Puddle at H2O Installation and Administration

Release v1.2.0

H2O.ai

Oct 17, 2019
# CONTENTS

## 1 Change Log
1.1 Version 1.2.0 (10/17/2019) .................................................. 3
1.2 Version 1.1.0 (8/21/2019) .................................................. 3

## 2 Installation
2.1 Azure VPC Setup Guide .................................................... 5
2.2 AWS Setup Guide .......................................................... 21

## 3 Starting Puddle at H2O
3.1 Logging Out ............................................................... 39

## 4 Administration
4.1 Systems ................................................................. 41
4.2 Experiments ............................................................. 42
4.3 Tags ......................................................................... 43
4.4 Users ....................................................................... 44
4.5 System Types ........................................................... 46
4.6 Images ................................................................. 48
4.7 Check Updates .......................................................... 49
4.8 Stats Board ............................................................. 50
4.9 Logs ...................................................................... 51
4.10 About ................................................................. 51
Welcome to Puddle at H2O, H2O.ai’s platform for running Driverless AI and H2O-3.

This document describes how Admins can install and monitor Puddle on Azure and AWS. For information on how to create Driverless AI and H2O-3 systems, refer to the Puddle User Guide.

Have Questions?

If you have questions about using Puddle, post them on the H2O.ai Community Slack workspace in the #cloud channel. If you have not signed up for the H2O.ai Community Slack workspace, you can do so here: https://www.h2o.ai/community/.
1.1 Version 1.2.0 (10/17/2019)

Download at https://s3.amazonaws.com/puddle-release.h2o.ai/1.2.0/index.html

- Added a Python client
- Users can now manually fail a system that is in transition
- Automatically install a Driverless AI license on provisioned systems
- The number of available Compute Units is now visible in the UI and can be obtained using the Python client
- Fixed an issue with Azure Active Directory when a user wasn’t granted an Administrator role even though the user was present in a Group that was assigned the Administrator role
- Fixed a glitch in the UI when users directly access /login and they need to log in two times
- Improved handling of background tasks

1.2 Version 1.1.0 (8/21/2019)

Download at http://artifacts.h2o.ai.s3.amazonaws.com/snapshots/ai/h2o/puddle/1.1.0/index.html

- Added Amazon AWS support
- New auth provider - AWS Cognito
- New auth provider - Active Directory
- New auth provider - LDAP
- Automatically deallocate the running VM of failed systems to reduce cost impact
- Email notifications for failed systems, failed deallocation of a running VM, and unknown VM state
- Added proxy support
- Added an option to never timeout a system
- New backup utility
- Swapped the lock/unlock icons in the UI
- Fixed a rare failure that occurred when terminating a system - due to the Public IP still being assigned
Puddle is currently supported on Microsoft Azure and Amazon AWS. Additional cloud offerings will be added at a later date.

2.1 Azure VPC Setup Guide

This topic describes how to set up Puddle on Azure Cloud. This topic is divided into multiple sections:

- Puddle for Azure Architecture
- Setting up Azure Resources like a resource group, vnet, etc.
- Setting up Runtime Dependencies like PostgreSQL database and Redis
- Setting up the Puddle Application

2.1.1 Puddle for Azure Architecture

The image below describes the components that work together to build and run Puddle for Azure.
2.1.2 Set up Azure Resources

The first step is to set up Azure resources. All of these operations require a user login in at portal.azure.com.

Create a Resource Group

1. After you are logged in to Azure, go to the Resource groups blade.
2. Click Add.
3. Fill in the form.
   - Remember the name. We will need the name in later steps.
   - Remember the location. We will need the location in later steps.
4. Click Create.

Create a Network Security Group

1. Search for Network security groups. Note: Do not use the option with (classic) suffix.
2. Click Add.
3. Fill in the form.
   - Place the Network security group into the Resource group created in first step.
• Set the location of the Network security group to the same value that you specified when creating the Resource group.
• Remember the name of the Network security group. We will need the name in later steps.

4. Click **Create**.
5. Select the newly created Network security group.
6. Select **Inbound security rules**.
7. Click **Add**.
8. Set **22, 8888, 12345, 54321** as the **Destination port ranges**.
   • **22** is SSH. We need to open this port to be able to SSH into Virtual machines launched by Puddle.
   • **8888** is Jupyter. We need to open this port to be able to access Jupyter.
   • **12345** is the Driverless AI UI. We need to open this port to be able to access Driverless AI.
   • **54321** is H2O-3. We need to open this port to be able to access H2O-3 and H2O Flow.

### Create Virtual Network and Subnet

1. Go to the **Virtual networks** blade.
2. Click **Add**.
3. Fill in the form.
   • Place the Virtual network into the Resource group created in the first step.
   • Set the location of the Virtual network to the same value that you specified when creating the Resource group.
   • Remember the name of the Virtual network. We will need the name in later steps.
   • Remember the name of the Subnet. We will need the name in later steps.

### Create a Virtual Machine

1. Go to the **Virtual Machines** blade.
2. Click **Add**.
3. Fill in the form.
   • Place the Virtual machine into the Resource group created in the first step.
   • Set the location of the Virtual machine to the same location you created in the first step.
   • Use **Ubuntu Server 18.04 LTS** as Image.
   • Select **Standard B2s** as Size.
   • Select the **Authentication type** that best suites your needs. Note that the **SSH public key** is strongly recommended.
4. Click **Next: Disks**.
5. Under **OS disk type**, select **Standard SSD**.
6. Click **Next: Networking**.
7. Under Virtual network, specify the Virtual network created in the previous step.
8. Under Subnet, specify the Subnet created in the previous step.
9. Set NIC network security group to Advanced.
10. Click Review + create.
11. Click Create.
12. Wait for provisioning to complete.
13. Go to the Virtual machines blade.
14. Select the newly created Virtual machine.
15. Click on Configure next to DNS name.
16. Set Static under Assignment.
17. Pick a DNS name. We will need this DNS name later.
18. Click Save.
19. Go back to the newly created Virtual machine.
20. Select Networking.
21. There should be two security groups available. Make sure the one not explicitly created has rules that allow inbound from ports 22, 80 and 433.
   • If you do not want to allow HTTP connections, then port 80 should not be allowed.
   • If you need to add a rule, then click on Add inbound port rule and fill in the form.
      – set 22, 443 and possibly 80 under the destination port ranges.
   • Click Add.

Create App Registration and Enterprise Application

1. Go to Azure Active Directory blade.
2. Select App Registrations.
3. Click New registration.
4. Fill in the form.
   • In Supported account types, select Accounts in this organizational directory only (msmarketplaceh2o (Default Directory)).
   • Remember the name of the app registration. We will need the name in later steps.
   • This will create the Enterprise application as well.
5. Click Register.
6. Select Manifest.
7. Set the “appRoles” key to the following value:

```json
"appRoles": [
  {
    "allowedMemberTypes": [
      "User"
    ],
    "description": "Users have basic set of permissions in Puddle."
  }
]
```
Each role definition in this manifest must have a different valid GUID for the id key. We will need the id of the Administrator role in later steps.

8. Click Save.
11. Enable both Access tokens and ID Tokens.
12. Click Save.
13. Go to the Azure Active Directory blade.
15. Select the newly created application.
16. Select the Properties blade.
17. Set Yes for the Enabled for users to sign-in? option.
18. If you want only selected users to be able to log in, then set User assignment required? to Yes.
19. Set Visible to users? to Yes.
20. Click Save.

Create First Administrator

1. Go to the Azure Active Directory blade.
2. Select Enterprise applications.
3. Select Users and groups.
4. Click Add user.
5. Select the desired user.
6. Click **Select**.
7. Assign the Administrator role.
8. Click **Select**.
9. Click **Assign**.

These steps can be used to add as many Administrators as required. The User role is used to revoke the Administrator access for the user.

### Add Roles to Service Principal

1. Go to the **Resource groups** blade.
2. Select the newly created Resource group.
3. Select **Access control (IAM)**.
4. Select **Role assignments**.
5. Click **Add**.
6. Set **Owner** as **Role**.
7. Fill in the App name to **Select**.
8. Click **Save**.
9. Click **Add**.
10. Set **User Access Administrator** as **Role**.
11. Fill in the App name to **Select**.
12. Click **Save**.

#### 2.1.3 Runtime Dependencies

After the basic setup of Azure resources is completed, the next step is to set up runtime dependencies for Puddle.

### PostgreSQL Database

Run the following steps to provision the PostgreSQL database.

1. Search for **Azure Database for PostgreSQL servers**.
2. Click **Add**.
3. Fill in the form.
   - Place the PostgreSQL database into the Resource group created in the first step.
   - Set the location of the PostgreSQL database to the location that you specified in the first step.
   - Remember the **Server admin login name** and **Password**. We will need them in later steps.
   - Set the version to **9.6**.
   - Click on **Pricing tier**.
     - Select **Basic**.
     - Set **vCore** to **2 vCores**.
– Set storage to 50GB.
4. Click Create to begin provisioning.

Provisioning of the PostgreSQL database will take a few minutes, but we can continue with other steps.

Redis

Run the following steps to provision Redis.
2. Click Add.
3. Fill in the form.
   • Place the Redis into the Resource group created in first step.
   • Set the location of the Redis to the same location that you specified in the first step.
   • Set Pricing tier to Standard C1.
4. Click Create

Provisioning of the Redis will take a few minutes, but we can continue with other steps.

2.1.4 Puddle Application

For this part, we will need to create a Virtual machine where the Puddle application will run. Then we will configure nginx and create a configuration file for Puddle. After those are complete, we can start Puddle.

Additional PostgreSQL Configuration

1. Search for Azure Database for PostgreSQL servers.
2. Select the newly created database.
3. Select Connection security.
4. Click Add client IP.
5. Use the Public IP of the Virtual machine as the Start IP and End IP.
6. Click Save.

Review the Resource Group

The newly created Resource group should now contain these items (some of them are created implicitly):
• Azure Database for PostgreSQL server
• Azure Cache for Redis
• Virtual machine
• Disk
• Network interface
• Public IP address
• Network security group
• Storage account
• Network security group

Create the License File

1. ssh into the Virtual Machine.
2. Create a file /opt/h2oai/puddle/license.sig containing the license. Different path might be used, but this is the default.

Configuring Puddle

Now we will need to fill in the config.yaml file, which is located at /etc/puddle/config.yaml. The config.yaml should contain the following:

```yaml
redis:
  connection:
    protocol: tcp
    address:
    password:
    tls: true

db:
  connection:
    drivername: postgres
    host:
    port: 5432
    user:
    dbname: puddle
    sslmode: require
    password:

tls:
  certFile:
  keyFile:

license:
  file: /opt/h2oai/puddle/license.sig

ssh:
  publicKey: /opt/h2oai/puddle/ssh/id_rsa.pub
  privateKey: /opt/h2oai/puddle/ssh/id_rsa

auth:
  token:
    secret:
  activeDirectory:
    enabled: false
    server:
    port: 389
    baseDN:
    security: tls
    objectGUIDAttr: objectGUID
    displayNameAttr: displayName
    administratorsGroup: Puddle-Administrators
    usersGroup: Puddle-Users
```
Puddle at H2O Installation and Administration, Release v1.2.0

```yaml
implicitGrant: false
azureAD:
  enabled: false
  useAADLoginExtension: true
awsCognito:
  enabled: false
  userPoolId:
  userPoolWebClientId:
  domain:
  redirectSignIn:
  redirectSignOut:
  adminsGroup: Puddle-Administrators
  usersGroup: Puddle-Users
  implicitGrant: false
ldap:
  enabled: false
  host:
  port: 389
  baseDN:
  baseDNGroup:
  bindDN:
  bindPassword:
  implicitGrant: false
  adminsGroup: Puddle-Administrators
  usersGroup: Puddle-Users
packer:
  path: /opt/h2oai/puddle/deps/packer
  usePublicIP: true
terraform:
  path: /opt/h2oai/puddle/deps/terraform
  usePublicIP: true
backend:
  baseUrl:
    connections:
      usePublicIP: true
webclient:
  usePublicIP: true
providers:
  azure:
    enabled: false
    authority:
    location:
    rg:
    vnetrg:
    vnet:
    sg:
    subnet:
    enterpriseApplicationObjectId:
    adminRoleId:
    publicIpEnabled: true
    packerInstanceType:
  aws:
    enabled: false
```

2.1. Azure VPC Setup Guide
1. ssh into the Virtual machine.

2. Fill in the fields in the config.yaml.
   - Values for redis.connection.* can be found in following way:
     - Microsoft Azure:
       * Search for Azure Cache for Redis.
       * Select newly created Redis instance.
       * Select Access keys.
     - Amazon AWS:
       * Go to ElastiCache Dashboard.
       * Select Redis.
       * Select cluster used by Puddle.
       * Select Description tab.
   - Values for db.connection.* can be found in following way:
     - Microsoft Azure:
       * Search for Azure Database for PostgreSQL servers.
       * Select the newly created PostgreSQL instance.
       * Select Connection strings.
       * Use the password that was provided when creating the PostgreSQL database.
– **Amazon AWS:**

* Go to **Amazon RDS**.
* Select **Databases**.
* Select database used by Puddle.

- `tls.certFile` should point to the PEM encoded **certificate** file if you want to use HTTPS. If you don’t want to use HTTPS, leave this property empty. If you set this property, then `tls.keyFile` must be set as well.

- `tls.keyFile` should point to the PEM encoded **private key** file if you want to use HTTPS. The private key **must be not encrypted by password**. If you don’t want to use HTTPS, leave this property empty. If you set this property, then `tls.certFile` must be set as well.

- `license.file` should be a path to the file containing the license (created in previous step).

- `ssh.publicKey` should be the path to ssh public key (for example `/opt/h2oai/puddle/ssh/id_rsa.pub`), which will be used by Puddle to talk to the Systems. **If this ssh key is changed, Puddle won’t be able to talk to the Systems created with old key, and these will have to be destroyed.**

- `ssh.privateKey` should be the path to ssh private key (for example `/opt/h2oai/puddle/ssh/id_rsa`), which will be used by Puddle to talk to the Systems. **If this ssh key is changed, Puddle won’t be able to talk to the Systems created with old key, and these will have to be destroyed.**

- `auth.token.secret` should be a random string. It is used to encrypt the tokens between the backend and frontend.

- For example the following could be used to generate the secret:

```
tr -cd '[:alnum:]' < /dev/urandom | fold -w32 | head -n1
```

- `auth.activeDirectory.enabled` should be true/false and is false by default. If true then authentication using ActiveDirectory is enabled.

- `auth.activeDirectory.server` should be the hostname of the ActiveDirectory server, for example `puddle-ad.h2o.ai`.

- `auth.activeDirectory.port` should be the port where ActiveDirectory is accessible, defaults to 389.

- `auth.activeDirectory.baseDN` should be the BaseDN used for search.

- `auth.activeDirectory.security` should be the security level used in communication with AD server. Could be none, start_tls, tls, defaults to tls.

- `auth.activeDirectory.objectGUIDAttr` should be the name of the attribute used as ID of the user, defaults to objectGUID.

- `auth.activeDirectory.displayNameAttr` should be the name of the attribute used to determine groups where user is member, defaults to memberOf.

- `auth.activeDirectory.administratorsGroup` should be the name of the Administrators group. Users in this group are assigned Administrator role in Puddle, users in Administrators group and Users group are considered Administrators.

- `auth.activeDirectory.usersGroup` should be the name of the Users group. Users in this group are assigned User role in Puddle, users in Administrators group and Users group are considered Administrators.
• auth.activeDirectory.implicitGrant should be true/false and is false by default. If true, then users are allowed access to Puddle (using user role) even if they are not members of Administrators nor Users group. If false, then users must be members of at least one group to be allowed access to Puddle.

• auth.azureAD.enabled should be true/false and is false by default. If true, then authentication using Azure Active Directory is enabled.

• auth.azureAD.useAADLoginExtension should be true/false and is false by default. If true, then ssh access to provisioned Virtual machines will use the Azure AD for authentication. Check [https://docs.microsoft.com/en-us/azure/virtual-machines/linux/login-using-aad](https://docs.microsoft.com/en-us/azure/virtual-machines/linux/login-using-aad) for more information. Cannot be enabled, if using proxy for egress.

• auth.awsCognito.enabled should be true/false and is false by default. If true, then authentication using AWS Cognito is enabled.

• auth.awsCognito.userPoolId should be the Pool Id, for example us-east-1_SlxxxxML1.

• auth.awsCognito.userPoolWebClientId should be the App client id.
  – The App client id can be found in following way:
    * Go to the AWS Cognito User Pool used by Puddle.
    * Select the **App client settings**.
    * Use the value under **ID**.

• auth.awsCognito.domain should be the domain of the AWS Cognito User Pool, for example [https://puddle.auth.<REGION>.amazoncognito.com](https://puddle.auth.<REGION>.amazoncognito.com).
  – The domain can be found in following way:
    * Go to the AWS Cognito User Pool used by Puddle.
    * Select the **Domain name**.

• auth.awsCognito.redirectSignIn should be the [https://<SERVER_ADDRESS>/aws-cognito-callback](https://<SERVER_ADDRESS>/aws-cognito-callback), please replace <SERVER_ADDRESS> with hostname where Puddle is running.

• auth.awsCognito.redirectSignOut should be [https://<SERVER_ADDRESS>/logout](https://<SERVER_ADDRESS>/logout), please replace <SERVER_ADDRESS> with hostname where Puddle is running.

• auth.awsCognito.adminsGroup should be the name of a group in AWS Cognito User Pool. If users are members of this group, they are assigned Administrator role in Puddle.

• auth.awsCognito.usersGroup should be the name of a group in AWS Cognito User Pool. If users are members of this group, they are assigned User role in Puddle.

• auth.awsCognito.implicitGrant should be true/false and is false by default. If true, then users are allowed access to Puddle (using user role) even if they are not members of Administrators nor Users group. If false, then users must be members of at least one group to be allowed access to Puddle.

• auth.ldap.enabled should be true/false and is false by default. If true, then authentication using LDAP is enabled.

• auth.ldap.host should be the LDAP server hostname.

• auth.ldap.port should be the port where LDAP is accessible, defaults to 389.

• auth.ldap.baseDN should be the BaseDN where authentication search will start.

• auth.ldap.bindDN should be the BindDN used by Puddle to query LDAP.
• auth.ldap.bindPassword should be the password of the user used by Puddle to query LDAP.
• auth.ldap.implicitGrant should be true/false and is false by default. If true, then users are allowed access to Puddle (using user role) even if they are not members of Administrators nor Users group. If false, then users must be members of at least one group to be allowed access to Puddle.
• auth.ldap.adminsGroup should be the name of the Administrators group. Users in this group are assigned Administrator role in Puddle, users in Administrators group and Users group are considered Administrators.
• auth.ldap.usersGroup should be the name of the Users group. Users in this group are assigned User role in Puddle, users in Administrators group and Users group are considered Administrators.
• packer.path should point to the packer binary. Defaults to /opt/h2oai/puddle/deps/packer.
• packer.usePublicIP should be true/false and is true by default. If true then packer will use public IP to communicate with the provisioned Virtual machines, otherwise private IP will be used.
• terraform.path should point to the terraform binary. Defaults to /opt/h2oai/puddle/deps/terraform.
• terraform.usePublicIP should be true/false and is true by default. If true then terraform will use public IP to communicate with the provisioned Virtual machines, otherwise private IP will be used.
• backend.baseUrl should be the URL where Puddle is running, for example https://puddle.h2o.ai.
• backend.connections.usePublicIp should be true/false and is true by default. If true then backend will use public IP to communicate with the provisioned Virtual machines, otherwise private IP will be used.
• webclient.usePublicIp should be true/false and is true by default. If true then public IP is shown in UI, otherwise private IP is displayed.
• providers.azure.enabled should be true/false and is false by default. If true then Microsoft Azure is enabled as provider in Puddle. All variables under providers.azure must be set if enabled.
• providers.azure.authority should be set to https://login.microsoftonline.com/<Azure ActiveDirectory Name>.onmicrosoft.com.
• The Azure Active Directory name can be found in following way:
  – Go to Azure Active Directory blade.
  – Select Overview.
• providers.azure.location should be set to the same value that was specified for the Resource group, for example eastus.
• providers.azure.rg should be set to the name of the newly created Resource group.
• providers.azure.vnetrg should be set to the name of the Resource group where VNET and Subnet are present.
• providers.azure.vnet should be set to the id of the newly created Virtual network.
• providers.azure.sg should be set to the id of the newly created Network security group.
• providers.azure.subnet should be set to the id of the newly created Subnet.
• **providers.azure.enterpriseApplicationObjectId** should be the Object ID of the Enterprise Application.

• The Enterprise Application Object ID can be found in following way:
  – Go to the **Azure Active Directory** blade.
  – Select **Enterprise Applications**.
  – Select the newly created Enterprise Application.
  – Use the **Object ID**.

• **providers.azure.adminRoleId** should be set to the ID of the newly created Administrator Role in the Application Registration Manifest.

• The Administrator Role ID can be found in following way:
  – Go to the **Azure Active Directory** blade.
  – Select **App registrations (preview)**.
  – Select the newly created App registration.
  – Select **Manifest**.
  – Search for **Administrator** role under **appRoles** and use the ID of this role.

• **providers.azure.publicIpEnabled** should be true/false and is true by default. Public IP is created if and only if this is set to true. Must be set to true if at least one of packer, terraform, backend or webclient uses public IP.

• **providers.azure.packerInstanceType** should be the instance type used by Packer to build images. Defaults to Standard_DS2_v2.

• **providers.aws.enabled** should be true/false and is false by default. If true then Amazon AWS is enabled as provider in Puddle. All variables under **providers.aws** must be set if enabled.

• **providers.aws.owner** should be the owner of the newly created resources.

• **providers.aws.vpcId** should be the ID of the VPC where Virtual machines will be launched.

• **providers.aws.sgId** should be the ID of the Security Group applied to provisioned Virtual machines.

• **providers.aws.subnetId** should be the ID of the Subnet where Virtual machines will be placed.

• **providers.aws.publicIpEnabled** should be true/false and is false by default. If true, then no public IP will be assigned. Must be set to true if at least one of packer, terraform, backend or webclient uses public IP.

• **providers.aws.packerInstanceType** should be the instance type used by packer to build images, defaults to m5.large.

• **products.dai.configTomlTemplatePath** should be the path to custom config.toml file, which will be used as default configuration for all new Driverless AI Systems. If not set, the default file is used.

• **products.dai.license** should be the path to DriverlessAI license file. If set, then this license will be automatically installed on all provisioned systems.

• **logs.dir** should be set to a directory where logs should be placed.

• **logs.maxSize** should be the max size of log file, in MB, defaults to 1000.
Puddle at H2O Installation and Administration, Release v1.2.0

• **logs.maxBackups** should be the number of old files retained, defaults to 15.
• **logs.maxAge** should be the max age of retained files, in days, defaults to 60. Older files are always deleted.
• **logs.compress** should be true/false and is true by default. If true then the files will be compressed when rotating.
• **mailing.enabled** should be true/false. If true then mailing is enabled. All fields under mailing are mandatory if this is set to true.
• **mailing.server** should be the hostname and port of the SMTP server, for example smtp.example.com:587.
• **mailing.username** should be the client username.
• **mailing.password** should be the client password.
• **mailing.fromAddress** should be the email address used as FROM, for example in case of an address `<Puddle> puddle@h2o.ai` this field should be set to puddle@h2o.ai.
• **mailing.fromName** should be the name used as FROM, defaults to Puddle, for example in case of an address `<Puddle> puddle@h2o.ai` this field should be set to Puddle.
• **mailing.recipients** should be the space-separated list of recipients.
• **mailing.offsetHours** should be a number of hours between repeated email notifications, defaults to 24, does not apply to FAILED system notifications.

### Configuring Environment Variables

The next step is to fill in the variables in EnvironmentFile file, which is located at `/etc/puddle/EnvironmentFile`. The EnvironmentFile should contain the following:

```bash
# Should point to dir with config.yaml
PUDDLE_CONFIG_DIR='/etc/puddle/

ARM_SKIP_PROVIDER_REGISTRATION=true  # AzureRM Provider should skip registering the Resource Providers

# Azure related environment variables, please fill-in all values if you use Azure as provider
# AZURE_SUBSCRIPTION_ID='YOUR-SUBSCRIPTION-ID'
# AZURE_TENANT_ID='YOUR-TENANT-ID'
# AZURE_CLIENT_ID='YOUR-CLIENT-ID'
# AZURE_CLIENT_SECRET='YOUR-CLIENT-SECRET'

# AWS related environment variables, please fill-in all values if you use AWS as provider
# AWS_ACCESS_KEY_ID='YOUR-AWS-ACCESS-KEY-ID'
# AWS_SECRET_ACCESS_KEY='YOUR-AWS-SECRET-ACCESS-KEY'
# AWS_REGION='AWS-REGION'
```

- **PUDDLE_CONFIG_DIR** directory where the config.yaml file is present.
- **ARM_SKIP_PROVIDER_REGISTRATION** - AzureRM Provider should skip registering the Resource Providers. This should be left as true.
- **AZURE_SUBSCRIPTION_ID** is the ID of the subscription that should be used.
  - This value can be found in following way:

2.1. Azure VPC Setup Guide
• Search for Subscriptions.
• Use the SUBSCRIPTION ID of the subscription you want to use.
  • AZURE_TENANT_ID is ID of tenant that should be used.
    – This value can be found in following way:
      • Select Azure Active Directory blade.
      • Select App registrations (preview).
      • Select the newly created App registration.
      • Use Directory (tenant) ID.
  • AZURE_CLIENT_ID is the Application ID that should be used.
    – This value can be found in following way:
      • Select Azure Active Directory blade.
      • Select App registrations (preview).
      • Select the newly created App registration.
      • Use Application (client) ID.
  • AZURE_CLIENT_SECRET client secret that should be used.
    – This value can be found in following way:
      • Select the Azure Active Directory blade.
      • Select App registrations (preview).
      • Select the newly created App registration.
      • Select Certificates & Secrets.
      • Click New client secret.
      • Fill in the form and click Add.
      • The secret value should be visible. Copy it because after refreshing the page, this value is gone and cannot be restored.
  • AWS_ACCESS_KEY_ID AWS Access Key Id used by Puddle to access the AWS services.
  • AWS_SECRET_ACCESS_KEY AWS Secret Access Key used by Puddle to access the AWS services.
  • AWS_REGION AWS Region used by Puddle to access the AWS services.

Running Puddle

After all of the previous steps are successfully completed, we can now start Puddle. Execute the following command to start the server and web UI:

```
systemctl start puddle
```

Puddle is accessible on port 443 if HTTPS is enabled, or on port 80 if HTTP is being used.
First Steps

At first, you will have to perform some initialization steps:

1. Log in to Puddle as the Administrator.
2. Go to Administration > Check Updates.
3. Either use the update plan from the default URL location, or specify a custom update plan file.
4. Click Submit.
5. Review the plan and click Apply.
6. Go to Administration > Images.
7. Build all the images you want to use. Please be aware this can take up to 1 hour.

Once the images are built, your Puddle instance is ready.

Stats Board (Optional)

The stats board is an optional component. It’s distributed as Python wheel, and it requires Python 3.6. It’s recommended (although not necessary) to run the board inside a virtual environment.

Use the following to install the required dependencies:

```
apt install gcc libpq-dev python3.6-dev python-virtualenv
```

```
yum install epel-release
yum install gcc postgresql-devel python36-devel python-virtualenv
```

Use the following to create the virtualenv:

```
mkdir -p /opt/h2oai/puddle/envs
cd /opt/h2oai/puddle/envs
virtualenv -p python3.6 puddle-stats-env
```

Please make sure that the virtualenv uses the same name and is available at the same path as in this provided snippet. Otherwise the systemd script used to manage Stats Board will not work.

Use the following to install the stats board. Please note that this command will install dependencies as well:

```
source /opt/h2oai/puddle/envs/puddle-stats-env/bin/activate
pip install puddle_stats_board-<VERSION>-py3-none-any.whl
```

Use the following to run the stats board:

```
systemctl start puddle-dashboard
```

The stats board is running on port 8050 and is accessible from Puddle UI at http://<PUDDLE_SERVER_ADDRESS>/board. There is a link in the Administration menu as well.

2.2 AWS Setup Guide

This topic describes how to set up Puddle on Amazon AWS. Note that this is a sample recipe on how to install Puddle, and it is not necessarily the only recipe. VPN ingress/egress rules might need to be customized depending on the user’s needs.
2.2.1 Recipe for Deploying on Public and Private Subnets

This recipe describes how to configure and deploy Puddle on a public subnet with both Redis and a database on private networks.

For quick navigation on AWS, we recommend that you pin the following to your toolbar:

![VPC Configuration](image)

**VPC Configuration**

Create two elastic IPs: one to be used for the NAT gateway and another one for management in case you need to log in to one of the EC2 machines in your network.

1. Go to the VPC Dashboard and click on the “Launch VPC Wizard”
2. Select the following configuration:

   ![Step 1: Select a VPC Configuration](image)

   3. In the configuration menu, enter names for the public/private subnets. Note that you can keep the default values for the IPv4 CIDR blocks and use one of the created Elastic IPs as an “Elastic IP Allocation ID”.

**Note:** The VPC wizard will only allow you to create one public and one private subnet. Once you create a VPC with the wizard, add a new subnet on a different availability zone than the one created previously. This will come in handy when configuring the Amazon RDS module.

**RDS for the Postgres Database**

1. Click on RDS on the navigation toolbar menu and select *Databases* from the left panel.
2. Click *Create Database* and enter the following details:
   - **Type:** PostgreSQL
   - **Version:** 9.6.15-R1
- **Credentials**: Specify a user and a master password. These credentials will be configured in Puddle’s `config.yaml`.
- **Connectivity**: Select the VPC ID created in the previous section, and set **Publicly Accessible** to **No**.

3. Expand the **Additional configuration** section and set the **Initial database name** to a value as shown in the image below:
Other settings are optional and will depend on your specific system configuration. **Note** that there might be fees associated with the type of RDS configuration you chose.

## Set Up an Inbound Rule to Enable a Connection from the Public Subnet to the Database

After the RDS database is up and running, go to **RDS** and then select **Databases** from the left menu. Click on the newly created database.

From here, you can select the security group associated with the database and modify it to include an inbound rule matching the public subnet where the backend server machine will live. (See VPC Security Groups in the image below.)

![VPC Security Groups](image.png)

### ElasticCache for the Redis Cache

1. Click on **ElasticCache** on the navigation toolbar menu and select **Redis** from the left panel.
2. Click on **CREATE** and enter the details as required. Refer to the image below:

![Redis settings](image.png)
3. Create a new subnet and select the private networks from the previously created VPC on the Advanced Redis settings section as shown in the following image:

### Advanced Redis settings

Advanced settings have common defaults set to give you the fastest way to get started. You can modify these now or after your cluster has been created.

- **Multi-AZ with Auto-Failover**: checked
- **Subnet group**: Create new

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>VPC ID</th>
<th>Subnets</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-subnet-group</td>
<td></td>
<td>vpc-07d16b6a4add2a17f</td>
<td>subnet-0954888178b836c48 ca-central-1b 10.0.2.0/24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>subnet-06875bbaa29b3a03 ca-central-1a 10.0.0.0/24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>subnet-02716e0954d2040a ca-central-1a 10.0.1.0/24</td>
</tr>
</tbody>
</table>

### Backend Server Host for Puddle

Spin up a new EC2 instance for an Ubuntu box and run the following commands to get dependencies installed:

#### Linux Packages

The following Linux packages will be used to check connectivity to PostgreSQL and Redis.

```bash
sudo apt-get update
sudo apt-get install -y wget unzip redis-tools postgresql-client

yum install postgresql redis
```

### Confirm and Test the RDS and ElasticCache Installations

1. Look up the primary endpoint for ElasticCache and make sure the following command succeeds and connects to Redis. (Do not use the read only endpoint.)

   ```
   redis-cli -h xxxxxxxxxxxxx.cacl.cache.amazonaws.com
   ```

2. Look up the database endpoint from the RDS menu and run the following:
3. Once connected to postgresql using the previously defined user and password, install the “uuid-ossp” extension as follows:

```
CREATE EXTENSION "uuid-ossp";
```

4. Validate that the extension is created using:

```
select * from pg_extension;
```

**Configure Puddle AWS Authentication**

This section describes a typical configuration of AWS Cognito to authenticate to Puddle.

1. Click on **Cognito** from the navigation toolbar menu.

2. Click on **Manage User Pools**.

3. Click on the **Create a user pool** button at the top right of the page.

4. Give your new user pool a name, e.g. “puddle-users”.
   
   For this example, we will be using the default settings and will create a new app. Please note that you might need to modify these default settings further depending on your needs.

5. Create a new app client by editing the **App clients** sections as shown in the image below:

![App client configuration form](image)

6. Go back to continue editing the user pool and then click on the **Create pool** button at the bottom of the page.

7. On the newly created user pool, select **Users and Groups** on the left pane menu and create two groups as follows:
Note that the two configured group names above will be used for the config parameters “adminsGroup” and “usersGroup” to denote whether users will be treated as admins or non admins on the Puddle system.

Configure AWS Provider

1. Create a new security group to be used for the DAI and H2O-3 instances that Puddle creates with the following inbound rules:

<table>
<thead>
<tr>
<th>Port Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>8888</td>
<td>Jupyter</td>
</tr>
<tr>
<td>9999</td>
<td>REST API</td>
</tr>
<tr>
<td>12345</td>
<td>DAI web app</td>
</tr>
<tr>
<td>54321</td>
<td>H2O-3</td>
</tr>
</tbody>
</table>

2. Capture the subnet ID from where you want the DAI and H2O-3 instances to be installed. Create a new one if needed.

3. Create a new ssh key pair or use an existing pair. This key pair will be used for the ssh connection to the DAI and H2O-3 instances. Make sure to have this key available on the file system where the Puddle backend is running.

Enable Public IPv4 Addressing for the Public Subnet

1. Open the Amazon VPC console and choose Subnets from the navigation pane.
2. Select the public subnet and choose Subnet Actions > Modify auto-assign IP settings.
3. The Enable auto-assign public IPv4 address check box, if selected, requests a public IPv4 address for all DAI instances launched into the selected subnet.

Create the License File

1. ssh into the Virtual Machine.
2. Create a file `/opt/h2oai/puddle/license.sig` containing the license. Different path might be used, but this is the default.

Configuring Puddle

Now we will need to fill in the config.yaml file, which is located at `/etc/puddle/config.yaml`. The config.yaml should contain the following:
redis:
  connection:
    protocol: tcp
    address: 
    password: 
    tls: true

db:
  connection:
    drivername: postgres
    host: 
    port: 5432
    user: 
    dbname: puddle
    sslmode: require
    password: 

tls:
  certFile: 
  keyFile: 

license:
  file: /opt/h2oai/puddle/license.sig

ssh:
  publicKey: /opt/h2oai/puddle/ssh/id_rsa.pub
  privateKey: /opt/h2oai/puddle/ssh/id_rsa

auth:
  token:
    secret: 
  activeDirectory:
    enabled: false
    server:
      port: 389
      baseDN:
      security: tls
      objectGUIDAttr: objectGUID
      displayNameAttr: displayName
      administratorsGroup: Puddle-Administrators
      usersGroup: Puddle-Users
      implicitGrant: false
  azureAD:
    enabled: false
    useAADLoginExtension: true
  awsCognito:
    enabled: false
    userPoolId:
    userPoolWebClientId:
    domain:
    redirectSignIn:
    redirectSignOut:
    adminsGroup: Puddle-Administrators
    usersGroup: Puddle-Users
    implicitGrant: false
  ldap:
    enabled: false
    host:
port: 389
baseDN:
baseDNGroup:
bindDN:
bindPassword:
implicitGrant: false
adminsGroup: Puddle-Administrators
usersGroup: Puddle-Users

packer:
  path: /opt/h2oai/puddle/deps/packer
  usePublicIP: true

terraform:
  path: /opt/h2oai/puddle/deps/terraform
  usePublicIP: true

backend:
  baseUrl:
    connections:
      usePublicIP: true

webclient:
  usePublicIP: true

providers:
  azure:
    enabled: false
    authority:
      location:
      rg:
      vnetrg:
      vnet:
      sg:
      subnet:
      enterpriseApplicationObjectId:
      adminRoleId:
      publicIpEnabled: true
      packerInstanceType:

  aws:
    enabled: false
    owner:
    vpcId:
    sgId:
    subnetId:
    publicIpEnabled: true
    packerInstanceType:

products:
  dai:
    configTomlTemplatePath:
    license:

logs:
  dir: /opt/h2oai/puddle/logs
  maxSize: 1000
  maxBackups: 15
  maxAge: 60
1. ssh into the Virtual machine.
2. Fill in the fields in the config.yaml.
   • Values for redis.connection.* can be found in following way:
     - Microsoft Azure:
       * Search for Azure Cache for Redis.
       * Select newly created Redis instance.
       * Select Access keys.
     - Amazon AWS:
       * Go to ElastiCache Dashboard.
       * Select Redis.
       * Select cluster used by Puddle.
       * Select Description tab.
   • Values for db.connection.* can be found in following way:
     - Microsoft Azure:
       * Search for Azure Database for PostgreSQL servers.
       * Select the newly created PostgreSQL instance.
       * Select Connection strings.
       * Use the password that was provided when creating the PostgreSQL database.
     - Amazon AWS:
       * Go to Amazon RDS.
       * Select Databases.
       * Select database used by Puddle.
   • tls.certFile should point to the PEM encoded certificate file if you want to use HTTPS. If you don’t want to use HTTPS, leave this property empty. If you set this property, then tls.keyFile must be set as well.
   • tls.keyFile should point to the PEM encoded private key file if you want to use HTTPS. The private key must be not encrypted by password. If you don’t want to use HTTPS, leave this property empty. If you set this property, then tls.certFile must be set as well.
   • license.file should be a path to the file containing the license (created in previous step).
• `ssh.publicKey` should be the path to ssh public key (for example /opt/h2oai/puddle/ssh/id_rsa.pub), which will be used by Puddle to talk to the Systems. If this ssh key is changed, Puddle won’t be able to talk to the Systems created with old key, and these will have to be destroyed.

• `ssh.privateKey` should be the path to ssh private key (for example /opt/h2oai/puddle/ssh/id_rsa), which will be used by Puddle to talk to the Systems. If this ssh key is changed, Puddle won’t be able to talk to the Systems created with old key, and these will have to be destroyed.

• `auth.token.secret` should be a random string. It is used to encrypt the tokens between the backend and frontend.

• for example the following could be used to generate the secret:

```
tr -cd '[:alnum:]' < /dev/urandom | fold -w32 | head -n1
```

• `auth.activeDirectory.enabled` should be true/false and is false by default. If true then authentication using ActiveDirectory is enabled.

• `auth.activeDirectory.server` should be the hostname of the ActiveDirectory server, for example puddle-ad.h2o.ai.

• `auth.activeDirectory.port` should be the port where ActiveDirectory is accessible, defaults to 389.

• `auth.activeDirectory.baseDN` should be the BaseDN used for search.

• `auth.activeDirectory.security` should be the security level used in communication with AD server. Could be none, start_tls, tls, defaults to tls.

• `auth.activeDirectory.objectGUIDAttr` should be the name of the attribute used as ID of the user, defaults to objectGUID.

• `auth.activeDirectory.displayNameAttr` should be the name of the attribute used to determine groups where user is member, defaults to memberOf.

• `auth.activeDirectory.administratorsGroup` should be the name of the Administrators group. Users in this group are assigned Administrator role in Puddle, users in Administrators group and Users group are considered Administrators.

• `auth.activeDirectory.usersGroup` should be the name of the Users group. Users in this group are assigned User role in Puddle, users in Administrators group and Users group are considered Administrators.

• `auth.activeDirectory.implicitGrant` should be true/false and is false by default. If true, then users are allowed access to Puddle (using user role) even if they are not members of Administrators nor Users group. If false, then users must be members of at least one group to be allowed access to Puddle.

• `auth.azureAD.enabled` should be true/false and is false by default. If true, then authentication using Azure Active Directory is enabled.

• `auth.azureAD.useAADLoginExtension` should be true/false and is false by default. If true, then ssh access to provisioned Virtual machines will use the Azure AD for authentication. Check https://docs.microsoft.com/en-us/azure/virtual-machines/linux/login-using-aad for more information. Cannot be enabled, if using proxy for egress.

• `auth.awsCognito.enabled` should be true/false and is false by default. If true, then authentication using AWS Cognito is enabled.

• `auth.awsCognito.userPoolId` should be the Pool Id, for example us-east-1_SLxxxxML1.
• **auth.awsCognito.userPoolWebClientId** should be the App client id.
  
  – The App client id can be found in following way:
    * Go to the AWS Cognito User Pool used by Puddle.
    * Select the **App client settings**.
    * Use the value under **ID**.

• **auth.awsCognito.domain** should be the domain of the AWS Cognito User Pool, for example https://puddle.auth.<REGION>.amazoncognito.com.
  
  – The domain can be found in following way:
    * Go to the AWS Cognito User Pool used by Puddle.
    * Select the **Domain name**.

• **auth.awsCognito.redirectSignIn** should be https://<SERVER_ADDRESS>/aws-cognito-callback, please replace <SERVER_ADDRESS> with hostname where Puddle is running.

• **auth.awsCognito.redirectSignOut** should be https://<SERVER_ADDRESS>/logout, please replace <SERVER_ADDRESS> with hostname where Puddle is running.

• **auth.awsCognito.adminsGroup** should be the name of a group in AWS Cognito User Pool. If users are members of this group, they are assigned Administrator role in Puddle.

• **auth.awsCognito.usersGroup** should be the name of a group in AWS Cognito User Pool. If users are members of this group, they are assigned User role in Puddle.

• **auth.awsCognito.implicitGrant** should be true/false and is false by default. If true, then users are allowed access to Puddle (using user role) even if they are not members of Administrators nor Users group. If false, then users must be members of at least one group to be allowed access to Puddle.

• **auth.ldap.enabled** should be true/false and is false by default. If true, then authentication using LDAP is enabled.

• **auth.ldap.host** should be the LDAP server hostname.

• **auth.ldap.port** should be the port where LDAP is accessible, defaults to 389.

• **auth.ldap.baseDN** should be the BaseDN where authentication search will start.

• **auth.ldap.baseDNGroup** should be the BaseDN where search for user’s group will start.

• **auth.ldap.bindDN** should be the BindDN used by Puddle to query LDAP.

• **auth.ldap.bindPassword** should be the password of the user used by Puddle to query LDAP.

• **auth.ldap.implicitGrant** should be true/false and is false by default. If true, then users are allowed access to Puddle (using user role) even if they are not members of Administrators nor Users group. If false, then users must be members of at least one group to be allowed access to Puddle.

• **auth.ldap.adminsGroup** should be the name of the Administrators group. Users in this group are assigned Administrator role in Puddle, users in Administrators group and Users group are considered Administrators.

• **auth.ldap.usersGroup** should be the name of the Users group. Users in this group are assigned User role in Puddle, users in Administrators group and Users group are considered Administrators.

• **packer.path** should point to the packer binary. Defaults to /opt/h2oai/puddle/deps/packer.
- `packer.usePublicIP` should be true/false and is true by default. If true then `packer` will use public IP to communicate with the provisioned Virtual machines, otherwise private IP will be used.

- `terraform.path` should point to the `terraform` binary. Defaults to `/opt/h2oai/ puddle/deps/terraform`.

- `terraform.usePublicIP` should be true/false and is true by default. If true then `terraform` will use public IP to communicate with the provisioned Virtual machines, otherwise private IP will be used.

- `backend.baseUrl` should be the URL where Puddle is running, for example `https://puddle.h2o.ai`.

- `backend.connections.usePublicIp` should be true/false and is true by default. If true then `backend` will use public IP to communicate with the provisioned Virtual machines, otherwise private IP will be used.

- `webclient.usePublicIp` should be true/false and is true by default. If true then public IP is shown in UI, otherwise private IP is displayed.

- `providers.azure.enabled` should be true/false and is false by default. If true then Microsoft Azure is enabled as provider in Puddle. All variables under `providers.azure` must be set if enabled.

- `providers.azure.authority` should be set to `https://login.microsoftonline.com/<Azure ActiveDirectory Name>.onmicrosoft.com`.

- The Azure Active Directory name can be found in following way:
  - Go to the **Azure Active Directory** blade.
  - Select **Overview**.

- `providers.azure.location` should be set to the same value that was specified for the Resource group, for example `eastus`.

- `providers.azure.rg` should be set to the name of the newly created Resource group.

- `providers.azure.vnetrsg` should be set to the name of the Resource group where VNET and Subnet are present.

- `providers.azure.vnet` should be set to the id of the newly created Virtual network.

- `providers.azure.sg` should be set to the id of the newly created Network security group.

- `providers.azure.subnet` should be set to the id of the newly created Subnet.

- `providers.azure.enterpriseApplicationObjectId` should be the Object ID of the Enterprise Application.

- The Enterprise Application Object ID can be found in following way:
  - Go to the **Azure Active Directory** blade.
  - Select **Enterprise Applications**.
  - Select the newly created Enterprise Application.
  - Use the **Object ID**.

- `providers.azure.adminRoleId` should be set to the ID of the newly created Administrator Role in the Application Registration Manifest.

- The Administrator Role ID can be found in following way:
  - Go to the **Azure Active Directory** blade.

---

2.2. AWS Setup Guide 33
– Select App registrations (preview).
– Select the newly created App registration.
– Select Manifest.
– Search for Administrator role under appRoles and use the ID of this role.

• providers.azure.publicIpEnabled should be true/false and is true by default. Public IP is created if and only if this is set to true. Must be set to true if at least one of packer, terraform, backend or webclient uses public IP.
• providers.azure.packerInstanceType should be the instance type used by Packer to build images. Defaults to Standard_DS2_v2.
• providers.aws.enabled should be true/false and is false by default. If true then Amazon AWS is enabled as provider in Puddle. All variables under providers.aws must be set if enabled.
• providers.aws.owner should be the owner of the newly created resources.
• providers.aws.vpcId should be the ID of the VPC where Virtual machines will be launched.
• providers.aws.sgId should be the ID of the Security Group applied to provisioned Virtual machines.
• providers.aws.subnetId should be the ID of the Subnet where Virtual machines will be placed.
• providers.aws.publicIpEnabled should be true/false and is false by default. If true, then no public IP will be assigned. Must be set to true if at least one of packer, terraform, backend or webclient uses public IP.
• providers.aws.packerInstanceType should be the instance type used by packer to build images, defaults to m5.large.
• products.dai.configTomlTemplatePath should be the path to custom config.toml file, which will be used as default configuration for all new Driverless AI Systems. If not set, the default file is used.
• products.dai.license should be the path to DriverlessAI license file. If set, then this license will be automatically installed on all provisioned systems.
• logs.dir should be set to a directory where logs should be placed.
• logs.maxSize should be the max size of log file, in MB, defaults to 1000.
• logs.maxBackups should be the number of old files retained, defaults to 15.
• logs.maxAge should be the max age of retained files, in days, defaults to 60. Older files are always deleted.
• logs.compress should be true/false and is true by default. If true then the files will be compressed when rotating.
• mailing.enabled should be true/false. If true then mailing is enabled. All fields under mailing are mandatory if this is set to true.
• mailing.server should be the hostname and port of the SMTP server, for example smtp.example.com:587.
• mailing.username should be the client username.
• mailing.password should be the client password.
- `mailing.fromAddress` should be the email address used as FROM, for example in case of an address `<Puddle> puddle@h2o.ai` this field should be set to `puddle@h2o.ai`.
- `mailing.fromName` should be the name used as FROM, defaults to Puddle, for example in case of an address `<Puddle> puddle@h2o.ai` this field should be set to Puddle.
- `mailing.recipients` should be the space-separated list of recipients.
- `mailing.offsetHours` should be a number of hours between repeated email notifications, defaults to 24, does not apply to FAILED system notifications.

### Configuring Environment Variables

The next step is to fill in the variables in `EnvironmentFile` file, which is located at `/etc/puddle/EnvironmentFile`. The `EnvironmentFile` should contain the following:

```bash
# Should point to dir with config.yaml
PUDDLE_CONFIG_DIR='/etc/puddle/

ARM_SKIP_PROVIDER_REGISTRATION=true  # AzureRM Provider should skip registering the Resource Providers

# Azure related environment variables, please fill-in all values if you use Azure as provider
# AZURE_SUBSCRIPTION_ID='YOUR-SUBSCRIPTION-ID'
# AZURE_TENANT_ID='YOUR-TENANT-ID'
# AZURE_CLIENT_ID='YOUR-CLIENT-ID'
# AZURE_CLIENT_SECRET='YOUR-CLIENT-SECRET'

# AWS related environment variables, please fill-in all values if you use AWS as provider
# AWS_ACCESS_KEY_ID='YOUR-AWS-ACCESS-KEY-ID'
# AWS_SECRET_ACCESS_KEY='YOUR-AWS-SECRET-ACCESS-KEY'
# AWS_REGION='AWS-REGION'
```

- **PUDDLE_CONFIG_DIR directory** where the `config.yaml` file is present.
- **ARM_SKIP_PROVIDER_REGISTRATION** - AzureRM Provider should skip registering the Resource Providers. This should be left as true.
- **AZURE_SUBSCRIPTION_ID** is the ID of the subscription that should be used.
  - This value can be found in following way:
    - Search for Subscriptions.
    - Use the SUBSCRIPTION ID of the subscription you want to use.
- **AZURE_TENANT_ID** is ID of tenant that should be used.
  - This value can be found in following way:
    - Select Azure Active Directory blade.
    - Select App registrations (preview).
    - Select the newly created App registration.
    - Use Directory (tenant) ID.
- **AZURE_CLIENT_ID** is the Application ID that should be used.
  - This value can be found in following way:
• Select Azure Active Directory blade.
• Select App registrations (preview).
• Select the newly created App registration.
• Use Application (client) ID.

• AZURE_CLIENT_SECRET client secret that should be used.
  – This value can be found in following way:
    • Select the Azure Active Directory blade.
    • Select App registrations (preview).
    • Select the newly created App registration.
    • Select Certificates & Secrets.
    • Click New client secret.
    • Fill in the form and click Add.
    • The secret value should be visible. Copy it because after refreshing the page, this value is gone and cannot be restored.

• AWS_ACCESS_KEY_ID AWS Access Key Id used by Puddle to access the AWS services.
• AWS_SECRET_ACCESS_KEY AWS Secret Access Key used by Puddle to access the AWS services.
• AWS_REGION AWS Region used by Puddle to access the AWS services.

Running Puddle

After all of the previous steps are successfully completed, we can now start Puddle. Execute the following command to start the server and web UI:

```
systemctl start puddle
```

Puddle is accessible on port 443 if HTTPS is enabled, or on port 80 if HTTP is being used.

First Steps

At first, you will have to perform some initialization steps:

1. Log in to Puddle as the Administrator.
2. Go to Administration > Check Updates.
3. Either use the update plan from the default URL location, or specify a custom update plan file.
4. Click Submit.
5. Review the plan and click Apply.
6. Go to Administration > Images.
7. Build all the images you want to use. Please be aware this can take up to 1 hour.

Once the images are built, your Puddle instance is ready.
Stats Board (Optional)

The stats board is an optional component. It’s distributed as Python wheel, and it requires Python 3.6. It’s recommended (although not necessary) to run the board inside a virtual environment.

Use the following to install the required dependencies:

```
apt install gcc libpq-dev python3.6-dev python-virtualenv
```

```
yum install epel-release
yum install gcc postgresql-devel python36-devel python-virtualenv
```

Use the following to create the virtualenv:

```
mkdir -p /opt/h2oai/puddle/envs
cd /opt/h2oai/puddle/envs
virtualenv -p python3.6 puddle-stats-env
```

Please make sure that the virtualenv uses the same name and is available at the same path as in this provided snippet. Otherwise the systemd script used to manage Stats Board will not work.

Use the following to install the stats board. Please note that this command will install dependencies as well:

```
source /opt/h2oai/puddle/envs/puddle-stats-env/bin/activate
pip install puddle_stats_board-<VERSION>-py3-none-any.whl
```

Use the following to run the stats board:

```
systemctl start puddle-dashboard
```

The stats board is running on port 8050 and is accessible from Puddle UI at http://<PUDDLE_SERVER_ADDRESS>/board. There is a link in the Administration menu as well.
Navigate to your Puddle URL, and sign in using one of the configured authentication providers. Upon success, the My Systems page displays and shows any current systems. From this page, you can create a new Driverless AI and/or H2O-3 system. Refer to the Puddle User Guide for information about how to create new systems.

### 3.1 Logging Out

Click on your username in the upper-right corner to log out of Puddle. You will then have the option to log out of the cloud environment as well.
Uses with Admin status are provided with an Administrator dropdown menu. From here, Admins can manage systems, experiments, and users. They can also define system types and add new Driveless AI or H2O-3 images. Finally, Admins can also check for Puddle updates and review logs.

4.1 Systems

The All Systems page lists all systems that have been created in your cloud environment. From here, Admins can start (_starts), stop (stops), reboot (reboots), and destroy (destroys) systems. Admins can click on a system name to view a system and click on a URL for a running system to view that system.

The information that displays for each system includes:

- Name
- Owner
- Cost
- Status
- Type
- Version
- Product
• Created date

### 4.2 Experiments

The **All Experiments** shows all Driverless AI experiments that have been run through Puddle. The information that displays for each experiments includes:

- System
- User
- Description (experiment name)
- Dataset used for training
- Target column
- Score
- Test Score
- Scorer used in the experiment
- Progress
- Status
- Accuracy setting used in the experiment
- Time setting used in the experiment
- Interpretability setting used in the experiment
- Training Duration
4.3 Tags

Puddle allows Admins to set up tags that can be applied to specific system. The Tags page shows all tags that can be specified when provisioning a new system along with the following information for each tag:

- Label
- Input Type
- Default Value
- Required (Yes or No)
- Date when the tag was created
- Date when the tag was last updated

4.3.1 Creating New Tags

Perform the following steps to create a new tag. After new tags are created, they will automatically be available when users create new systems.

1. On the Tags page, click the Create New Tag button.
2. Specify a label for the tag.
3. Specify the tag input type. This can be one of the following:
   - Text
   - Email
• Number
• Date
• Checkbox

4. Specify a default value for the input type.
5. Specify whether the label is required.
6. Click Submit when you are done.

### 4.3.2 Editing Tags

On the Tags page, click the Edit icon ( ) beside the tag that you want to delete. Edit the Input Type, Default Value, and/or Required checkbox, then click Submit to complete the delete. Note that currently tagged systems are not affected when changes are made to tags.

### 4.3.3 Deleting Tags

On the Tags page, click the Delete icon ( ) beside the tag that you want to delete. A confirmation page will display informing you that current systems will not be affected by this deletion. Click Yes to complete the deletion.

### 4.4 Users

The All Users page shows all users who have access to Puddle. From this page, Admins can lock users, unlock users, and edit system limits for each user. This page also shows the following for each user:

• Name
• Email address
• Role (Note that this role comes from the user’s appRole in the AD.)
• Status
• Current Spending
• Date when user was added
• Date when user was last updated
4.4.1 Locking/Unlock Users

The Unlock icon (▲) indicates that a user is able to log in to Puddle. Click on this icon to lock that user—locked users will not be able to log in to Puddle. If you lock a user who is currently logged in, that user’s session will be invalidated. The user’s systems, however, will not be deleted. Systems can be deleted manually from the All Systems page. (See Systems.)

The Lock icon (▼) indicates that a user is locked out of Puddle. Click on this icon to unlock that user—unlocked users will be able to log in to Puddle.

Note: Admins cannot lock/unlock themselves. The lock/unlock icons will not be available for the logged in Admin.

4.4.2 Editing User’s System Limits

Perform the following steps to edit a user’s system limits. Note that these limits are per system. So, for example, if the GPU Compute limit is set to 1, then users can create at most one of each a Driverless AI and a H2O-3 GPU Compute system.

1. On the All Users page, click the Edit icon (▶) beside a user to edit instance limits for that user for each system type. On the form that displays, you can specify limits for the following:
   • CPU Small systems. Defaults to 10. Disable the Use Default checkbox to set away from the default value.
   • CPU Large systems. Defaults to 1. Disable the Use Default checkbox to set away from the default value.
   • GPU Small systems. Defaults to 1. Disable the Use Default checkbox to set away from the default value.
   • GPU Memory systems. Defaults to 1. Disable the Use Default checkbox to set away from the default value.
   • GPU Compute systems. Defaults to 1. Disable the Use Default checkbox to set away from the default value.
2. Click Submit when you are done or Cancel to leave the form without applying any changes.
4.5 System Types

The System Types page shows all system types that are available in Puddle. The available types are defined during installation; however, this page allows you to lock a system type or edit the default limit and label for a system type. The information that displays on this page includes the following:

- Cloud Identifier
- Default Limit
- Label for the identifier
- Whether the system type is GPU
- Number of Compute Units for the system type
- Price per hour
- Created date
- Updated date
4.5.1 Locking/Unlocking System Types

The **Unlock** icon (علامة المطلقة) indicates that a system type is unlocked. Click on this icon to lock a system type. When it is locked, users will not be able to create new systems with the locked system type. Existing systems with this system type will not be affected.

The **Lock** icon (امة المطلقة) indicates that a system type is locked. Click on this icon to unlock a system type. When it is unlocked, users will once again be able to create new systems with the unlocked system type.

4.5.2 Editing a System Type

Click the **Edit** icon (امة المطلقة) beside a system type to edit that system’s label and maximum default limit. Click **Submit** when you are done or **Cancel** to return the **All Systems** page without applying any changes.
4.6 Images

The Images page shows all Driverless AI and H2O-3 images available in your environment. The information that displays on this page for each image includes the following:

- Version
- Product
- Status
- Cloud Identifier
- Text (product version)
- Created date
- Updated date

<table>
<thead>
<tr>
<th>Version</th>
<th>Product</th>
<th>Status</th>
<th>Cloud Identifier</th>
<th>Text</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>DAI</td>
<td>Available</td>
<td>/subscriptions/92429150-401b-4...</td>
<td>16.2</td>
<td>5/13/2019 14:34 AM</td>
<td>5/13/2019 21:55 AM</td>
</tr>
<tr>
<td>1.1</td>
<td>DAI</td>
<td>Available</td>
<td>/subscriptions/92429150-401b-4...</td>
<td>16.1</td>
<td>4/30/2019 9:20 AM</td>
<td>4/30/2019 12:35 PM</td>
</tr>
<tr>
<td>1.5</td>
<td>DAI</td>
<td>Available</td>
<td>/subscriptions/92429150-401b-4...</td>
<td>15.4</td>
<td>4/30/2019 9:20 AM</td>
<td>5/22/2019 7:15 PM</td>
</tr>
<tr>
<td>2.0</td>
<td>DAI</td>
<td>Building...</td>
<td>/subscriptions/92429150-401b-4...</td>
<td>17.0</td>
<td>8/21/2019 6:54 AM</td>
<td>8/21/2019 7:29 AM</td>
</tr>
<tr>
<td>1.5</td>
<td>DAI</td>
<td>Not yet built</td>
<td>/subscriptions/92429150-401b-4...</td>
<td>16.3</td>
<td>8/21/2019 6:54 AM</td>
<td>8/21/2019 6:54 AM</td>
</tr>
</tbody>
</table>
4.6.1 Locking/Unlocking Images

The Unlock icon (Unlock) indicates that an image is unlocked. Click on this icon to lock an image type. When it is locked, users will not be able to create new systems with the locked image type. Existing systems with this image type will not be affected.

The Lock icon (Lock) indicates that an image is locked. Click on this icon to unlock an image type. When it is unlocked, users will once again be able to create new systems with the unlocked image type.

4.6.2 Deleting Images

Click the Delete icon (Delete) beside a locked image to delete that image. Note that unlocked images cannot be deleted.

4.7 Check Updates

The Check for Updates page allows you to verify whether you have the latest version of Puddle.

1. Specify the URL for your Puddle image or click Browse to locate an image on your local machine.

2. Click Submit when you are done. After the image is located, the Review Update Plan page displays. This page shows images and system types that are available on your Puddle.

3. Review the images and system types that are available.

   Note: If desired, you can delete any of the available images or system types from this page.

4. Click Apply to apply the update or Cancel to cancel the request.
4.8 Stats Board

The **Stats Board** provides admins with detailed information about the number of new users, systems, experiments, sessions, and money spent for the current day, current week, current month, last 7 days, last 30 days, and overall.

The lower section of the page includes interactive [Plotly](https://plotly.com) charts for the number of Experiments and Spending totals (in US dollars) on your system.
4.9 Logs

The Logs page allows you to view/and or download puddle (full and daily) and stats logs.

<table>
<thead>
<tr>
<th>Logs</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>puddle-2019-04-26T15:01-06:769.log.gz</td>
<td>1.73MB</td>
</tr>
<tr>
<td>puddle.log</td>
<td>36.24MB</td>
</tr>
<tr>
<td>stats.log</td>
<td>1.86MB</td>
</tr>
</tbody>
</table>

4.10 About

The About page provides the current build information including the Puddle version, build number, git commit, build time, built by system, and built on system.
### About

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.1.0</td>
</tr>
<tr>
<td>Build Number</td>
<td>2</td>
</tr>
<tr>
<td>Git Commit</td>
<td>5a9d6e9</td>
</tr>
<tr>
<td>Build Time</td>
<td>2019/08/21 14:59:30 UTC</td>
</tr>
<tr>
<td>Built By</td>
<td>jenkins</td>
</tr>
<tr>
<td>Built On</td>
<td>mr-8xc10</td>
</tr>
</tbody>
</table>

---

*Puddle at H2O Installation and Administration, Release v1.2.0*

Chapter 4. Administration