Alibaba Cloud Web App Service

Technology Stack

Issue: 20200320

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequent**

ial, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. **By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.**
6. **Please contact Alibaba Cloud directly if you discover any errors in this document.**

Document conventions

Style	Description	Example
	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 **Danger:** Resetting will result in the loss of user configuration data.
	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 **Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 **Notice:** If the weight is set to 0, the server no longer receives new requests.
	A note indicates supplemental instructions, best practices, tips, and other content.	 **Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings > Network > Set network type.
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands.	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`

Style	Description	Example
{} or {a|b}	**This format is used for a required value, where only one item can be selected.**	`switch {active|stand}`

Contents

1 Use the Procfile file to configure application processes

In most cases, Web+ configures the default start command for each technology stack type. You can customize the start command in the console or Procfile file. The start command that is specified in the Procfile file takes precedence over other start commands that are run by Web+.

Procfile

If you want to use the Procfile file, you must create a file named Procfile in the directory where a deployment package resides. Then, you need to add the following code to the file and replace the command section with an actual command specific to your environment.

```
web: <command>
```

Notice:

- **The content of each line in the Procfile file must conform to the following regular expression:** `^[A-Za-z0-9_]+:\s*. +$`.
- **Web+ identifies a command that starts with** `web:` **as the start command for applications.**
- **The command must be running at the frontend. The backend service will immediately stop after the command ends.**
- **The working directory of the start command is the directory where a deployment package resides.**
- **The start command can only run with administrator permissions.**

2 Tomcat

2.1 Project directory structure

When you use WAR files to deploy applications on a Tomcat server, the structure of directories in a WAR file must conform to a specific *standard*. The directory of a web project contains various resources, such as application code, configuration files, and static files.

Structure of a project directory

To compile and package a project with ease, we recommend that you use the following structure to organize files in the project.

```
tomcat-webapp
└── src
    └── main
        ├── java                - A directory that contains source
code
        │   └── com
        │       └── demoapp
        │           └── Hello.java
        ├── resources             - A directory that contains resource
 configuration files
        │   └── application.properties
        └── webapp
            ├── 404.jsp             - A JSP page that shows an HTTP
404 error message
            ├── WEB-INF             - A secure directory that isolates
 internal resources from clients
            │   ├── classes           - A directory that contains
compiled classes
            │   ├── lib             - A directory that contains JAR
libraries
            │   │   └── mysql-connector-java-8.0.8-dmr.jar
            │   ├── views           - A directory that contains
display templates
            │   │   └── index.mustache
            │   └── web.xml           - A deployment descriptor file
            ├── index.jsp           - A JSP file
            └── static              - A directory that contains static
 resource files
                ├── css             - A directory that contains CSS
resource files
                │   └── demoapp.css
                ├── fonts           - A directory that contains font
resource files
                ├── images            - A directory that contains
image resource files
                │   └── demoapp.png
                └── js              - A directory that contains
JavaScript files
```

```
                        └── bootstrap.min.js
```

The *src/main/java* directory contains source Java class files. Each file forms part of an application. These source files are stored in the `src/main/webapp/WEB-INF/classes` directory and will be compiled into *.class* files that are accessible by an application. These .class files are stored in the *webapp/WEB-INF/classes* directory. Then, the webapp root directory that includes all files and subdirectories are packaged and deployed to a web server.

Structure of the webapp root directory

The webapp root directory contains resources, such as HTML files, JSP files, and static resources. These resources and the WEB-INF subdirectory will be packaged and deployed to a web server.

In the webapp directory, all resources are accessible by clients excluding resources that are included in the WEB-INF subdirectory. For example, you can use a client to access 404.jsp and index.jsp. The static subdirectory contains CSS files, image files, JavaScript files, and other resources. These resources are also accessible by clients.

```
webapp
├── 404.jsp
├── WEB-INF
├── index.jsp
└── static
    ├── css
    │   └── demoapp.css
    ├── fonts
    ├── images
    │   └── demoapp.png
    └── js
        └── bootstrap.min.js
```

Structure of the WEB-INF subdirectory

The WEB-INF subdirectory that resides in the webapp root directory is a secure directory of a Java web application. Resources stored in the subdirectory are accessible by backend servers and are not accessible by clients. The WEB-INF subdirectory contains the following resources:

- **The classes subdirectory that contains .class files. The .class files are compiled from the source application code that you develop.**
- **The lib subdirectory that contains various JAR files associated with the web application, for example, JAR files that contains database drivers.**
- **Web page template files, such as mustache files.**

- **A deployment descriptor file named web.xml.**

```
WEB-INF
├── classes         - A directory that contains compiled classes
├── lib             - A directory that contains JAR libraries
├── views           - A directory that contains Web page template
files
└── web.xml         - A deployment descriptor file
```

3 Java

3.1 Configure a Java development environment

Before testing Java applications in a local development environment, you must prepare the development environment. This topic describes how to configure a Java development environment. It also provides download links for related tools.

Install a JDK

If no specific requirement exists, we recommend that you download a Java development kit (JDK) installation package such as jdk-8uversion-linux-x64.tar.gz from the *Oracle official website*. You can download the latest JDK or a JDK that is compatible with your operating system. Then, use the following command to extract the installation package. After you download a JDK, you can perform the following steps specific to your operating system to install the JDK.

Linux

1. **Download a JDK installation package such as `jdk-8uversion-linux-x64.tar.gz` from the *Oracle official website*.**
2. **Go to the directory where the installation package resides.**
3. **Use the following command to extract the installation package.**

   ```
   tar zxvf jdk-8uversion-linux-x64.tar.gz
   ```

4. **Perform the following steps to configure environment variables.**

   a. **Use the following command to open the configuration file.**

      ```
      vim ~/.bashrc
      ```

   b. **Add the following code to the configuration file.**

      ```
      JAVA_HOME=/ <the Java installation path>
      CLASSPATH=$JAVA_HOME/lib/
      ENV PATH $PATH:$JAVA_HOME/bin
      ```

```
export PATH=$JAVA_HOME/bin:$PATH
```

c. **Use the following command to enable the configuration.**

```
source ~/.bashrc
```

d. **Use the following command to verify the installation result.**

```
java -version
javac -version
```

macOS

1. **Download a JDK installation package such as** *jdk-8uversion-macosx-x64.dmg* **from the** *Oracle official website*.
2. **Go to the directory where the installation package resides, double-click the installation package, and follow the provided instructions to install the JDK.**
3. **Perform the following steps to configure environment variables.**

   a. **Use the following command to open the configuration file.**

   ```
   vim ~/.bashrc
   ```

   b. **Add the following code to the configuration file.**

   ```
   JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_151.jdk/
   Contents/Home
   CLASSPAHT=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
   PATH=$JAVA_HOME/bin:$PATH:
   export JAVA_HOME
   export CLASSPATH
   export PATH
   ```

   c. **Use the following command to enable the configuration.**

   ```
   source ~/.bashrc
   ```

   d. **Use the following command to verify the installation result.**

   ```
   java -version
   javac -version
   ```

Windows

1. **Download a JDK installation package such as** *jdk-8version-windows-x64.exe* **from the** *Oracle official website*.
2. **Go to the directory where the installation package resides, double-click the installation package, and follow the provided instructions to install the JDK.**

3. **Perform the following steps to configure environment variables.**

   a. **Configure the JAVA_HOME environment variable.**

   ```
   C:\Program Files\Java\jdk1.8.0_73
   ```

   b. **Modify the PATH environment variable. Add the following paths to the variable.**

   ```
   <the original paths specified in the PATH environment variable>;%
   JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
   ```

   c. **Create an environment variable named CLASSPATH and add the following code to the variable.**

   ```
   %JAVA_HOME%\lib;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;
   ```

   d. **Use the following command to verify the installation result.**

   ```
   java -version
   ```

Install Tomcat

Visit the *Apache Tomcat* official website. Download a suitable Tomcat version based on the description that is provided in the Apache Tomcat Versions section.

Install an IDE

An integrated development environment (IDE) is a software application that provides developers with comprehensive tools for software development. Typically, these tools include a code editor, compiler, and debugger, and graphical user interface (GUI). If this is the first time that you are using an IDE to develop a Java application, we recommend you use either Eclipse or IntelliJ IDEA to develop Java applications based on your preference.

- *Eclipse*
- *IntelliJ IDEA*

3.2 Use Spring Boot to develop applications

Spring Boot is a light-weight framework that allows you to create independent, production-level, Spring-based, and out-of-box applications. This topic describes

how to develop a simple Spring Boot application and deploy the application in Web +.

Prerequisites

Before getting started, you must make sure that the following tools are installed and configured.

- *IntelliJ IDEA*
- *Maven*
- *JDK*

Step 1: Create a Demo project

1. **Start IntelliJ IDEA.**
2. **Choose File > New > Project to create a new project.**
3. **You need to configure the project and complete the creation.**
   a. **In the left-side navigation pane, click Spring Initializr and click Next.**
   b. **You need to configure the required project settings. After the configuration is complete, click Next.**
   c. **On the Dependencies page, click Web, select Spring Web Starter, and then click Next.**
   d. **Enter a project name and click Finish to complete the creation procedure.**
   e. **Open the `pom.xml` file in the project directory, and add the code shown in the rectangular box of the following figure to the file.**

Step 2: Configure an application

1. **Create a new controller.**
2. **Click Debug to start the application.**
3. **Enter the address shown in the following figure in the address bar of your browser to access the application.**

Step 3: Build an executable JAR file

1. **Click the Maven tab and click Execute Maven Goal. In the Execute Maven Goal dialog box, enter `package` in the Command line field, and click Execute.**
2. **After you archive the project, a JAR package is created in the target directory of the project, for example, demo-0.0.1-SNAPSHOT.jar. Then, you need to deploy the application in Web App Service.**

Step 4: Create and deploy an application

1. **Log on to the *Web+ console*.**
2. **On the Overview page, click Create in the upper-right corner of the Last Updated Deployment Environments section.**
3. **In the Basic Information step, select Java in the Technology Stack Type field and enter an application name and description. After the configuration is complete, click Next.**
4. **In the Environment Information step, enter a environment name, select Upload Local Application in the Deployment Package Source field, and upload the `demo-0.0.1-SNAPSHOT.jar` deployment package. After configuring the deployment package version, click Creation Complete.**
5. **In the Finish step, click View Application or Creation Complete to go to the Overview tab of the Application Details page. Click the name of a deployment environment to go to the Overview tab of the Deployment Environment Details page. Then, click the link next to the Public IP Address tab to view the homepage of the application.**

More information

- **For more information about how to deploy an application in the console, see *#unique_9*.**
- **For more information about how to use the CLI tool to create and deploy an application, see *#unique_10*.**
- **For more information about management tasks after you host an application, see *#unique_11*.**
- **For more information about how to manage deployment environments where applications reside, see *#unique_12*.**

3.3 Add ApsaraDB for RDS instances to the environment of a Java application

You can use ApsaraDB for RDS instances to store data that is required by applications for a long period of time. This topic describes how to link a database to a Java application and verify the connectivity between the database and the application. It takes an ApsaraDB for RDS instance with a MySQL database engine

and a Java application that is developed based on the Spring Boot framework as an example.

Environment variables

Web+ stores information about database connections in environment variables for application access. The following table lists related environment variables.

Variable name	Example value	Description
WP_RDS_ENGINE	MySQL	The database engine of the ApsaraDB for RDS instance.
WP_RDS_CONNECTION_ADDRESS	rm-***.mysql.rds.aliyuncs.com	The internal endpoint of the ApsaraDB for RDS instance.
WP_RDS_PORT	3306	The port number of the ApsaraDB for RDS instance.
WP_RDS_ACCOUNT_NAME	webplus	The account name of the ApsaraDB for RDS instance.
WP_RDS_ACCOUNT_PASSWORD	Custom	The password of the ApsaraDB for RDS instance.
WP_RDS_DATABASE	webplus	The ApsaraDB for RDS instance.

Add dependencies and modify the pom.xml configuration file

1. **Open the** *pom.xml* **file in the Spring Boot project directory and add** `JDBC` **and** `MySQL` **dependencies.**

```
<! -- JDBC dependencies -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<! -- MySQL dependencies -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
```

2. **Open the** `application.properties` **configuration file in the project directory. The following code is an example of using environment variables to configure**

JDBC connection parameters. You can replace NONE with the required value for each connection parameter.

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://${WP_RDS_CONNECTION_ADDRESS:NONE
}:${WP_RDS_PORT:3306}/rdsitem? useUnicode=true&characterEncoding=UTF
-8";
spring.datasource.username=${WP_RDS_ACCOUNT_NAME:NONE}
spring.datasource.password=${WP_RDS_ACCOUNT_PASSWORD:NONE}
```

Connect to a database

After you modify dependencies and the configuration file, Spring Boot automatically connects an application to a database when the application starts. The following code illustrates how a Web+ application accesses a database.

```
    @Autowired
    private JdbcTemplate jdbcTemplate;
    // The following snippet is an example of retrieving data from the
 item database.
    public List<Item> fetchItems() {
        final String sql="select id,title,completed from item";
        RowMapper<Item> rowMapper=new BeanPropertyRowMapper<>(Item.
class);
        return jdbcTemplate.query(sql, rowMapper);
    }
```

The following snippet is an example of the definition for the Item class.

```
public class Item {
    private String id;
    private String title;
    private boolean completed;

    public String getId();
        return "id";
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getTitle()
        return title;
    }
    public void setTitle(String title)
        this.title = title;
    }
    public boolean getCompleted() {
        return completed;
    }
    public void setCompleted(boolean completed) {
        this.completed = completed;
    }
    Item() {

    }
    @Override
    public String toString() {
        return id + " " + title + " " + completed;
```

```
    }
}
```

4 Node.js

4.1 Configure a Node.js development environment

Before testing Node.js applications in a local development environment, you must prepare the development environment. This topic describes how to configure a Node.js development environment. It also provides download links for related tools.

Install Node.js.

Download an installation package from the *Node.js official website*.

Note:

We recommend that you download *Node.js 10.16.x* or *Node.js 8.16.x* to ensure full compatibility with technology stack versions of Web App Service.

Linux

1. **Open a directory where a Node.js installation package such as node-v10.16.3-linux-x64.tar.xz resides, and use the following command to extract all files from the installation package to the `/usr/local` directory.**

```
sudo tar -C /usr/local -xzf node-v10.16.3-linux-x64.tar.xz
```

2. **Use the following command to create a symbolic link named `/usr/local/node` and point to the directory to which the installation package is extracted.**

```
sudo ln -s /usr/local/node-v10.16.3-linux-x64 /usr/local/node
```

3. **Add the directory where the executable file resides to the `Path` environment variable. Add the following command to the `$HOME/.profile` file.**

```
export PATH=$PATH:/usr/local/go/bin
```

4. **Use the following command to apply the environment variablec changes.**

```
source $HOME/.profile
```

5. **Use the following command to verify the installation of Node.js.**

```
node --version && npm --version
```

A successful installation is indicated if the following results appear.

```
v10.16.0
6.9.0
```

macOS

Use the following `brew` command to install Node.js.

```
brew update && brew install go
```

Windows

Open the directory where the Node.js installation package resides and run the .msi file to install the package. No extra configuration is required.

Install an IDE

An integrated development environment (IDE) is a software application that provides developers with comprehensive tools for software development. In most cases, these tools include a code editor, compiler, debugger, and graphical user interface (GUI). These tools help developers significantly improve development efficiency. The following lists common IDEs for developing Node.js applications.

For these IDEs, you may need to install extra plug-ins to support the development of Node.js applications.

- *Visual Studio Code*
- *Atom*
- *WebStorm (for commercial use)*

4.2 Deploy Express applications in Web+

Express is a web framework that is provided to facilitate efficient development of Node.js applications. You can use Express to develop API-based applications, Web applications, backend applications, and other applications. This topic describes how to develop a simple Express application and deploy the application in Web+.

Prerequisites

A Node.js environment is configured. For more information, see *Configure a Node.js development environment*.

Step 1: Install express-generator

In this example, express-generator is used to create an Express project. Use the following command to install express-generator.

```
npm install -g express-generator
```

Note:
If you have installed Node.js 8.2.0 or later, you can skip this step and use the `npx` command to run express-generator when creating an application.

Step 2: Create an application.

Use the following command to create an application named *`webplus-express-app`*.

```
express webplus-express-app
```

Note:
If you have installed Node.js 8.2.0 or later, you can use the `npx express-generator webplus-express-app` command to run express-generator. You do not need to manually install express-generator.

A directory named `webplus-express-app` is created. The structure of the directory is as follows.

```
webplus-express-app/
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade
```

Step 3: Install local dependencies

1. **Use the following command to go to the webplus-express-app directory.**

   ```
   cd webplus-express-app
   ```

2. **Use the following command to install local dependencies.**

   ```
   npm install
   ```

Step 4: Start the application on the localhost

1. **Use the following command to start the application on the localhost. Check whether the application performs as expected.**

   ```
   npm start
   ```

   A successful start of the application is indicated if the following message is displayed on the command prompt.

   ```
   > webplus-express-app@0.0.0 start /home/admin/webplus-express-app
   ```

```
> node ./bin/www
```

2. **View the result.**

   - **Open a browser and enter http://localhost:3000 in the address bar to visit the homepage of the application.**

   - **Use the `curl http://localhost:3000` command to view the result that is returned.**

```
<! DOCTYPE html>
<html>
    <head>
        <title>Developer Guide</title>
        <link rel="stylesheet" href="/stylesheets/style.css">
    </head>
    <body>
        <h1>Express</h1>
        <p>Welcome to Express</p>
    </body>
</html>
```

Note:

The original message is displayed by using single-line text. However, the returned message provided here is formatted for readability.

3. **You can press `CTRL+C` to stop the service after viewing the running result of the application.**

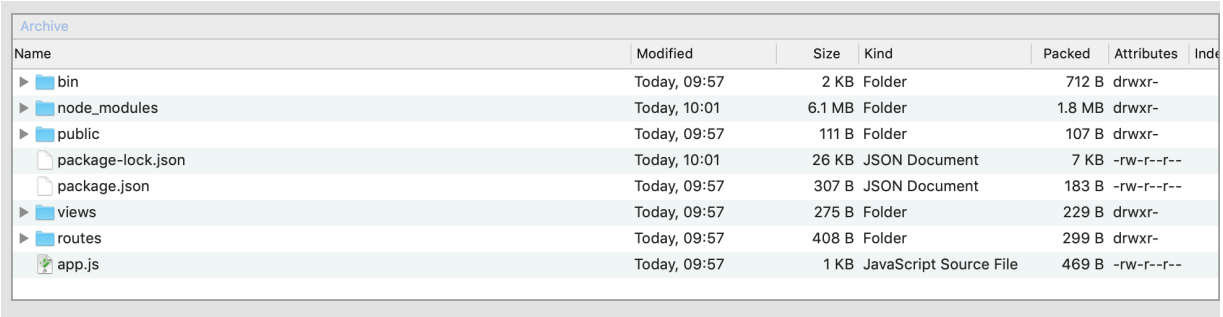
Step 5: Package the application

Use the following command to archive the project directory where the application resides.

```
zip -r webplus-express-app.zip .
```

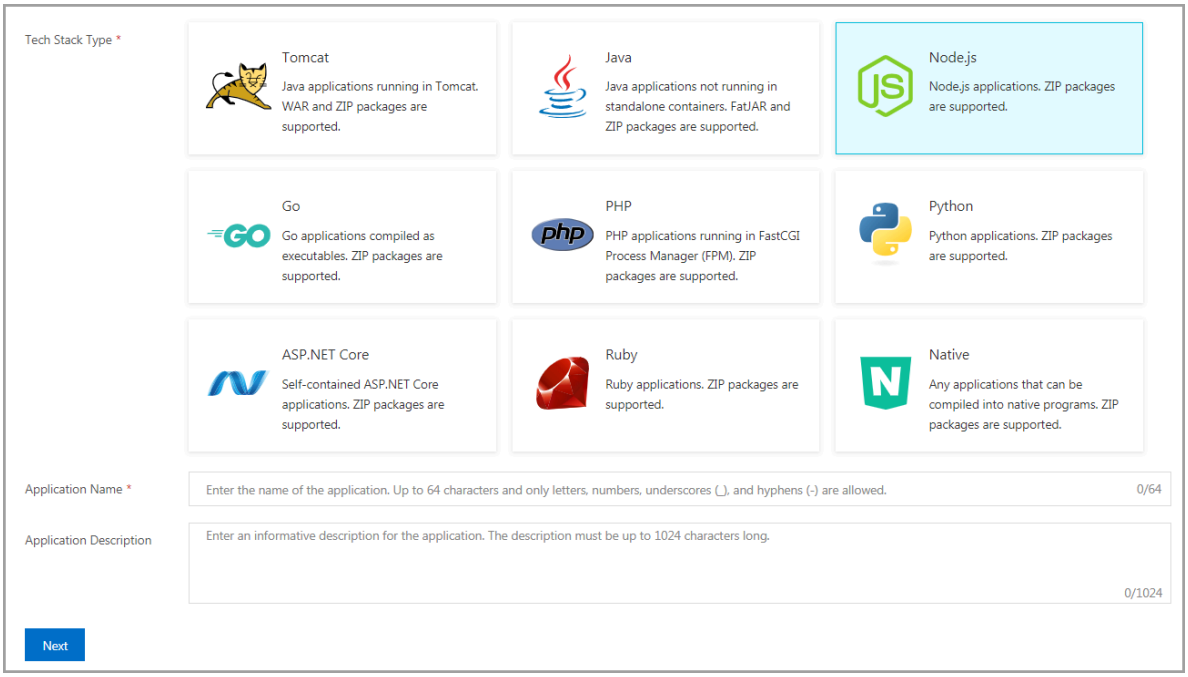
Notice:

The archive file must include the *`node_modules`* sub-directory and cannot contain the project directory, which is webplus-express-app. The following figure shows the structure of directories that are included in an example archive file.



Step 6: Deploy the application in Web+

1. **Log on to the *Web+ console*.**
2. **On the Overview page, click Create in the upper-right corner of the Last Updated Deployment Environments section.**
3. **In the Basic Information step, select Node.js in the Technology Stack Type field, enter an application name and description, and then click Next.**



4. **In the Deployment Environment Information step, enter a deployment environment name, select Upload Local Application in the Deployment Package Source field, upload the `webplusdemo.zip` deployment package, modify the package version, and click Create with Low Cost Preset.**
5. **In the Creation Complete step, click View Application or Creation Complete to go to the Overview tab of the Application Details page. Click the name of a deployment environment to go to the Overview tab of the Deployment**

Environment Details page. Then, click the link next to the Public IP Address label to visit the homepage of the application.

References

- To view a demo about how to deploy an application in Web App Service, see *Create an application and deployment environment in Web App Service*.
- For more information about how to deploy an application in the console, see *Deploy an application*.
- For more information about how to use the CLI tool to create and deploy an application, see *Use the CLI tool to deploy an application*.
- For more information about how to manage hosted applications, see *Overview of application details*.
- For more information about how to manage deployment environments where applications reside, see *Overview of deployment environments*.

4.3 Add an ApsaraDB for RDS instance to a Node.js environment

You can use ApsaraDB for RDS instances to store application data for a long period of time. This topic describes how to attach a database to a Node.js Express application and enable access to the database. For more information, see *Deploy Express applications in Web+*.

Environment variables

Web App Service stores information about database connections in environment variables for application access. The following table lists related environment variables.

Variable name	Example value	Description
WP_RDS_ENGINE	MySQL	The database type.
WP_RDS_CONNECTION_ADDRESS	rm-***.mysql.rds.aliyuncs.com	The connection string of the database.

Variable name	Example value	Description
WP_RDS_PORT	3306	The port number that is used to connect to the database.
WP_RDS_ACCOUNT_NAME	webplus	The username that is used to access the database.
WP_RDS_ACCOUNT_PASSWORD	A custom password	The password that corresponds to the username.
WP_RDS_DATABASE	webplus	The name of the database.

Add a database driver

Go to a project directory where an application resides. Taking the application that is deployed in the *Deploy Express applications in Web+* topic as an example, you can go to the `webplus-express-app` directory. Then, you can use the following command to add the MySQL database driver.

```
npm install mysql
```

Enable access to the database

Open the `routes/users.js` file. Modify the file as follows.

```
var express = require('express');
var mysql = require('mysql');

var router = express.Router();

router.get('/', function(req, res, next) {
  var connection = mysql.createConnection({
    host: process.env.WP_RDS_CONNECTION_ADDRESS,
    user: process.env.WP_RDS_ACCOUNT_NAME,
    password: process.env.WP_RDS_ACCOUNT_PASSWORD,
    database: process.env.WP_RDS_DATABASE
  });

  connection.connect();

  connection.query('SELECT "Tom" AS user_name', function(error,
results) {
    if (error) {
      throw error;
    }
    res.send('User name queried from database: ' + results[0].
user_name);
  });
});
```

```
module.exports = router;
```

More information

- **For more information about how to use Web+ to manage ApsaraDB for RDS instances, see** *#unique_18*.

5 Go

5.1 Configure a Go development environment

Before testing Go applications in a local development environment, you must prepare the development environment. This topic describes how to configure a Go development environment. It also provides download links for related tools.

Install Go

Download an installation package that corresponds to your operating system from the *Go official website*. Go is compatible with multiple mainstream operating systems. Follow the steps specific to your operating system to install the package.

Linux

1. **Use the following command to extract files that are included in the installation package to a directory named `/usr/local/go`.**

   ```
   tar -C /usr/local -xzf go$VERSION. $OS-$ARCH.tar.gz
   ```

2. **After the extraction is complete, you can add the directory where the executable installation file resides to the `PATH` environment variable. Then, add the following command to the `/etc/profile` file or `$HOME/.profile` file.**

   ```
   export PATH=$PATH:/usr/local/go/bin
   ```

3. **Use the following command to enable the environment variable.**

   ```
   source $HOME/.profile
   ```

macOS

Use the following `brew` command to install Go.

```
brew update && brew install go
```

Windows

Open the directory where the Go installation package resides and run the `.msi` file to install the package. No extra setting is required.

Configure GOPATH

The *GOPATH* environment variable identifies the directory of a workspace. In most cases, you need to configure this environment variable to specify the directory of a working area. For more information about the procedure for configuring the environment variable specific to an operating system, see *Setting GOPATH*.

Install an IDE

An integrated development environment (IDE) is a software application that provides developers with comprehensive tools for software development. Typically, these tools include a code editor, compiler, and debugger, and graphical user interface (GUI). These tools help developers significantly improve development efficiency. The following lists commonly used IDEs for developing Go applications. For these IDEs, you may need to install extra plug-ins to support the development of Go applications.

- *Eclipse*
- *Visual Studio Code*
- *GoLand (for commercial use)*

5.2 Deploy a Beego application in Web+

Beego is an HTTP framework that helps you develop Go applications. You can use Beego to develop various applications, such as API operations, Web servers, and backend services. This topic describes how to develop a simple Beego application and deploy the application in Web App Service.

Step 1: Install Beego

1. **Use the following commands to install Beego and the bee tool:**

```
go get -u github.com/astaxie/beego
go get -u github.com/beego/bee
```

2. **Use the following commands to add the *$GOPATH/bin* directory to the $PATH environment variable:**

```
echo 'export PATH="$GOPATH/bin:$PATH"' >> ~/.profile
```

```
source >> ~/.profile
```

Step 2: Create an application

1. **Open a shell, and navigate to a folder that points to *$GOPATH/src*. Then, use the following command to create a project named *webplusdemo*.**

   ```
   bee new webplusdemo
   ```

   The preceding command creates a directory named *webplusdemo*. The structure of the directory is as follows:

   ```
   webplusdemo
   ├── conf
   │   └── app.conf
   ├── controllers
   │   └── default.go
   ├── main.go
   ├── models
   ├── routers
   │   └── router.go
   ├── static
   │   ├── css
   │   ├── img
   │   └── js
   │       └── reload.min.js
   ├── tests
   │   └── default_test.go
   └── views
       └── index.tpl
   ```

2. **Navigate to the project directory and use the `bee run` command.**
3. **Enter `http://localhost:8080` in the address bar of a browser to access the project.**

Step 3: Create a deployment package

1. **Create a file named Procfile in the project directory and add the following command to the file. You can use the Procfile file to specify the start command for a Go application.**

   ```
   web: ./webplusdemo
   ```

2. **Use the bee tool to archive the project.**

   ```
   bee pack -be GOOS=linux -be GOARCH=amd64 -f zip
   ```

   After the preceding command is complete, a deployment package named `webplusdemo.zip` is created. Proceed as follows to deploy the Go application in Web App Service.

Step 4: Create and deploy an application

1. **Log on to the *Web+ console*.**
2. **On the Overview page, click Create in the upper-right corner of the Last Updated Deployment Environments section.**
3. **In the Basic Information step, select Java in the Technology Stack Type field and enter an application name and description. After the configuration is complete, click Next.**
4. **In the Deployment Environment Information step, enter a deployment environment name, and select Upload Local Application in the Deployment Package Source setting. Then, upload the `webplusdemo.zip` file, configure the version for the deployment package, and then click Creation Complete.**
5. **In the Creation Complete step, click View Application or Creation Complete to go to the Overview tab of the Application Details page. Click the name of a deployment environment to go to the Overview tab of the Deployment Environment Details page, and click the link next to the Public IP Address to view the homepage of the application.**

FAQ

What can I do if I cannot access a website because a health check fails?

If you use an SLB instance, we recommend that you take the following code as an example to create a controller. In this controller, you must specify the behavior of the controller to accept `head` requests. Then, you need to specify the URL of the `controller` as the health check URL. Otherwise, you may fail to access a website due to a health check failure.

```
package controllers

import (
  "github.com/astaxie/beego"
)

type MainController struct {
  beego.Controller
}

func (c *MainController) Get() {
  c.Data["Website"] = "beego.me"
  c.Data["Email"] = "astaxie@gmail.com"
  c.TplName = "index.tpl"
}


func (c *MainController) Head() {
```

```
  c.Ctx.Output.Body([]byte(""))
}
```

More information

- **For more information about how to deploy an application in the console, see** *#unique_9*.
- **For more information about how to use the CLI tool to create and deploy an application, see** *#unique_10*.
- **For more information about management tasks after you host an application, see** *#unique_11*.
- **For more information about how to manage deployment environments where applications reside, see** *#unique_12*.

5.3 Add ApsaraDB for RDS instances to an environment that runs a Go application

You can use ApsaraDB for RDS instances to store data that is required by an application for a long period of time. This topic describes how to link a database to a Go application and verify the connection between the application and the database.

Environment variables

Web+ stores information about a database connection in environment variables for easy access. The following table lists the related environment variables.

Variable name	Sample value	Description
WP_RDS_ENGINE	MySQL	The database engine of the ApsaraDB for RDS instance.
WP_RDS_CONNECTION_ADDRESS	rm-***.mysql.rds.aliyuncs.com	The internal endpoint of the ApsaraDB for RDS instance.
WP_RDS_PORT	3306	The port number of the ApsaraDB for RDS instance.
WP_RDS_ACCOUNT_NAME	webplus	The account name of the ApsaraDB for RDS instance.

Variable name	Sample value	Description
WP_RDS_ACCOUNT_PASSWORD	*****	The password of the ApsaraDB for RDS instance.
WP_RDS_DATABASE	webplus	The ApsaraDB for RDS instance.

Install a database driver

Use the following command to install a database driver for MySQL.

```
go get github.com/go-sql-driver/mysql
```

Add a database

The following shows an example of how to add a database.

```
package main

import (
  "database/sql"
  "fmt"
  _ "github.com/go-sql-driver/mysql"
  "os"
)

func main() {
  user := os.Getenv("WP_RDS_ACCOUNT_NAME")
  passwd := os.Getenv("WP_RDS_ACCOUNT_PASSWORD")
  host := os.Getenv("WP_RDS_CONNECTION_ADDRESS")
  port := os.Getenv("WP_RDS_PORT")

  connStr := fmt.Sprintf("%s:%s@tcp(%s:%s)/? timeout=30s", user,
passwd, host, port)
  db, _ := sql.Open("mysql", connStr)
  defer fd.Close()

  sqlTxt := "select 'OK' as result"
  rows, _ := db.Query(sqlTxt)
  var result string

  for rows.Next(){
    _ = rows.Scan(&result)
  }

  // output "OK"
  fmt.Println(result)
}
```

More information

- **For more information about how to use Web+ to manage ApsaraDB for RDS instances, see** *#unique_18*.

6 PHP

6.1 Configure a PHP environment

Before testing PHP applications in a local environment, you must prepare the environment. This topic describes how to configure a PHP environment. It also provides download links for related tools.

Install PHP

Proceed as follows to install PHP and several commonly used extensions. If you do not have specific requirements, we recommend that you download the latest installation package.

Linux

1. **Download an installation package for Linux from the *PHP official site*. For example, *php-7.3.8.tar.bz2 (sig)*.**
2. **Open the directory where the installation package resides.**
3. **Use the following command to install the package.**

```
$ sudo yum install php
```

macOS

1. **Download an installation package for macOS from the *PHP official site*. For example, *php-7.3.8.tar.bz2 (sig)*.**
2. **Use the following command to install the package.**

```
$ brew install php
```

Windows

1. **Download an installation package for Windows from the *PHP official website*. For example, *PHP 7.3 (7.3.8)*.**
2. **Open the directory where the installation package resides and double-click the package to install PHP. No further configuration is required.**

Install Composer

Composer is an application-level package manager that helps you manage dependencies. You can use Composer to install libraries, track dependencies for applications, and create projects for popular frameworks.

1. **Use the following command to retrieve a PHP script from `getcomposer.org` to install Composer.**

   ```
   # curl -sS https://getcomposer.org/installer | php
   ```

2. **The installer creates a PHAR file in the current directory. Move the file to a directory that is added to the PATH environment variable. Then, you can run the file under all folders.**

   ```
   $ mv composer.phar ~/.local/bin/composer
   ```

3. **Use the require command to install Twig.**

   ```
   $ composer require twig/twig
   ```

Composer adds libraries that are installed on your localhost to the *composer.json* file of your project. When you deploy an application, Web App Service uses Composer to install libraries that are listed in the file on an application instance of your deployment environment. If you have any questions when you install Compose, visit the *Composer official website*.

Install an IDE

An integrated environment (IDE) is a software application that provides developers with comprehensive tools for software development. Typically, these tools include a code editor, compiler, debugger, and graphical user interface (GUI). These tools help developers significantly improve development efficiency. The following lists IDEs that are commonly used for PHP development:

- *Eclipse*
- *PhpStorm*

6.2 Use the Laravel framework to develop applications

Laravel is a PHP Web framework with an expressive and elegant syntax that is intended for the development of Web applications. This topic describes how to use

Laravel to create an application, link a MySQL database, and deploy the application in Web+.

Prerequisites

Configure a PHP environment

Notice:
Before using the Laravel framework to develop applications, you must install PHP 5.5.9 or later.

Step 1: Create an application

1. **Use the following command to call Composer and create a project named `webplusdemo`. The process requires a few minutes to complete.**

```
composer create-project --prefer-dist laravel/laravel webplusdemo
```

2. **Use the following command to start the PHP built-in Web server to run the project.**

```
php artisan serve
```

3. **Open a browser, and enter the address shown in the following figure to access the application.**

Step 2: Create a deployment package

1. **Navigate to the project directory, and use the following command to enable Laravel built-in authentication components.**

```
php artisan make:auth
```

2. **Specify the corresponding environment variables that are defined in Web+ for database-related settings in the `.env` file.**

```
DB_CONNECTION=mysql
DB_HOST=${WP_RDS_CONNECTION_ADDRESS}
DB_PORT=${WP_RDS_PORT}
DB_DATABASE=${WP_RDS_DATABASE}
DB_USERNAME=${WP_RDS_ACCOUNT_NAME}
```

```
DB_PASSWORD=${WP_RDS_ACCOUNT_PASSWORD}
```

3. **Use the following command to install the required dependencies.**

```
composer install
```

4. **Use the `zip` command to archive all files in the project and create a deployment package named `webplusdemo.zip`.**

```
zip -r webplusdemo.zip . /
```

Step 3: Deploy the application in Web+

1. **Log on to the *Web+ console*.**
2. **On the Overview page, click Create in the upper-right corner of the Last Updated Deployment Environments section.**
3. **In the Basic Information step, select PHP in the Technology Stack Type setting, and enter an application name and application description. After the configuration is complete, click Next.**
4. **In the Deployment Environment Information step, enter a deployment environment name, and select Upload Local Application in the Deployment Package Source setting. Upload the newly compressed deployment package named `webplusdemo.zip`, configure the deployment package version, and click Next.**
5. **In the Configurations step, select Custom in the Predefined Configuration setting.**
6. **View the ApsaraDB for RDS field and select MySQL in the Database Type setting. Then, configure the other required settings, such as the database version, database edition, and instance type.**
7. **View the Lifecycle Hooks field and enter the following command in the PostPrepareApp setting.**

```
cd $APP_HOME && /usr/local/php/bin/php artisan migrate
```

8. **Click Creation Complete at the bottom of the page.**
9. **In the Creation Complete step, click View Application or Creation Complete and go to the Overview tab of the Application Details page. Click the name of a deployment environment to go to the Overview tab of the Deployment Environment Details page. Then, click the link next to the Public IP Address label to view the homepage of the application.**

More information

- **For more information about how to deploy an application in the console, see** *#unique_9*.
- **For more information about how to use the CLI tool to create and deploy an application, see** *#unique_10*.
- **For more information about management tasks after you host an application, see** *#unique_11*.
- **For more information about how to manage deployment environments where applications reside, see** *#unique_12*.

6.3 Use Symfony to develop applications

Symfony is an object-oriented PHP framework that is based on the Model-View-Controller (MVC) pattern. This topic describes how to use Symfony to create an application and deploy the application in Web App Service.

Prerequisites

Configure a PHP environment

Notice:

Before using the Laravel framework to develop applications, you must install PHP 5.5.9 or later.

Step 1: Install Symfony

1. **Use the following command to install the Symfony CLI tool.**

   ```
   curl -sS https://get.symfony.com/cli/installer | bash
   ```

2. **Use the following command to move the executable file of the Symfony CLI tool to a Linux system directory that contains binary files for common commands.**

   ```
   mv ~/.symfony/bin/symfony /usr/local/bin/symfony
   ```

Step 2: Create an application

1. **Use the following command to call the Symfony CLI tool and create a Symfony demo project.**

   ```
   symfony new --demo webplusdemo
   ```

   The process requires a few minutes to complete. After the process is complete, a project named `webplusdemo` **is created.**

2. **Navigate to the project directory and use the following command to install the required dependencies.**

   ```
   composer install
   ```

3. **Use the following command to start the Web server that is included in the Symfony CLI tool.**

   ```
   symfony server:start
   ```

4. **Enter** `http://localhost:8000` **in the address bar of a browser and view the homepage of a sample Symfony application.**

Step 3: Create a deployment package

1. **Web App Service allows you to use NGINX or Apache to serve as a Web server. If you want to use Apache, you can navigate to a project directory and use the**

following command to create a file named `.htaccess`. If you want to use NGINX, you can skip this step.

```
composer require symfony/apache-pack
```

2. Use the following command to archive the directory and create a deployment package named `webplusdemo.zip`.

```
zip -r webplusdemo.zip . /
```

Step 4: Deploy the application in Web App Service

1. Log on to the *Web+ console*.
2. On the Overview page, click Create in the upper-right corner of the Last Updated Deployment Environments section.
3. In the Basic Information step, select PHP in the Technology Stack Type setting, and enter an application name and application description. After the configuration is complete, click Next.
4. In the Deployment Environment Information step, enter a deployment environment name, and select Upload Local Application in the Deployment Environment Source setting. Upload the newly compressed deployment package named `webplusdemo.zip`, configure the deployment package version, and click Creation Complete.
5. In the Creation Complete step, click View Application or Creation Complete to go to the Overview tab of the Application Details page. Click the name of a deployment environment to go to the Overview tab of the Deployment Environment Details page. Then, click the link next to the Public IP Address label and view the homepage of the application.

More information

- For more information about how to deploy an application in the console, see *#unique_9*.
- For more information about how to use the CLI tool to create and deploy an application, see *#unique_10*.
- For more information about management tasks after you host an application, see *#unique_11*.
- For more information about how to manage deployment environments where applications reside, see *#unique_12*.

7 Python

7.1 Configure a Python development environment

Before testing Python applications in a local development environment, you must prepare the development environment. This topic describes how to configure a Python development environment. It also provides download links for related tools.

Install Python

Download an installation package from the *Python official website* based on your operating system.

> **Note:**
> **We recommend that you download *Python 3.7.4* or *Python 2.7.16* because either one is more compatible with the technology stack version of Web App Service.**

Linux

1. **Go to the directory where a Python installation package such as `Python-3.7.4.tgz` resides and use the following command to extract the package.**

   ```
   tar xvf Python-3.7.4.tgz
   ```

2. **Go to the directory where the Python installation package resides and use the following commands to compile and install Python.**

   ```
   ./configure --with-ensurepip=install
   make && make install
   ```

macOS

Use the following `brew` command to install Python.

```
brew update && brew install python
```

Windows

Go to the directory where a Python installation package resides, double-click the installation package, and follow the provided instructions to install Python.

Virtual environments

If you develop multiple projects in parallel, we recommend that you isolate these projects by using Python virtual environments. This method helps you eliminate version conflicts between dependency packages of different projects.

For more information about how to use virtual environments, see *Virtual environments and packages*.

Use an IDE

An integrated development environment (IDE) is a software application that provides developers with comprehensive tools for software development. It also helps developers significantly improve development efficiency. The following lists commonly used IDEs for developing Python applications:

- *Eclipse*
- *PyCharm (for commercial use)*

7.2 Use Flask to develop applications

Flask is a Python-based light-weight Web framework. This topic describes how to use Flask to create an application and deploy the application in Web+.

Step 1: Create an application.

1. **Create a directory named `weplusdemo` and create a file named `application.py` in the directory.**

```
webplusdemo
└── application.py
```

2. **Add the following snippet to the `application.py` file.**

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!
```

Notice:

- **Web+ automatically identifies the `application.py` file and the global variable named `app`. If you want to use a different file name, you must specify the**

required start command in the Procfile file or Web+ console. For more information, see *Use the Procfile file to configure application processes* and *#unique_30*.

- Web+ uses `Gunicorn` as the HTTP server for Flask applications by default.

Step 2: Create a deployment package

Go to the `webplusdemo` directory, and use the following command to create a deployment package named `webplusdemo.zip`.

```
zip -r webplusdemo.zip . /
```

Step 3: Deploy the application in Web+

1. Log on to the *Web+ console*.
2. On the Overview page, click Create in the Last Updated Environments section.
3. In the Basic Information step, select Python in the Technology Stack Type field and enter an application name and description. After the configuration is complete, click Next.
4. In the Environment Information step, enter a environment name, select Upload Local Application in the Deployment Package Source field, and upload the `webplusdemo.zip` deployment package. After configuring the deployment package version, click Creation Complete.
5. In the Creation Complete step, click View Application or Creation Complete to go to the Overview tab of the Application Details page. Click the name of an environment to go to the Overview tab of the Environment Details page. Then, click the link next to the Public IP Address label to view the homepage of the application.

More information

- For more information about how to deploy an application in the console, see *#unique_9*.
- For more information about how to use the CLI tool to create and deploy an application, see *#unique_10*.
- For more information about management tasks after you host an application, see *#unique_11*.
- For more information about how to manage deployment environments where applications reside, see *#unique_12*.

7.3 Use Django to develop applications

Django is a Python-based open-source Web framework. This topic describes how to use Django to create an application, link a MySQL database to the application, and deploy the application in Web+.

Step 1: Install Django

Use the following command to install Django. You need to install the `pymysql` package. It includes MySQL, which will be used later to link the application.

```
pip install Django pymysql
```

Notice:
Before installing Django 2.2 or later, you need to install a version that is later than Python 3.5. This topic takes Python 3.7.4 as an example.

Step 2: Create an application.

1. **Use the following `django-admin` command to create a project.**

   ```
   django-admin startproject webplusdemo
   ```

2. **The structure of the new project folder is listed as follows:**

   ```
   webplusdemo/
   ├── manage.py
   └── webplusdemo
       ├── __init__.py
       ├── settings.py
       ├── urls.py
   ```

```
    └── wsgi.py
```

3. **Use the following** `django-admin` **command to create a project.**

```
django-admin startproject webplusdemo
```

Step 3: Create a deployment package

1. **Use the following command to modify the** `ALLOWED_HOSTS` **setting in the** `settings.py` **file to allow access from all domains.**

```
ALLOWED_HOSTS = ['*']
```

2. **Use the following commands to modify database settings in the** `settings.py` **file. Django uses** `SQLite` **databases by default. This topic takes an ApsaraDB for RDS instance with a MySQL database engine as an example.**

```
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.environ['WP_RDS_DATABASE'],
        'USER': os.environ['WP_RDS_ACCOUNT_NAME'],
        'PASSWORD': os.environ['WP_RDS_ACCOUNT_PASSWORD'],
        'HOST': os.environ['WP_RDS_CONNECTION_ADDRESS'],
        'PORT': os.environ['WP_RDS_PORT'],
    }
}
```

3. **Open the** `webplusdemo` **folder, use the following command to archive the directory and create a deployment package named** `webplusdemo.zip`**.**

```
zip -r webplusdemo.zip . /
```

Step 4: Deploy the application in Web+

1. **Log on to the** *Web+ console*.
2. **On the Overview page, click Create in the Last Updated Environments section.**
3. **In the Basic Information step, select Python in the Technology Stack Type field and enter an application name and description. After the configuration is complete, click Next.**
4. **In the Deployment Information step, enter a environment name, select Upload Local Application in the Deployment Package Source field, and upload the** `webplusdemo.zip` **deployment package. After configuring the deployment package version, click Next.**

5. In the Configurations step, select Custom in the Predefined Configuration setting.
6. View the ApsaraDB for RDS field and select MySQL in the Database Type setting. Then, configure the other required settings, such as the database version, database edition, and instance type.
7. View the Lifecycle Hooks setting, and enter the following command in the PostPrepareApp field. After the configuration is complete, click Creation Complete.

   ```
   source /etc/bashrc && cd $APP_HOME && python manage.py migrate
   ```

8. In the Creation Complete step, click View Application or Creation Complete to go to the Overview tab of the Application Details page. Click the name of an environment to go to the Overview tab of the Environment Details page. Then, click the link next to the Public IP Address label to view the homepage of the application.

   After the database is configured in the preceding steps, you can view the logon page of the application. You can add information about a user to the user table of the database or use `python manage.py createsupersuer` command to create a new user.

More information

- For more information about how to deploy an application in the console, see *#unique_9*.
- For more information about how to use the CLI tool to create and deploy an application, see *#unique_10*.
- For more information about management tasks after you host an application, see *#unique_11*.
- For more information about how to manage deployment environments where applications reside, see *#unique_12*.

8 ASP.NET Core

8.1 Configure an ASP.NET Core environment

Before testing ASP.NET Core applications in a local environment, you must prepare the environment. This topic describes how to configure an ASP.NET Core environment. It also provides download links for related tools.

Install Visual Studio

The Visual Studio family of products provides many easy-to-use features that help you develop ASP.NET Core applications. If you are using macOS or Windows, we recommend that you download Visual Studio from the *Visual Studio* official website and install the tool on your system.

Install an SDK

If you are using Linux, you need to install a .NET Core SDK. The following uses CentOS 7 that has a .NET Core 2.2 SDK installed as an example.

Use the following command to install an SDK. You can also refer to the instructions provided in the *Ubuntu 19.04 Package Manager - Install .NET Core* topic to install an SDK.

```
sudo rpm -Uvh https://packages.microsoft.com/config/centos/7/packages-
microsoft-prod.rpm
sudo yum update
sudo yum install dotnet-sdk-2.2
```

9 Ruby

9.1 Configure a Ruby environment

Before testing Ruby applications in a local environment, you must prepare the environment. This topic describes how to configure a Ruby environment. It also provides download links for related tools.

Install Ruby

You can install Ruby by following the instructions provided in the *Ruby official documentation*. We recommend that you install Ruby 2.6.3. This is a stable version that is supported by Web App Service.

Linux

In Linux, the simplest method to install Ruby is to use a package manager. For example, you can use the following command to install Ruby on CentOS 7.

```
yum install ruby
```

If you want to install Ruby of the latest version or a specified version, you need to download the required source code version. Then, you need to extract the source code file, and use the following commands to compile and install Ruby.

```
./configure
make && make install
```

macOS

Use the following brew command to perform a quick installation of Ruby.

```
brew update && brew install go
```

Windows

If you are using Windows, you can use *RubyInstaller* to install Ruby.

Configure RubyGems

RubyGems is a package manager for Ruby. If you access the source repository of RubyGems from mainland China, the access speed is low. We recommend that you

access mirror sites of the source repository to accelerate the access speed. You can use the following commands to configure RubyGems after Ruby is installed.

```
gem sources --add https://gems.ruby-china.com/ --remove https://
rubygems.org/
bundle config mirror.https://rubygems.org https://gems.ruby-china.com
```

Install an IDE

An integrated environment (IDE) provides comprehensive facilities. It allows developers to develop software and improve overall productivity. RubyMine is a common IDE for developing Ruby applications. We recommend that you download the IDE from the *RubyMine (for commercial use)* link.

10 Native

10.1 Deploy native applications in Web+

Web+ supports deploying applications that are developed in a variety of programming languages. You can deploy a native application if the suitable technology stack is not available. This topic describes how to deploy native applications in Web+.

Introduction to native applications

- **Web+ requires manual installation of infrastructure software. You can customize the installation procedure in the *#unique_30* sections. For example, you can use customized commands in the PostPrepareEnv section to install the required software or dependencies.**
- **Web+ does not provide default start commands. You must configure start commands in the *#unique_30/unique_30_Connect_42_section_pks_hk7_jtq* section or the *Procfile* file.**
- **In Web+, port 8080 is used as the service port of a native application by default. We recommend that you specify a service port that is specific to your environment. You can use the `$WP_SERVICE_PORT` environment variable to configure a service port for an application. Health checks for an application may fail if the service port is different from the listening port.**
- **A native application must be able to run as expected on the `AliyunLinux2.1903` operating system (environment).**
- **You can attach a database to a native application. For more information, see *#unique_18*. You can retrieve database options that are predefined by Web+ from environment variables. To enable database access after configuring database options, you only need to select a compatible database driver. For more information about related environment variables, see *Environment variables*.**

Archive a native application

This topic uses a simple HTTP application named `simpleserver` as an example. This application only includes an executable file. After the application is started, it

listens on port 8080, accepts HTTP GET requests, and returns a message showing "OK".

```
└── simpleserver
```

1. **Create a file named Procfile in the project directory of the application. Add the following statement to the file.**

   ```
   web: ./simpleserver
   ```

2. **Use the following command to archive the application and create a deployment package. The package is used to deploy the application in Web+.**

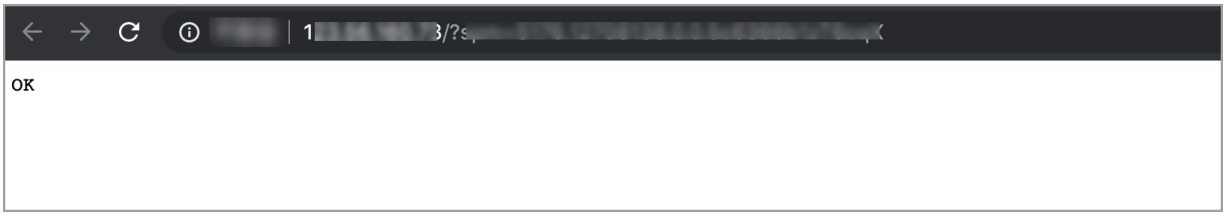
   ```
   zip -r simpleserver.zip . /
   ```

Deploy a native application in Web+

1. **Log on to the *Web+ console*.**
2. **On the Overview page, click Create in the upper-right corner of the Last Updated Environments section.**
3. **In the Basic Information step, select Native in the Tech Stack Type field, enter an application name and description, and then click Next.**
4. **In the Environment Information step, enter an environment name, select Upload Local Application in the Package Source field, upload the deployment package, and then modify the deployment package version. Click Create with Low Cost Preset after the configuration is complete.**

Access an application

In the Creation Complete step, click View Application or Creation Complete to go to the Overview tab of the Application Details page. Click the name of a deployment environment to go to the Overview tab of the Environment Details page. Then, click the link next to the Public IP Address label to visit the homepage of the application.



References

- **To view a demo about how to deploy an application in Web App Service, see** *Create an application and deployment environment in Web App Service*.
- **For more information about how to deploy an application in the console, see** *Deploy an application*.
- **For more information about how to use the CLI tool to create and deploy an application, see** *Use the CLI tool to deploy an application*.
- **For more information about how to manage hosted applications, see** *Overview of application details*.
- **For more information about how to manage deployment environments where applications reside, see** *Overview of deployment environments*.