# Introduction

**Scenario**

Imagine that you recently heard about a new managed Azure service for running Spring Cloud applications called Azure Spring Cloud (ASC). You would like to try it out with a sample Spring Cloud application, which already exists, called PiggyMetrics.

**Instructions**

The following pages will contain various tasks for you to try out, as well as more information about the PiggyMetrics application.

Please think out loud as you work through the tasks, so that we can learn about what is working well and what is confusing or unclear. By helping us understand what you are thinking about, you will help us improve the design of the product.

Please let us know when you think you’re done with a task. After each task, we will ask you to rate how easy or difficult the task was on a scale of 1-5: (1) = Very Difficult, (5) = Very Easy

Do you have any questions before we begin?

# Task 1: Provision the Service

*If you have an Azure account and are already logged into the Azure Portal, please start a new private browsing window.*

Navigate to the Azure porta using this link: [https://ms.portal.azure.com/?microsoft\_azure\_marketplace\_ItemHideKey=AppPlatformExtension#blade/Microsoft\_Azure\_Marketplace/MarketplaceOffersBlade/selectedMenuItemId/home/searchQuery/spring](https://ms.portal.azure.com/?microsoft_azure_marketplace_ItemHideKey=AppPlatformExtension" \l "blade/Microsoft_Azure_Marketplace/MarketplaceOffersBlade/selectedMenuItemId/home/searchQuery/spring)

Log into the Portal with the following credentials:

* Username: [mstest\_chpay@outlook.com](mailto:mstest_chpay@outlook.com)
* Password: 1234567890MS

Provision a new Azure Spring Cloud (ASC) service with the following details:

* **Name:** *Choose your own*
* **Subscription:** Java Tooling Tests with TTL = 7 Days
* **Resource Group:** rg-usabilitytesting
* **Location:** *Choose your own*

# Task 2: Configure your Application

**Context:**

Imagine that you have a Spring Cloud application called PiggyMetrics. PiggyMetrics is built on Java 8 and consists of three functional microservices: gateway, account-service and auth-service. You would like to configure the application.



*PiggyMetrics application*

In the Spring Cloud world, a configuration server fetches configurations from a repository and provides the configuration to microservices when needed.

The configurations for your PiggyMetrics application are located in the following Git repository: <https://github.com/Azure-Samples/piggymetrics>. PiggyMetrics’s Config Server needs know the repository in order to fetch configuration data.

**Tasks:**

1. Switch to the **asc-usabilitytesting** service in the Azure Portal. This is an instance of the Azure Spring Cloud service that we have pre-created for you to save time.
2. Set up the config server in the **asc-usabilitytesting** service so that the applications read configuration data from the following Git repository: <https://github.com/Azure-Samples/piggymetrics>

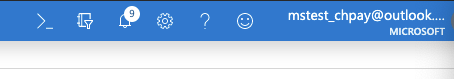
# Task 3 Build and deploy three applications

**Context:**

You would like to build, create on Azure and deploy to Azure three PiggyMetrics microservices: gateway, account-service, and auth-service.

**Tasks:**

Open the Azure Cloud Shell



# Clone git repository

git clone https://github.com/Azure-Samples/piggymetrics

# Change directory and build the project:

cd PiggyMetrics  
mvn clean package -DskipTests

# Login to Azure CLI

az login

# Set active subscription

az account set --subscription 685ba005-af8d-4b04-8f16-a7bf38b2eb5a

# Configure the default resource group name and service instance name.

az configure --defaults group=rg-usabilitytesting

az configure --defaults spring-cloud=<service instance name>

Create an application called **gateway** via the Azure Portal UI (not via Cloud Shell). Remember that PiggyMetrics is built on Java 8.

You deploy gateway to Azure.

# Install the Azure Spring Cloud CLI extension

az extension add --name spring-cloud

# Deploy the gateway application

az spring-cloud app deploy -n gateway --jar-path ./gateway/target/gateway.jar

# Create and deploy the account-service and auth-service applications using the Cloud Shell command line

az spring-cloud app create -n account-service

az spring-cloud app deploy -n account-service --jar-path ./account-service/target/account-service.jar

az spring-cloud app create -n auth-service

az spring-cloud app deploy -n auth-service --jar-path ./auth-service/target/auth-service.jar

# Task 4: Assign a public endpoint

Make it so that there is a publicly accessible endpoint to the gateway application. Access the application endpoint to ensure that it is up and running.

# Task 5: Restart

Restart the gateway application.

# Task 6: Adjust your application's resources

Make it so that the gateway app is running on 3 instances with 3 CPU and 2GB of memory each.

# Task 7 Enable and Use Distributed Tracing

Imagine there is a sample Spring application running on an Azure Spring Cloud service instance of **asc-usabilitytesting**, called PiggyMetrics. The application consists of three micro services, gateway, account-service and auth-service. You are interested in enabling and using distributed tracing for PiggyMetrics.



Go to asc-usabilitytesting service in the Azure Portal

Enable Distributed Tracing for this service

Launch an application map to view

Find out what is the top error response code for gateway during last 24 hours

# Task 7 Deploy to staging

Imagine you made a CSS file change, fliptext’s background color from white to black, on gateway’s launch page. You will deploy staging and validate firstly, and after the validation, you will continue deploying the change to production.

To save your time, we have made the change and re-built.

Deploy new build to staging

CD green/PiggyMetrics

az spring-cloud app deployment create --app gateway -n green --jar-path ./gateway/target/gateway.jar