## Steps

* Create a MySQL database in Azure
* Connect the app to the Azure MySQL database

In the next two posts, we will:

* Deploy the app to an Azure Linux VM
* Deploy the app to Docker via Azure Container Service

## Prerequisites

* [A Git Client](https://git-scm.com/)
* [Java 7 JDK or above](http://www.oracle.com/technetwork/java/javase/downloads/index.html)
* [The Azure CLI 2.0](https://docs.microsoft.com/cli/azure/install-azure-cli)
* An Azure account ([Free Trial here](https://azure.microsoft.com/free/))

## Clone the GitHub Sample

From the command prompt, navigate to a working directory and clone [the sample repository](https://github.com/bbenz/spring-boot-todo).

git clone https://github.com/bbenz/spring-boot-todo

Configure the app to use the MySQL database

## Create an Azure MySQL Database With the Azure CLI

Next up, let’s create an [Azure Database for MySQL](https://azure.microsoft.com/en-us/services/mysql/) instance using the [Azure CLI](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest). We will Use the Azure CLI 2.0 in a terminal window to create the resource group and a MySQL instance.

## Log In and Create a Resource Group

Log in to your Azure subscription with the az login command, then follow the on-screen directions.

## Create an Azure Resource Group

[Azure resource groups](https://docs.microsoft.com/en-us/cli/azure/group?view=azure-cli-latest#az_group_create) manage Azure services together as a unit. Each resource group must have a location. To see all possible values you can use for — location, use the az appservice list-locations command.

The following example creates an Azure resource group in the North Europe region.

az group create — name myResourceGroup — location “North Europe”

## Create a MySQL Server

Create a server in Azure Database for MySQL.

Substitute your own unique MySQL server name where you see the <mysql\_server\_name> placeholder. This name is part of your MySQL server’s hostname, <mysql\_server\_name>.mysql.database.azure.com, so it needs to be globally unique. Also substitute <admin\_user> and <admin\_password> with your own values.

az mysql server create — name <mysql\_server\_name> — resource-group myResourceGroup — location “North Europe” — admin-user <admin\_user> — admin-password <admin\_password>

When the MySQL server is created, the Azure CLI returns information similar to this example:

{

“administratorLogin”: “admin\_user”,

“administratorLoginPassword”: null,

“fullyQualifiedDomainName”: “mysql\_server\_name.mysql.database.azure.com”,

“id”: “/subscriptions/00000000–0000–0000–0000–000000000000/resourceGroups/myResourceGroup/providers/Microsoft.DBforMySQL/servers/mysql\_server\_name”,

“location”: “northeurope”,

“name”: “mysql\_server\_name”,

“resourceGroup”: “mysqlJavaResourceGroup”,

}

## Configure a MySQL Firewall

Create a firewall rule for your MySQL server to allow client connections by using the az mysql server firewall-rule create command. Here’s an example that creates a firewall rule for a broad range of IP addresses (You will likely want to narrow your actual firewall IP address range):

az mysql server firewall-rule create — name allIPs — server <mysql\_server\_name> — resource-group myResourceGroup — start-ip-address 0.0.0.0 — end-ip-address 255.255.255.255

## Configure the Azure MySQL Database

Connect to the MySQL server using the values you specified previously for <admin\_user> and <mysql\_server\_name>.

mysql -u <admin\_user>@<mysql\_server\_name> -h <mysql\_server\_name>.mysql.database.azure.com -P 3306 -p

In the mysqlprompt, create a database and a table for the to-do items.

CREATE DATABASE tododb;

Create a database user and give it all privileges in the tododb database. Replace the placeholders <Javaapp\_user> and <Javaapp\_password> with your own unique app name:

CREATE USER ‘<Javaapp\_user>’ IDENTIFIED BY ‘<Javaapp\_password>’;

GRANT ALL PRIVILEGES ON tododb.\* TO ‘<Javaapp\_user>’;

Exit your server connection by typing “quit”.

## Update the Values in the application.properties File

Update the following values in src/main/resources/application.properties:

spring.datasource.url=jdbc:mysql:// >@<mysql\_server\_name>.mysql.database.azure.com:3306/tododb

spring.datasource.username=adminname@<mysql\_server\_name>

spring.datasource.password=password

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

## Build and Run the Sample

Build and run the sample locally using the Maven wrapper included in the repo:

mvn package spring-boot:run

In a browser, open [http://localhost:8080](http://localhost:8080/)to work with the todo app.

**Create a Linux VM on Azure**

There are several ways to create a new Linux VM on Azure, for this article, we’re going to go with the command line, using the az vm create command.

NOTES:

* Use the same location as the resource group you created in part 1.
* generate-ssh-key generates a key you can use to log into the VM from this command line
* You can use a specific image or the latest image (ubuntults for the latest ubuntu)

az vm create -n <vm name> -g <resource group from part 1> -l eastus2 — generate-ssh-key — image “Canonical:UbuntuServer:16.04-LTS:16.04.201702240”

If successful, this command returns something like this. Make note of the PublicIpAddress for later:

ResourceGroup PowerState PublicIpAddress PrivateIpAddress MacAddress Location — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — rgJan2018 VM running 152.23.190.52 10.0.0.4 00–0D-3A-00-C0–0E eastus2

**Open port 8080**

Next, open port 8080 on the target VM:

az vm open-port -n <vm name> -g <resource group from part 1> — port 8080

This should return:

Location Name ProvisioningState ResourceGroup ResourceGuid — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — eastus2 bbmysqlvmdemo42NSG Succeeded rgJan2018 c9e163fc-a9e4–4975-aa9f-1254e0bee9335

**SSH into the VM**

Get into the VM with the SSH key you generated as part of the az vm create command. Use the <username> from your VM — if no <username> was specified, then your account email to the left of the @ is usually the default.

ssh <username>@< PublicIpAddress>

You should get this response the first time. Say yes.

The authenticity of host ‘< PublicIpAddress> (< PublicIpAddress>)’ can’t be established.ECDSA key fingerprint is …………Are you sure you want to continue connecting (yes/no)? yesWarning: Permanently added < PublicIpAddress> (ECDSA) to the list of known hosts.

Next, make sure you have the latest Linux available, and add OPenJDK 8:

**Prep the VM**

sudo apt-get update  
sudo apt-get install -y openjdk-8\*

**Copy the Java app to the VM**

You’re now ready to transfer the files to the VM. The easiest way is scp (secure copy).

If you’re using Windows subsystem for Linux, use mnt to source the file to copy. Here’s an example from a using a user’s documents directory:

scp /mnt/c/Users/<username>/Documents/GitHub/todo-app-java-on-azure/target/todo-app-java-on-azure-1.0-SNAPSHOT.jar <username>@< PublicIpAddress>:~/

**Run the app on the VM**

Next, run the file on the vm:

java -jar target/todo-app-java-on-azure-1.0-SNAPSHOT.jar

The app can be accessed at:

http://< PublicIpAddress>:8080/