National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Ryan Bannon
Student ID: 14488478

School of Computing
National College of Ireland

Supervisor: Horacio Gonzalez-Velez

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ryan Bannon
Student ID:	14488478
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Horacio Gonzalez-Velez
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	1052
Page Count:	8

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:	Ryan Bannon
Date:	14th August 2022

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

Ryan Bannon
14488478

1 Introduction

This document provides detailed instructions for setting up and running the research experiments for this project on t2.large AWS EC2 instances. It also highlights the necessary software and platforms used. The conducted research experiments tested the solutions ability to improve latency of cold starts, with primary focus on Apache OpenWhisk. Table 1 lists the integral tools and platforms used in this research. Table 2 lists the software libraries and packages used.

Table 1: Tools and Platforms

Type	**Tool/Platform**
Virtual Machines	Amazon Web Services (AWS) EC2
Operating System	Linux Ubuntu Server 18.04 LTS (HVM)
Serverless Platform	Apache OpenWhisk 1.0.0 (open-source)
Container Technology	Docker 20.10.17 CE
Machine Learning	Google Colab & Keras Tensorflow 2.8.0
Performance/Load Testing	Java (openjdk-11) & Apache JMeter 5.4.3
Programming Language	Python 3.10.4

Table 2: Software pre-requisites

Bash	wsk (openwhisk cli v1), docker-compose 1.21.2, docker-ce-cli, python-pip, containerd, make, curl, npm, ca-certificates, gnupg, lsb-release, zip
Python	pandas, numpy, matplotlib

2 Experiments

A total of 4 main experiments were conducted to validate the hypothesis put forward by this research paper. Phase 1 experiments simulated the Azure Functions data throughput of 2 separate days against Apache OpenWhisk. Phase 2 experiments also simulated the

Azure Functions data of the same days, however, only a window 6 hours was selected. In the background a process consumes the machine learning predictions and heats function containers accordingly. Table 3 breaks this explanation down further.

Table 3: Experiment Breakdowns

Phase	Server	Description
Phase 1	Server 1	Azure Function execution data on 04/02/2021 simulated through Apache OpenWhisk (alias: experiment_1)
	Server 2	Azure Function execution data on 12/02/2021 simulated through Apache OpenWhisk (alias: experiment_2)
Phase 2	Server 1	Azure Function execution data on 04/02/2021 between 12:00 & 18:00 simulated through custom modules with Regression predictions applied (alias: experiment_1_2)
	Server 2	Azure Function execution data on 12/02/2021 between 12:00 & 18:00 simulated through custom modules with GRU predictions applied (alias: experiment_2_2)

2.1 Provisioning AWS Resources

Step 1: Log into Amazon Web Services Portal

Step 2: Locate and open the EC2 service

Step 3: Open Key Pairs in Network & Security group along the left panel

Step 4: Create a key pair by supplying the name, leaving the default RSA type and exporting as .pem file format

Note: Store the downloaded .pem file in a safe location on your client

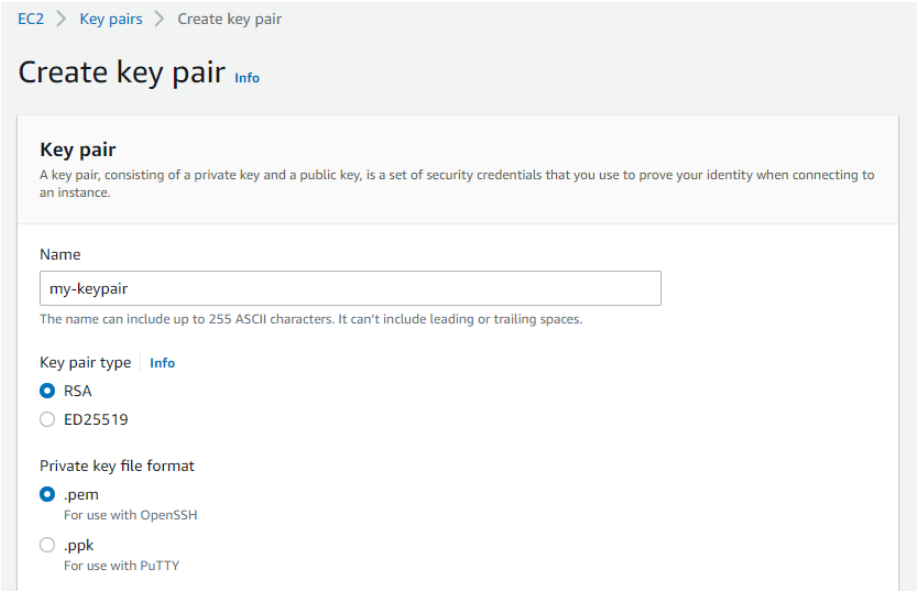

Figure 1: Creating a key pair in AWS

Step 5: Navigate to Instances and click to launch new

Step 6: Enter a name for the virtual machines

Step 7: Search for and select the 'Ubuntu Server 18.04 LTS (HVM), SSD Volume Type' (64-bit) AMI

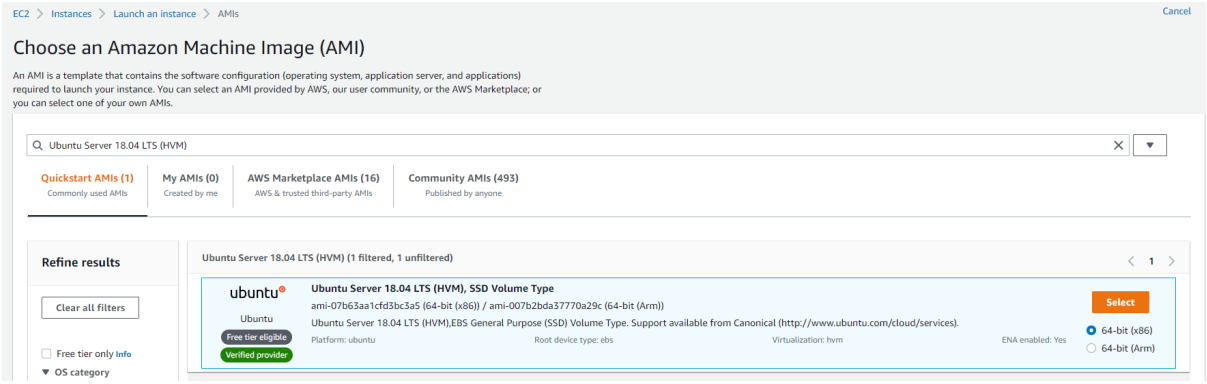

Figure 2: Selecting the required Ubuntu AMI in AWS EC2

Step 8: Increase the number of instances to 2

Step 9: Increase the instance type to t2.large (2vCPUs/8GB RAM), otherwise the experiments will fail

Step 10: Select the previously created key pair

Step 11: Increase the storage of the virtual machines to 16GB

Step 12: Launch the instances

2.2 Connecting to Instances

Optional: Append '-1' & '-2' to the end of the VM names to distinguish one from another when the instances are up and running

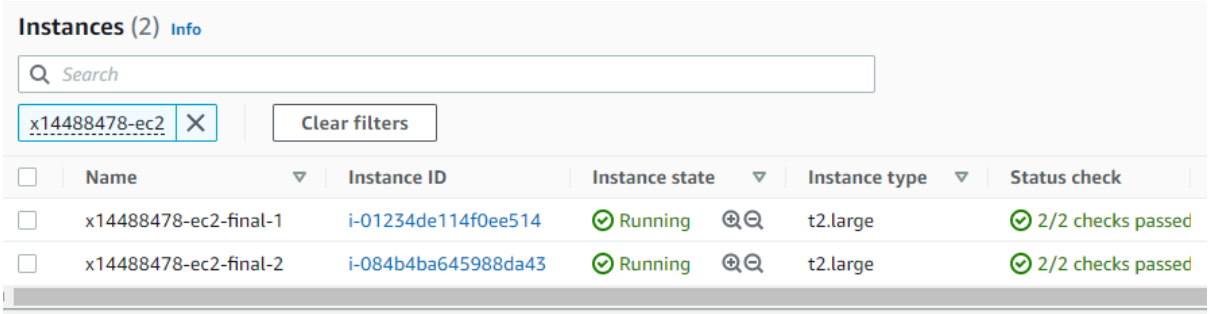

Figure 3: Running Instances in AWS EC2

Step 1: Copy the public IPv4 DNS in the details tab for each of the instances to your clipboard or any other location for quick retrieval

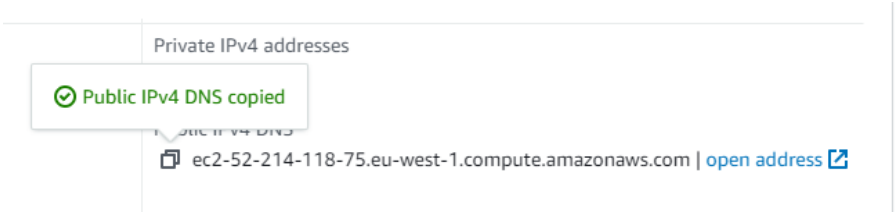

Figure 4: Copying the IP DNS of an EC2 instance to the clipboard

Step 2: Open a terminal session from your client (in the same directory as your .pem file for ease)

Step 3: Enter `ssh -i "{YOUR_KEYPAIR}.pem" ubuntu@{YOUR_EC2_PUBLIC_IP_DNS}` to open an SSH session to the EC2 instances

```
The authenticity of host 'ec2-52-214-118-75.eu-west-1.compute.amazonaws.com (52.214.118.75)' can't be established.
ECDSA key fingerprint is SHA256:qSqBLql85RqY9a9bb+Ho1WTrsFiCO4UJYLp3b7GMafQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-214-118-75.eu-west-1.compute.amazonaws.com,52.214.118.75' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1078-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug 14 09:13:42 UTC 2022

  System load:  0.0               Processes:           106
  Usage of /:   11.4% of 15.33GB  Users logged in:     0
  Memory usage: 4%                IP address for eth0: 172.31.21.62
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

ubuntu@ip-172-31-21-62:~$
```

Figure 5: Connected to EC2 instance via SSH

2.3 Running the Installation

Step 1: Verify git is installed on the VM (which it should be) with the command `git --version`. Type `sudo apt install git` if it's not already installed

Step 2: Now clone the projects codebase from the Github repository with the following command `sudo git clone https://github.com/ryanbannon/openwhisk.git`

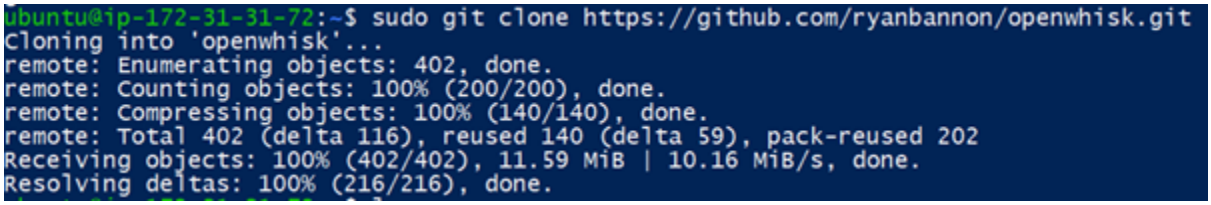

Figure 6: Cloning project repo onto EC2 instance

Step 3: The download and installation of required software tools and dependencies is automated with the install.sh bash file. To run this file enter `sudo bash ~/openwhisk/install.sh > ~/installation.log`

Note: This may take up to 15 minutes

Optional: Once completed, the installation log is available for users to observe and validate completion `nano ~/installation.log`

Step 4: Validate that Docker and OpenWhisk were both installed by observing running containers `sudo docker ps --format '{{.Names}}'`

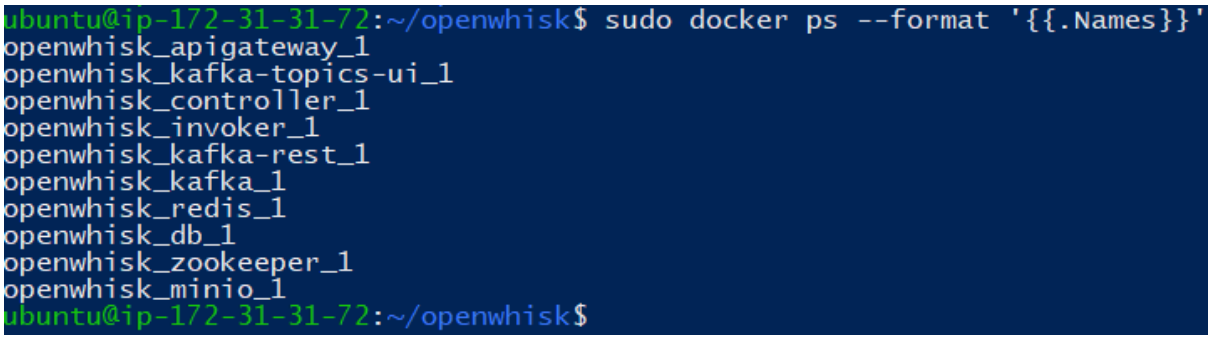

Figure 7: Listing the running Docker containers on the EC2 instance

Step 5: Validate that the required Openwhisk actions were created for the Jmeter workload `sudo wsk -i action list`

Figure 8: Listing OpenWhisk actions on the EC2 instance

Step 6: Use the following commands to move the downloaded Jmeter files and executables to an appropriate location on the server, otherwise Jmeter cannot run
`sudo unzip ~/apache-jmeter-5.4.3.zip`
,
`sudo mv ~/apache-jmeter-5.4.3 ~/jmeter`
,
`sudo mv ~/jmeter /tmp`
,
`echo 'export PATH="\$PATH:/tmp/jmeter/bin"' >> ~/.bashrc`
,
`source ~/.bashrc`

Step 7: Verify Jmeter can be executed `sudo /tmp/jmeter/bin/jmeter.sh --version`

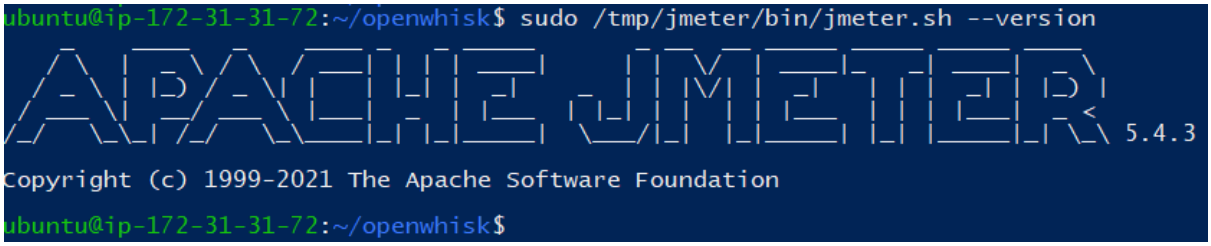

Figure 9: Verifying Jmeter install on the EC2 instance

2.4 Running the Experiments

2.4.1 Experiment Phase 1

Server 1:

Note: This experiment will run for 24 hours

Step 1: Make sure you're in the root directory `cd /home/ubuntu/`

Step 2: Run the Jmeter test plan against OpenWhisk with this command `sudo nohup /tmp/jmeter/bin/jmeter.sh -n -t "/home/ubuntu/openwhisk/jmeter/Experiment_1/Experiment_1.jmx" -l "/home/ubuntu/experiment_1_logs.csv" > /home/ubuntu/experiment_1.log &`

Note: Above command can also be found in the project README file in Github

Server 2:

Note: This experiment will run for 24 hours

Step 1: Make sure you're in the root directory `cd /home/ubuntu/`

Step 2: Run the Jmeter test plan against OpenWhisk with this command `sudo nohup /tmp/jmeter/bin/jmeter.sh -n -t "/home/ubuntu/openwhisk/jmeter/Experiment_2/Experiment_2.jmx" -l "/home/ubuntu/experiment_2_logs.csv" > /home/ubuntu/experiment_2.log &`

Note: Above command can also be found in the project README file in Github

2.4.2 Experiment Phase 2

Server 1:

Note: This experiment will run for 6 hours

Step 1: Make sure you're in the root directory `cd /home/ubuntu/`

Step 2: Run the Jmeter test plan against the custom modules with this command `sudo nohup python /home/ubuntu/openwhisk/controller.py 1 experiment_serverless_1 & /tmp/jmeter/bin/jmeter.sh -n -t "/home/ubuntu/openwhisk/predictions/Experiment_1/Experiment_1_2.jmx" -l "/home/ubuntu/experiment_1_2_logs.csv" > /home/ubuntu/experiment_1_2.log &`

Note: Above command can also be found in the project README file in Github

Server 2:

Note: This experiment will run for 6 hours

Step 1: Make sure you're in the root directory `cd /home/ubuntu/`

Step 2: Run the Jmeter test plan against the custom modules with this command `sudo nohup python /home/ubuntu/openwhisk/controller.py 2 experiment_serverless_2 & /tmp/jmeter/bin/jmeter.sh -n -t "/home/ubuntu/openwhisk/predictions/Experiment_2/Experiment_2_2.jmx" -l "/home/ubuntu/experiment_2_2_logs.csv" > /home/ubuntu/experiment_2_2.log &`

Note: Above command can also be found in the project README file in Github

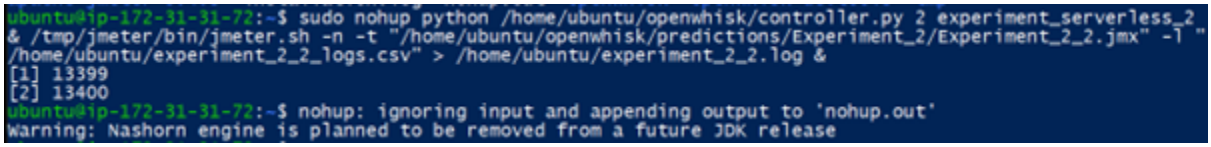

Figure 10: Running experiment_2_2 on EC2 instance

2.5 Evaluating the Results

All log files are retrievable from the root directory. Phase 2 experiments, have an additional results CSV file that contains cold start information. These files were written directly to:
/home/ubuntu/openwhisk/predictions/Experiment_x/experiment_x_2_results.csv