**Lab Guide for Setting Up and Deploying a Java Web Application to Azure App Service**

**Prerequisites:**

1. **Visual Studio Code** installed on your machine.  
   Download: [VS Code](https://code.visualstudio.com/).
2. **Git** installed for version control.  
   Download: [Git](https://git-scm.com/).
3. **Java Development Kit (JDK 21)** installed.  
   Download: [JDK 21](https://www.oracle.com/in/java/technologies/downloads/#jdk21-windows).
4. **Apache Maven** installed.  
   Download: [Apache Maven 3.9.9](https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.zip).
5. Azure subscription with Azure App Services configured.

**Step 1: Clone the Java Web Application**

1. Open a terminal or Git Bash.
2. Clone the repository:

bash

Copy code

git clone https://github.com/ramannkhanna2/JavaWebAppp.git

1. Verify the repository is cloned:

bash

Copy code

cd JavaWebAppp

ls

**Step 2: Install Java**

1. Download the Java JDK 21 installer from [Oracle's Java Downloads](https://www.oracle.com/in/java/technologies/downloads/#jdk21-windows).
2. Run the installer and follow the on-screen instructions.
3. Set the JAVA\_HOME environment variable:
   * Open **System Properties** > **Advanced** > **Environment Variables**.
   * Add a new system variable:
     + Variable name: JAVA\_HOME
     + Variable value: Path to your JDK installation (e.g., C:\Program Files\Java\jdk-21).
   * Add %JAVA\_HOME%\bin to the Path variable in **System Variables**.
4. Verify the installation:

bash

Copy code

java -version

**Step 3: Install Maven**

1. Download Maven binaries from [Apache Maven](https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.zip).
2. Extract the archive to a folder (e.g., C:\Users\HP\Downloads\apache-maven-3.9.9-bin\apache-maven-3.9.9).
3. Add Maven to the Path environment variable:
   * Add: C:\Users\HP\Downloads\apache-maven-3.9.9-bin\apache-maven-3.9.9\bin.
4. Verify the installation:

bash

Copy code

mvn -version

**Step 4: Build the Java Web Application**

1. Open **Visual Studio Code** and navigate to the JavaWebAppp folder:

bash

Copy code

cd JavaWebAppp

1. Open the folder in VS Code:

bash

Copy code

code .

1. Use Maven to build the artifact:

bash

Copy code

mvn clean package

1. Locate the JAR file in the target directory (e.g., target/JavaWebApp.jar).

**Step 5: Deploy to Azure App Service**

1. Install the **Azure App Service** extension in VS Code:
   * Open the **Extensions** view (Ctrl+Shift+X).
   * Search for and install **Azure App Service**.
2. Sign in to your Azure account via VS Code:
   * Use the Azure icon in the Activity Bar.
   * Click **Sign In to Azure** and complete the authentication process.
3. Deploy the application:
   * Right-click on the generated JAR file (target/JavaWebApp.jar).
   * Select **Deploy to Web App**.
   * Follow the prompts to select your Azure subscription and create a new App Service (or select an existing one).

**Step 6: Test and Redeploy**

1. Open your Azure App Service URL in a browser to test the deployed application.
2. Make changes to your code in the JavaWebAppp project as needed.
3. Rebuild and redeploy:

bash

Copy code

mvn clean package

Repeat the deployment process.

**Troubleshooting and Tips:**

1. **Verify Azure App Service**: Ensure you have the proper permissions in your Azure subscription.
2. **Check Logs**:
   * Use the Azure App Service logs to debug deployment issues.
   * Access logs from the **Azure Portal** or via the VS Code extension.
3. **Update Maven**:
   * Use mvn dependency:resolve to resolve dependency issues during builds.

This guide will help you set up the environment, build the web application, and deploy it to Azure App Service effectively.

4o

**Lab Guide for Creating Deployment Slots and Performing A/B Testing in Azure App Service**

**Objective:**

1. Create a deployment slot (otherversion) for testing.
2. Deploy a modified version of the application to the new slot.
3. Perform A/B testing by splitting traffic between slots.
4. Perform a slot swap.

**Step 1: Prerequisites**

Ensure the following are set up:

1. **Azure App Service** with a default production slot (ramanwebapp-auhye5chduakg4h5.canadacentral-01.azurewebsites.net).
2. **Azure App Service Extension** and **Azure Resources Extension** installed in Visual Studio Code.
   * Install via Extensions view (Ctrl+Shift+X).

**Step 2: Create a Deployment Slot**

1. Open the **Azure Portal** and navigate to your App Service.
2. Under the **Deployment Slots** section, click **Add Slot**.
3. Configure the slot:
   * **Name**: otherversion.
   * Clone settings from the default slot (optional).
4. After creation, the slot URL will look like:

Copy code

ramanwebapp-otherversion-hnhaccfyeqhsd6hc.canadacentral-01.azurewebsites.net

**Step 3: Modify Code for Deployment Slot**

1. Open the JavaWebAppp project in **Visual Studio Code**:

bash

Copy code

code .

1. Modify the code for the otherversion slot (e.g., change the background color of a UI component to purple).
2. Save changes.

**Step 4: Build and Deploy to the otherversion Slot**

1. **Build the artifact**:

bash

Copy code

mvn clean package

1. **Deploy to the otherversion slot**:
   * Go to the **Azure Resources** section in Visual Studio Code.
   * Navigate to your App Service > otherversion slot.
   * Right-click on the JAR file (target/JavaWebApp.jar) and select **Deploy to Slot**.
   * Follow the prompts to deploy to the otherversion slot.
2. Verify the deployment by accessing the slot URL:

Copy code

ramanwebapp-otherversion-hnhaccfyeqhsd6hc.canadacentral-01.azurewebsites.net

**Step 5: Perform a Slot Swap**

1. Navigate to **Deployment Slots** in the Azure Portal.
2. Click **Swap** and select the source and target slots:
   * Source: otherversion.
   * Target: production (default).
3. Confirm the swap.
4. Test the application using the production slot URL:

Copy code

ramanwebapp-auhye5chduakg4h5.canadacentral-01.azurewebsites.net

**Step 6: Perform A/B Testing (Canary Deployment)**

1. Go to the **Azure Portal** > **App Service** > **Deployment Slots**.
2. Click **Traffic Routing**.
3. Configure traffic distribution:
   * **Default Slot**: 50% (or your desired percentage).
   * **Otherversion Slot**: 50% (or remaining percentage).
4. Save the configuration.

**Step 7: Test the Traffic Split**

1. Access the default domain of your App Service:

Copy code

ramanwebapp-auhye5chduakg4h5.canadacentral-01.azurewebsites.net

1. Test functionality and note which version (default or otherversion) is displayed. Traffic is split based on the configured percentage.

**Optional Step: Reset Traffic Split**

1. If testing is complete, reset the traffic split:
   * Set **Default Slot** to 100%.
   * Set **Otherversion Slot** to 0%.
2. Save the changes.

**Best Practices:**

1. **Log Monitoring**: Use Azure Monitor or App Insights to track performance metrics during A/B testing.
2. **Backup Production Data**: Ensure you have a backup before swapping slots.
3. **Test Thoroughly**: Verify the functionality in both slots during A/B testing.

This guide provides a systematic approach to using deployment slots, swapping, and canary deployments for A/B testing in Azure App Service.