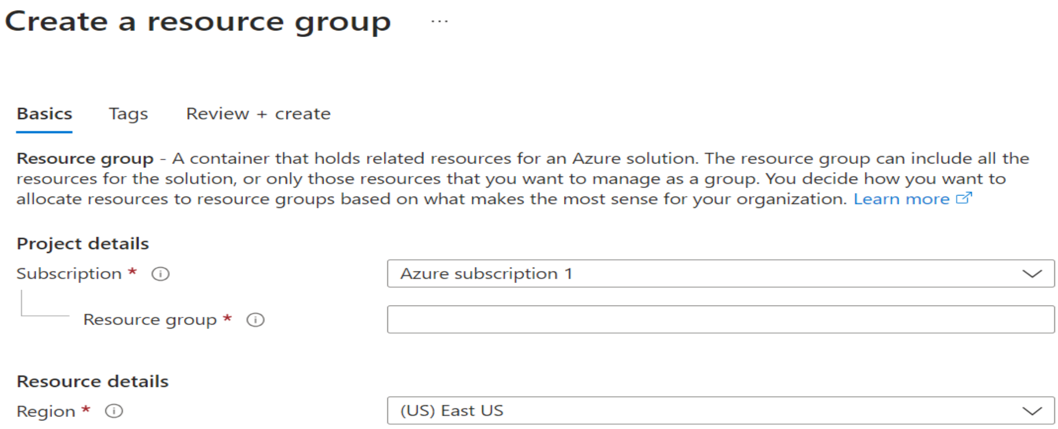
5 Azure ML Configuration



- Create an Azure Subscription for use with your research

Figure 2: Azure Subscription Details

- Create a resource group under your subscription name to add your Azure ML workspace with the subscription name and region information.
- . **Figure 3: Azure Resource Group Creation**



- Create an Azure ML workspace within the resource group

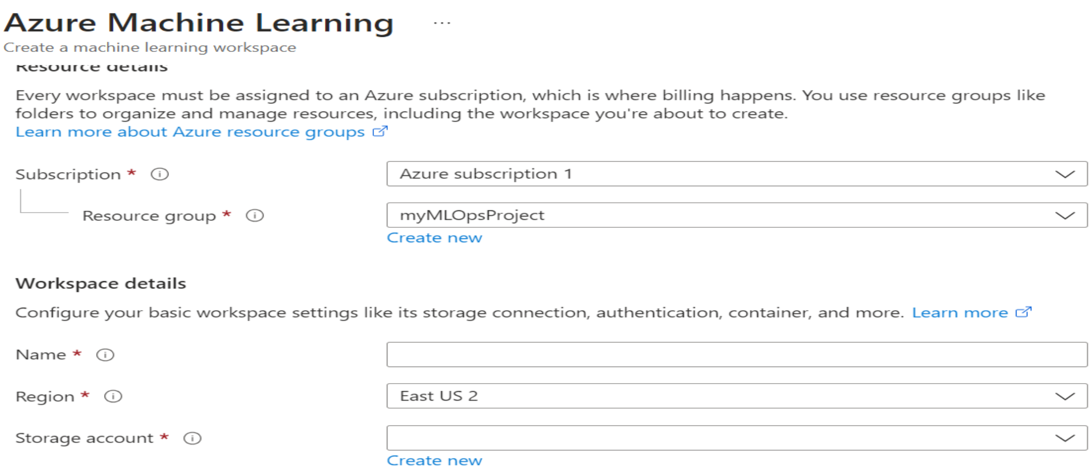


Figure 4: Azure ML Workspace creation

- Set to work with our code after the above steps. The ML workspace after creation is presented in Figure 4. Make sure, the details shown are noted down are saved locally in a notepad to be used later with Azure SDK for Python. Take a note of the subscription id, Subscription Name, Resource group, Location etc.,

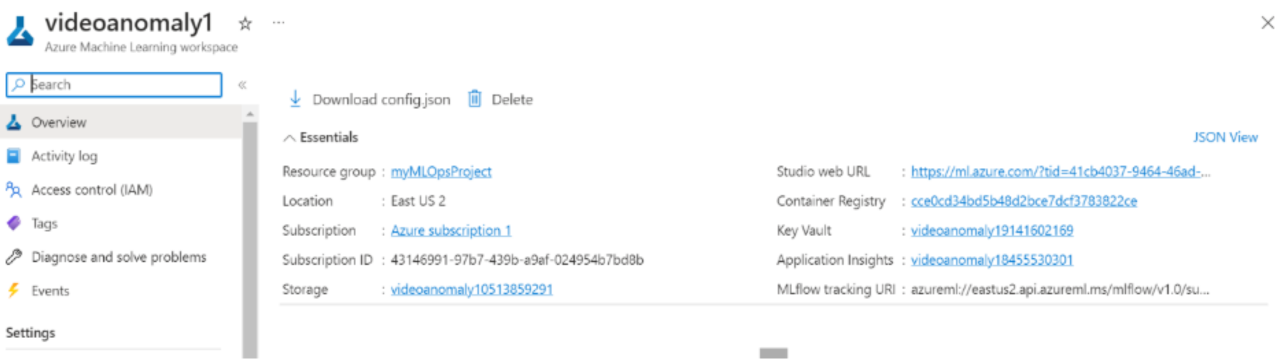


Figure 5: Azure ML Workspace Details

- Create Service Principal in Azure active directory for non-interactive authentication

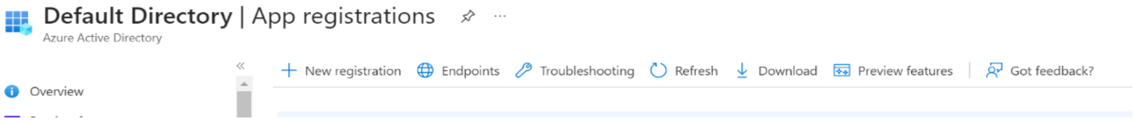


Figure 6: Azure Active Directory→App Registrations→ New Registration

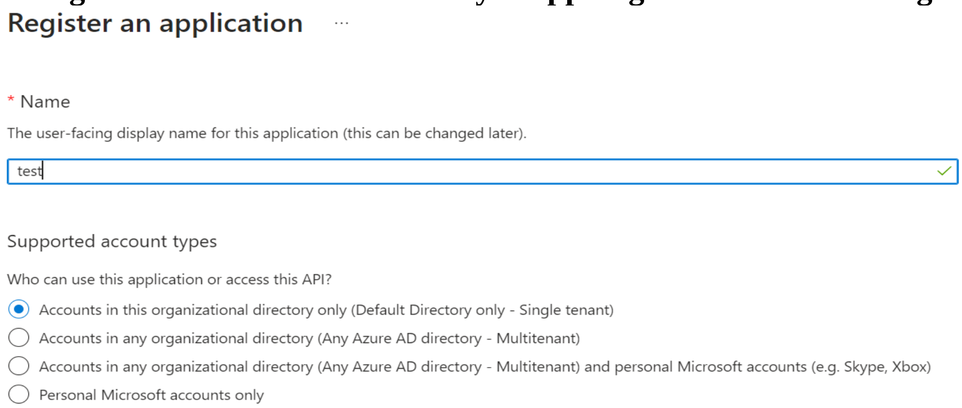


Figure 7: New Registration→Register an application

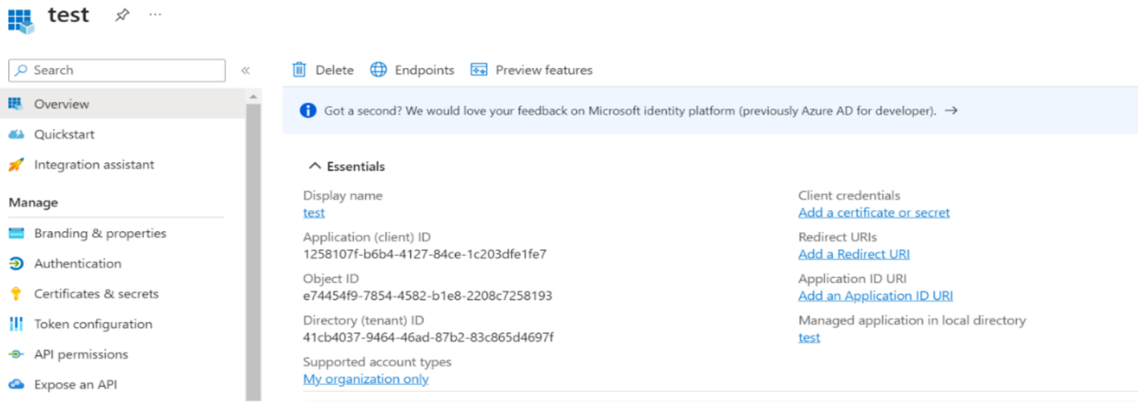


Figure 8: Created Service Principal Authentication.

Service principal authentication requires a secret key which can created from "*add a certificate or secret*" link from Figure 8. Note down, the service principal id, service principal password and tenant id.

```
"subscription_id": "43146991-97b7-439b-a9af-024954b7bd8b",
"resource_group": "myMLOpsProject",
"workspace_name": "videoanomaly1",
"workspace_region": "eastus2",
"cpu_compute": "cpu-cluster",
"gpu_compute": "gpu-cluster",
"service_principal_id":  "b0b8284b-95ef-4086-a8c0-209a5fae4727",
"service_principal_password": "tt~8Q~iEGoaroNfR17LY9toBCJmCjte1K_6EWcei",
"tenant_id": "41cb4037-9464-46ad-87b2-83c865d4697f"
```

- Config.json file with the Azure configuration details. This will be called upon when we create the pipelines using pipelines_master.py file.

Figure 9: Azure ML configuration File.

6 Azure ML Pipelines

- Azure ML Pipelines are created using pipelines master.ipynb file run using Jupyter notebook on your local system.

```python
base_dir = "."

config_json = os.path.join(base_dir, "config.json")
with open(config_json, "r") as f:
    config = json.load(f)

auth = ServicePrincipalAuthentication(
    tenant_id=config["tenant_id"],
    service_principal_id=config["service_principal_id"],
    service_principal_password=config["service_principal_password"],
)

ws = Workspace(
    config["subscription_id"],
    config["resource_group"],
    config["workspace_name"],
    auth=auth
)

print(ws.get_details)

keyvault = ws.get_default_keyvault()
keyvault.set_secret("tenantID", config["tenant_id"])
keyvault.set_secret("servicePrincipalId", config["service_principal_id"])
keyvault.set_secret(
    "servicePrincipalPassword",
    config["service_principal_password"])
```

Figure 10: Azure ML configuration for pipeline

```python
# folder for scripts that need to be uploaded to Aml compute target
script_folder = "./scripts/"
if os.path.exists(script_folder):
    print("Deleting:", script_folder)
    shutil.rmtree(script_folder)
os.makedirs(script_folder)

shutil.copy(os.path.join(base_dir, "utils.py"), script_folder)
shutil.copy(os.path.join(base_dir, "pipelines_slave.py"), script_folder)
shutil.copy(os.path.join(base_dir, "train.py"), script_folder)
shutil.copytree(
    os.path.join(base_dir, "models"),
    os.path.join(base_dir, script_folder, "models"))

shutil.copy(os.path.join(base_dir, "data_preparation.py"), script_folder)
shutil.copy(os.path.join(base_dir, "batch_scoring.py"), script_folder)
shutil.copy(os.path.join(base_dir, "train_clf.py"), script_folder)
shutil.copy(os.path.join(base_dir, "register_clf.py"), script_folder)
```

Figure 11: Pipeline folders to be uploaded to AML compute target

- Creating the CPU compute target

```python
cpu_compute_name = config["cpu_compute"]
print("creating new CPU compute target")

provisioning_config = AmlCompute.provisioning_configuration(
        vm_size="STANDARD_DS12_V2",
        max_nodes=2,
        idle_seconds_before_scaledown=1800,
        #vm_priority="lowpriority"
    )
cpu_compute_target = ComputeTarget.create(
        ws, cpu_compute_name, provisioning_config
    )
cpu_compute_target.wait_for_completion(
        show_output=True, min_node_count=None, timeout_in_minutes=20
    )
print(cpu_compute_target.get_status())
```

Figure 12: Create AML CPU compute target

- Create ML environment on Azure ML cloud

```
%%writefile conda_dependencies.yml

dependencies:
- python=3.6
- cudatoolkit=10.0
- pip:
  - azureml-sdk==1.47.0
  - azure-storage-blob==2.1.0
  - tensorflow-gpu==1.15
  - keras==2.1.6
  - h5py==2.10.0
  - scikit-learn
  - pandas
  - numpy
  - matplotlib
  - pillow==6.0.0
  - seaborn
  - hickle
  - requests==2.21.0
```

Figure 13: Target Run Environment on Azure ML Cloud

```
# Runconfigs
runconfig = RunConfiguration()
runconfig.environment = env
print("PipelineData object created")

create_pipelines = PythonScriptStep(
    name="create pipelines",
    script_name="pipelines_slave.py",
    compute_target=cpu_compute_target,
    arguments=[
        "--cpu_compute_name",
        cpu_compute_name,
       ],
    source_directory=script_folder,
    runconfig=runconfig,
    allow_reuse=False,
)
print("pipeline building step created")

pipeline = Pipeline(workspace=ws, steps=[create_pipelines])
print("Pipeline created")

pipeline.validate()
print("Validation complete")

pipeline_name = "prednet_master"
disable_pipeline(pipeline_name=pipeline_name, dry_run=False)
disable_pipeline(pipeline_name="prednet_UCSDped1", dry_run=False)
published_pipeline = pipeline.publish(name=pipeline_name)
```

Figure 14: Run configuration for Pipeline creation on Azure ML cloud

```
datastore = ws.get_default_datastore()

with open("placeholder.txt", "w") as f:
    f.write(
        "This is just a placeholder to ensure "
        "that this path exists in the blobstore.\n"
    )

datastore.upload_files(
    [os.path.join(os.getcwd(), "placeholder.txt")],
    target_path="prednet/data/raw_data/",
)
```

Figure 15: Datastore container path

```
schedule = Schedule.create(
    workspace=ws,
    name=pipeline_name + "_sch",
    pipeline_id=published_pipeline.id,
    experiment_name="prednet_master",
    datastore=datastore,
    wait_for_provisioning=True,
    description="Datastore scheduler for Pipeline" + pipeline_name,
    path_on_datastore="prednet/data/raw_data",
    polling_interval=5,
)

print("Created schedule with id: {}".format(schedule.id))

published_pipeline.submit(ws, published_pipeline.name)
```

Figure 16: Create schedule and publish pipeline

References

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* 27(5): 1310–1333.

Feng, G. and Buyya, R. (2016). Maximum revenue-oriented resource allocation in cloud, *IJGUC* 7(1): 12–21.

Gomes, D. G., Calheiros, R. N. and Tolosana-Calasanz, R. (2015). Introduction to the special issue on cloud computing: Recent developments and challenging issues, *Computers & Electrical Engineering* 42: 31–32.

Kune, R., Konugurthi, P., Agarwal, A., Rao, C. R. and Buyya, R. (2016). The anatomy of big data computing, *Softw., Pract. Exper.* 46(1): 79–105.