TasksApp: Deploying a PHP and MySQL app in Azure App Service

Introduction:

Azure App Service provides a highly scalable, self-patching web hosting service using the Linux operating system. In this project, I have explored two Azure Services: **Azure App Service** and **Azure Database for MySQL Server**. I have deployed a basic PHP Laravel App in Azure and connected it to a MySQL Database in Azure. I have used Local Git as the deployment source. The app is a basic Tasks App(a To-Do list app) that can be used to add, delete and display tasks persistently.

```
Source Code: https://github.com/ApurvPurohit/TasksApp
Deployed WebApp: https://tasksapp.azurewebsites.net/
```

Outline

1. Create a MySQL database in Azure.
2. Connect a PHP App to MySQL.
3. Deploy the app to Azure.
4. Update the data model locally and redeploy the app on Azure.

Configuration

```
OS: Ubuntu 18.04.5 LTS
Memory: 7.7 GiB
Processor: Intel(r) CoreTM i7-7700HQ CPU @ 2.80GHz × 8
OS Type: 64-bit
```

App Service Plan

Resource group (change)	: myResourceGroup	App Service Plan	: myAppServicePlan (F1: Free)
Status	: Ready	App(s) / Slots	: 1 / 0
Location	: East Asia	Operating System	: Linux
Subscription (change)	: Azure for Students		
Subscription ID	: c4c19fa2-a20a-473e-85e1-b959e20ea883		
Tags (change)	: Click here to add tags		

Prerequisites

- Git:

```
$sudo apt install git-all
```

- PHP (5.6.4 or above):

```
$sudo apt install php libapache2-mod-php
$sudo systemctl restart apache2
```

- Composer:

```
$sudo apt update
$curl -sS https://getcomposer.org/installer -o composer-setup.php
$HASH=544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a858088ffbc1f233e9b180f061
$php -r "if (hash_file('SHA384', 'composer-setup.php') === '$HASH') { echo 'Installer verified'; } else
{ echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
$sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

- Enable the following PHP extensions Laravel needs: OpenSSL, PDO-MySQL, Mbstring, Tokenizer, XML:

```
$sudo apt install openssl php-common php-curl php-json php-mbstring php-mysql php-xml php-zip
```

Version Specific Installation(E.g. php7.4)

```
$sudo apt install php7.4-common php7.4-bcmath openssl php7.4-json php7.4-mbstring
```

- MySQL:

```
$sudo apt update
$sudo apt install mysql-server
$sudo mysql_secure_installation
```

- Azure Cloud Shell, Azure CLI can also be installed locally to run CLI reference commands.

Workflow

- ### Preparing local MySQL

Connecting to our local MySQL server

```
$mysql -u root -p
```

Database creation

```
mysql>CREATE DATABASE sampledb;
mysql>quit
```

- ### Creating a PHP app locally

Cloning the sample PHP app and installing Composer in the repository root

```
$git clone https://github.com/Azure-Samples/laravel-tasks
$cd laravel-tasks
$composer install
```

Configuring MySQL connection

In the repository root, create a file named *.env* for the environment variables of the connection. Replace the *<root_password>* placeholder with the MySQL root user's password.

```
APP_ENV=local
APP_DEBUG=true
APP_KEY=

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_DATABASE=sampledb
DB_USERNAME=root
DB_PASSWORD=<root_password>
```

Run the sample locally

```
$php artisan migrate
$php artisan key:generate
$php artisan serve
```

The sample will be deployed locally if all the above commands are executed normally. Navigate to http://localhost:8000 to verify deployment.

- **Creating a MySQL Server in Azure**

Create a resource group(Azure CLI)

```
$az group create --name myResourceGroup --location "East Asia"
```

Create a MySQL server(Azure CLI)

```
$az mysql server create --resource-group myResourceGroup --name <mysql-server-name> --location "East Asia" --admin-user <admin-user> --admin-password <admin-password> --sku-name B_Gen5_1
```

Configure server firewall(Azure CLI)

```
$az mysql server firewall-rule create --name allAzureIPs --server <mysql-server-name> --resource-group myResourceGroup --start-ip-address 0.0.0.0 --end-ip-address 0.0.0.0
$az mysql server firewall-rule create --name AllowLocalClient --server <mysql-server-name> --resource-group myResourceGroup --start-ip-address=<your-ip-address> --end-ip-address=<your-ip-address>
```

Connect to production MySQL server locally (Local)

```
$mysql -u <admin-user>@<mysql-server-name> -h <mysql-server-name>.mysql.database.azure.com -P 3306 -p
```

Create a production database(Local)

```
mysql>CREATE DATABASE sampledb;
mysql>CREATE USER 'phpappuser' IDENTIFIED BY 'MySQLAzure2017';
mysql>GRANT ALL PRIVILEGES ON sampledb.* TO 'phpappuser';
mysql>quit
```

- **Connecting the app to Azure MySQL**

Configure the database connection(Local)

```
$php artisan migrate --env=production --force
$php artisan key:generate --env=production --force
$php artisan serve --env=production
```

Commit changes(Local)

```
$git add .
$git commit -m "database.php updates"
```

- **Deploying to Azure**

Configure a deployment user(Azure CLI)

```
$az webapp deployment user set --user-name <username> --password <password>
```

>> Local git is configured with url of https://apurohit@tasksapp.scm.azurewebsites.net/TasksApp.git

Create an App Service plan(Azure CLI)

```
$az appservice plan create --name myAppServicePlan --resource-group myResourceGroup --sku F1 --is-linux
```

Create a web app(Azure CLI)

```
$az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name <app-name> --runtime
"PHP|7.2" --deployment-local-git
```

Configure database settings(Azure CLI)

```
$az webapp config appsettings set --name <app-name> --resource-group myResourceGroup --settings
DB_HOST="<mysql-server-name>.mysql.database.azure.com" DB_DATABASE="sampledb"
DB_USERNAME="phpappuser@<mysql-server-name>" DB_PASSWORD="MySQLAzure2017" MYSQL_SSL="true"
```

Configure Laravel environment variables

Generate new application key(Local)

```
$php artisan key:generate --show
```

Configure environment variable APP_KEY(Azure CLI)

```
$az webapp config appsettings set --name <app-name> --resource-group myResourceGroup --settings
APP_KEY="<output_of_php_artisan_key:generate>" APP_DEBUG="true"
```

Push to Azure from Git(Local)

```
$git remote add azure <deploymentLocalGitUrl-from-create-step>
$git push azure master
```

Browse to http://<app-name>.azurewebsites.net

https://tasksapp.azurewebsites.net/

Settings Logs Local Git/FTPS credentials

Deploy and build code from your preferred source and build provider. Learn more

Source	Local Git
	Disconnect

Local Git

Git Clone Uri	https://tasksapp.scm.azurewebsites.net:443/TasksApp.git

Build

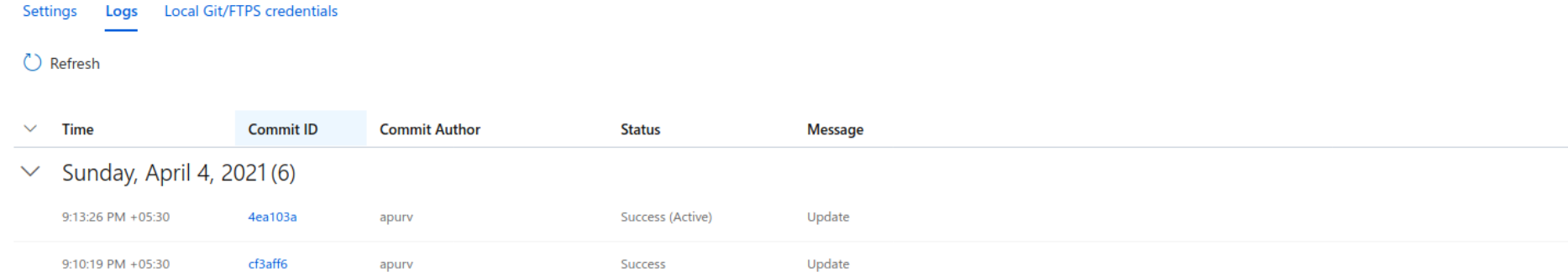
Build provider	App Service Build Service
Runtime stack	PHP
Version	PHP 7.2

- **Update model locally and redeploy**

For future updates and changes in the model, we can modify the source code locally, also test locally and then push the changes to redeploy the updated version.

Testing Deployment

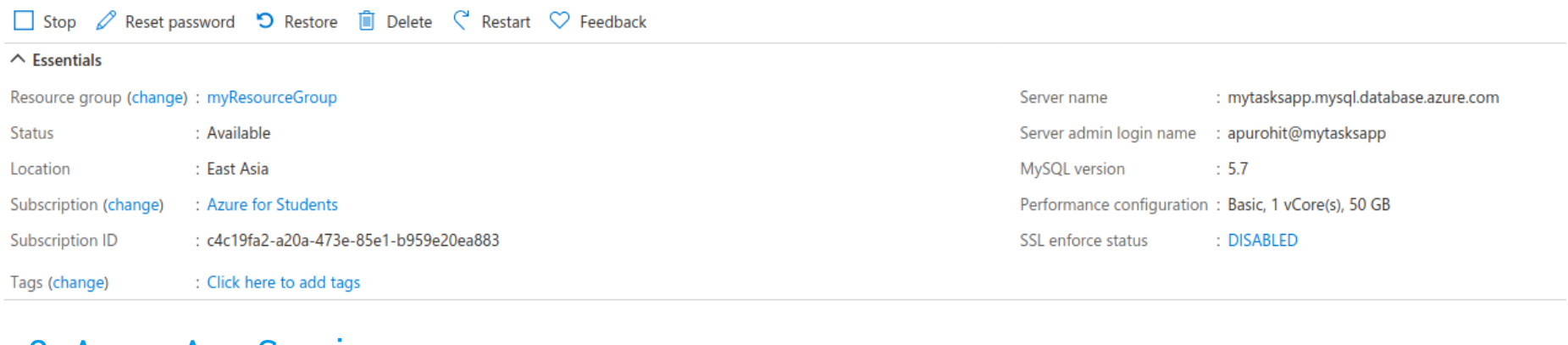
Verifying and testing our deployment is a crucial part of the whole process. To handle this, Azure provides the **Deployment Logs** at several points so that we can verify if everything worked fine or there was Error in deployment. When we push to Azure from our local Git, the execution is always terminated by the particular Deployment Logs which can be viewed easily. We can also view the Logs in the **Deployment Center** in the App Dashboard.



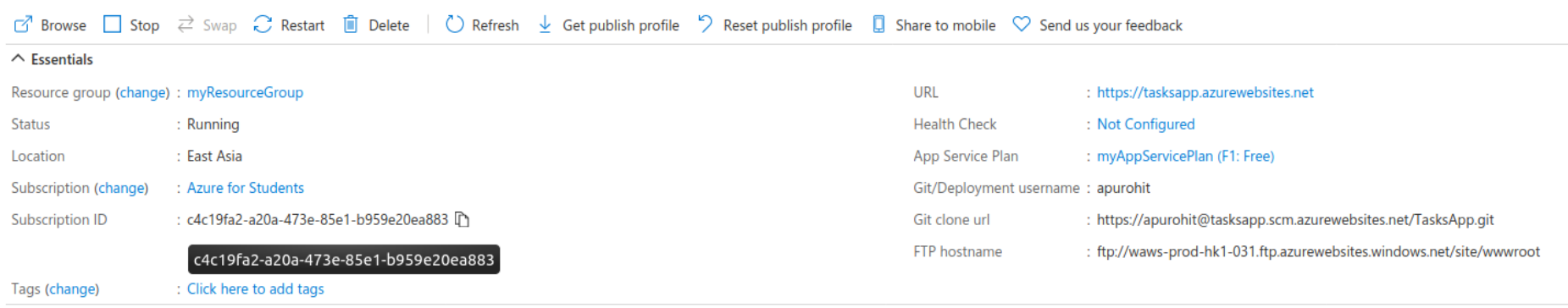

```
remote: found 49 vulnerabilities (21 low, 12 moderate, 16 high)
remote:   run `npm audit fix` to fix them, or `npm audit` for details
remote: Finished successfully.
remote: Running post deployment command(s)...
remote: Triggering recycle (preview mode disabled).
remote: Deployment successful.
remote: Deployment Logs : 'https://tasksapp.scm.azurewebsites.net/newui/jsonviewer?view_url=/api/deployments/4ea103ac35bad2dab712088450c1d7b9422036b4/log'
To https://tasksapp.scm.azurewebsites.net/TasksApp.git
   cf3aff6..4ea103a  master -> master
```

Azure Services

1. Azure Database for MySQL server

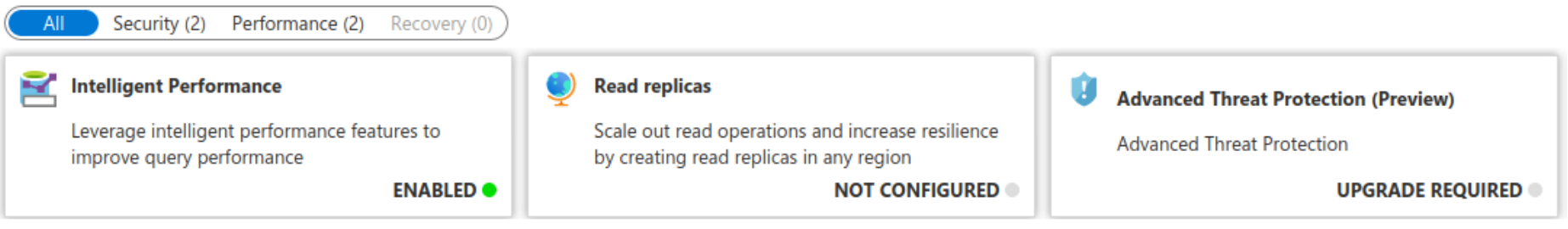


2. Azure App Service



Security & Performance (Additional)

For the Azure Database for MySQL server, I have enabled **Intelligent Performance** by setting **query_store_capture_mode = ALL**. When set to NONE, it does not capture any statements. This serves somewhat similar purpose to caching and improves the query performance.



For the security measures provided by Azure which includes Advanced Threat Protection, Azure Defender for App Services, an upgrade from the Basic(Free) plan is required.
