PALS Setup & Deployment Instructions

This document contains instructions to set up the full PALS stack including the data pipeline and web application as well as the development tools necessary to begin working with the stack.

To setup an Ubuntu VM on Cybera:

1. Ensure you have an account with Cybera and go to the following link: https://cloud.cybera.ca/auth/login/?next=/
2. Ensure you perform the steps to setup a private key here: https://wiki.cybera.ca/display/RAC/Rapid+Access+Cloud+Guide%3A+Part+1#RapidAccessCloudGuide:Part1-CreateaKeyPair
3. Go to the Project > Compute > Instances screen.
4. Create and launch an instance with the following configurations:
    a. Details:
        i. Flavor: m1.medium
        ii. Number of Instances: 1
        iii. Instance Boot Source: Boot from image
            1. Image Name: Ubuntu 18.04
    b. Access & Security:
        i. Select your previously made key pair.

    Note: Leave all other settings default.
5. After you create the instance, it will automatically launch.
6. Ensure you associate a floating IP to the instance by clicking on the drop-down after the Actions and selecting “Associate floating IP”. This will be the IP you will connect to when using SSH.
7. Go to Project > Compute > Network > Security Groups
8. Click on “Manage Rules” on the default security rules.
9. Add the following security rules
    a. Rule: SSH, CIDR: 0.0.0.0/0
    b. Rule: SSH, CIDR: ::/0
    c. Rule: Custom TCP Rule, Port: 9000
10. Test connecting to the server by SSH-ing to the above floating IP and login with default credentials.
11. If you have issues connecting through SSH or to any other ports, you may need to create duplicate rules for CIDR: 0.0.0.0/0 and CIDR: ::/0 to account for IPv4 and IPv6 addresses.

On a remote ubuntu 18.04 server:

1. Run ./provision.sh, found in data-pipeline/scripts
2. Add the following to ~/.bash_profile:
   export MINIO_ACCESS_KEY=<access_key>
   export MINIO_SECRET_KEY=<secret_key>
3. Ensure you have port 9000 open for MinIO.
4. Run:
   source ~/.bashrc
5. Run:
   Sudo mysql
6. In MySQL, to setup the users, run the following commands:
   CREATE DATABASE db;
   CREATE USER 'admin'@'localhost' IDENTIFIED BY 'user_password';
   GRANT ALL PRIVILEGES ON db. TO 'admin'@'localhost';*
   CREATE USER 'webapp'@'localhost' IDENTIFIED BY 'user_password';
   GRANT SELECT ON db. TO 'webapp'@'localhost';*

7. For anyone that wants to connect to the database add their ssh public key to the ~./authorized_keys file.
8. Run the following bash code:

```
sudo touch /etc/nginx/sites-available/serve-images
sudo echo 'server {
  server_name <YOUR_SERVER_IP>;
  root /data/images;
  location / {
     try_files $uri =404;
     autoindex on;
     expires 30d;
     add_header Pragma public;
     add_header Cache-Control "public";
  }
}' > /etc/nginx/sites-available/serve-images
sudo ln -s /etc/nginx/sites-available/serve-images
/etc/nginx/sites-enabled/serve-images
sudo service restart nginx
```

On your local machine
Assumptions:

- **python 3.6.8+ with pip installed**
- **Your environment has the Chrome web browser installed**

1. Clone the data-pipeline repo to your local computer
2. In the data pipeline root directory run:

    pip install -r requirements

3. Run the python interactive terminal
4. In the python interactive terminal run the following:

    Import nltk
    download('punkt')
    download('wordnet')

5. Install chromedriver using chocolatey (Windows) or homebrew (Mac).
6. To check that everything is properly installed, change directories into etl/ and run:

    pytest

7. Add a config.ini file into data-pipeline/etl/storage_clients/ with the following **exact** format:

    [minio]
    url=<server_ip>:<minio_server_port>
    access_key=<minio_access_key>
    secret_key=<minio_secret_key>

    [mysql]
    host=127.0.0.1
    port=3306
    user=data
    password=<my_sql_password_for_datapipeline>
    db=db

    [ssh]
    host=<server_ip>

port=22
user=ubuntu
pkey=<path_to_private_key, usually “~/.ssh/id_rsa”>

8. Untar topic_analysis/jst/model.tar.zip
9. Change directories into topic_analysis/jst/Debug and make the c++ project.

Data pipeline should be operational. Please see the user guide on using the data pipeline to populate the database.

To setup a Windows 2016 Server on Cybera:

1. Ensure you have an account with Cybera and go to the following link: https://cloud.cybera.ca/auth/login/?next=/
2. Ensure you perform the steps to setup a private key here: https://wiki.cybera.ca/display/RAC/Rapid+Access+Cloud+Guide%3A+Part+1#RapidAccessCloudGuide:Part1-CreateaKeyPair
3. Go to the Project > Compute > Instances screen.
4. Create and launch an instance with the following configurations:
    a. Details:
        i. Flavor: m1.medium
        ii. Number of Instances: 1
        iii. Instance Boot Source: Boot from image
            1. Image Name: windows-server-2016
    b. Access & Security:
        i. Select your previously made key pair.

    Note: Leave all other settings default.
5. After you create the instance, it will automatically launch.
6. Ensure you associate a floating IP to the instance by clicking on the drop-down after the Actions and selecting “Associate floating IP”. This will be the IP you will connect to when using SSH.
7. Go to Project > Compute > Network > Security Groups
8. Click on “Manage Rules” on the default security rules.
9. Add the following security rules
    a. Rule: SSH, CIDR: 0.0.0.0/0
    b. Rule: SSH, CIDR: ::/0
    c. Rule: HTTP
    d. Rule: RDP, CIDR: <your ip address here>
    e. Rule: Custom TCP Rule, Port: 8172

10. Test connecting to the server via Remote Desktop Connection to the above floating IP and login with default credentials.
11. If you have issues connecting through RDP or to any other ports, you may need to create duplicate rules for CIDR: 0.0.0.0/0 and CIDR: ::/0 to account for IPv4 and IPv6 addresses.

To setup web application hosting environment:

1. Ensure you followed the above steps for setting up a Windows 2016 Server on Cybera.
2. Use a remote desktop connection to connect to your server instance.
3. Once you have connected to your VM, open a PowerShell window in admin mode and run the following commands:
    a. Install Internet Information Services (IIS):
    Install-WindowsFeature -name Web-Server -IncludeManagementTools

    b. Install ASP.NET 4.6:
    Install-WindowsFeature Web-Asp-Net45

    c. Install Web Management Service:
    Install-WindowsFeature -Name Web-Mgmt-Service

    Install Web Deploy 3.6

    # Download file from Microsoft Downloads and save to local temp file (%LocalAppData%/Temp/2)
    $msiFile = [System.IO.Path]::GetTempFileName() | Rename-Item -NewName { $_ -replace 'tmp$', 'msi' } -PassThru
    Invoke-WebRequest -Uri http://download.microsoft.com/download/0/1/D/01DC28EA-638C-4A22-A57B-4CEF97755C6C/WebDeploy_amd64_en-US.msi -OutFile $msiFile
    # Prepare a log file
    $logFile = [System.IO.Path]::GetTempFileName()
    # Prepare the arguments to execute the MSI
    $arguments= '/i ' + $msiFile + ' ADDLOCAL=ALL /qn /norestart LicenseAccepted="0" /lv ' + $logFile
    # Execute the MSI and wait for it to complete
    $proc = (Start-Process -file msiexec -arg $arguments -Passthru)
    $proc | Wait-Process
    Get-Content $logFile

4. Open IIS from the Server Manager.
5. From the left menu, expand the tree and select Sites > Add Website. In this Add Website modal, enter the following:
    a. Set the physical path to your website folder location. Ensure you have permissions “IIS_IUSRS” correctly assigned to the folder.
    b. Set the site name to your web application.
    c. Leave the other settings the same.
6. You may have to pause the default website

Troubleshooting:

- Ensure you installed the .NET Core Hosting bundle from Microsoft: https://www.microsoft.com/net/permalink/dotnetcore-current-windows-runtime-bundle-installer
- If you run into file permission errors: https://stackoverflow.com/questions/5615296/cannot-read-configuration-file-due-to-insufficient-permissions
- Further errors with permissions when accessing the website: https://stackoverflow.com/questions/10418669/hosting-asp-net-in-iis7-gives-access-is-denied
- If you get Error 403 while deploying via MSdeploy.exe: https://stackoverflow.com/questions/8325259/msdeploy-is-returning-403-forbidden
- If you get Error 550 while deploying via MSdeploy.exe: https://stackoverflow.com/questions/22192477/error-550-occurred-by-deploying-via-msdeploy-exe
- Some more tips here: https://docs.microsoft.com/en-us/iis/publish/troubleshooting-web-deploy/troubleshooting-web-deploy-problems-with-visual-studio

To create a config.ini file to use with the web-application:

1. Create a file named “config.ini” and ensure it has the following information in the exact format as shown:

    [minio]
    url=<server_ip>:<minio_server_port>
    access_key=<minio_access_key>
    secret_key=<minio_secret_key>

[mysql]
host=127.0.0.1
port=3306
user=webapp
password=<my_sql_password_for_datapipeline>
db=db

[ssh]
host=<server_ip>
port=22
user=ubuntu
pkey=<path_to_private_key, usually “web_client_private”>

To setup the web application development environment:

1. Download and install at minimum Visual Studio 2019 which has support for our target framework: .NET Core 3.1
    a. In the Visual Studio Installer, ensure you install the “ASP.NET and web development” package.

    See the following for supported versions: https://dotnet.microsoft.com/download/visual-studio-sdks?utm_source=getdotnetsdk&utm_medium=referral

2. Download and install NodeJS: https://nodejs.org/en/download/
3. Clone the web-application repository:
    git clone https://github.com/The-Animals/web-application.git

4. Go to the “web-application\PALS\PALS\ClientApp” directory and run:
    npm install

5. Place the “config.ini” file created from above as well as the private key as specified in the config.ini file in the “web-application\PALS\PALS” directory.
6. Launch Visual Studio and use File>Open>Project/Solution, find and open “web-application\PALS\PALS.sln” to open the project.
7. From here, you should be able to test the web application locally by launching it with IIS Express built into Visual Studio.

Deploy from Visual Studio using WebDeploy:

1. Right-click on the project > select “Publish”.
2. Select the publish target as IIS and create a web deploy profile:
3. Enter your VM’s public IP.
4. Use the same site name you set in the IIS in the “Setup Azure VM with WebDeploy” step.
5. Enter your VM account credentials.
6. You can leave the destination URL blank.
7. Once you have entered the above information, click “Validate Connection”.
8. Accept the Certificate Error.
9. Once the connection is passed, click on “Publish”.
10. Go to your VM’s public IP on a browser and you should see your website up and running.

    Note: You may have to stop the web server through IIS before web deployment as the files will be locked while the server is running, this will interfere with deployment.

References:

https://www.c-sharpcorner.com/article/deploy-web-app-in-azure-vm-using-visual-studio/
https://github.com/aspnet/Tooling/blob/AspNetVMs/docs/create-asp-net-vm-with-webdeploy.md
https://docs.microsoft.com/en-us/azure/virtual-machines/windows/publish-web-app-from-visual-studio
https://support.microsoft.com/en-gb/help/943891/the-http-status-code-in-iis-7-0-iis-7-5-and-iis-8-0