Application Orchestration Service

Best Practices

Issue	01
Date	2023-05-05

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Contents

1 Application Orchestration Service

1.1 Using AOS and Container Technologies to Quickly Deploy the Magento E-commerce Platform

By using the **Magento E-commerce Application** template, you can quickly deploy a containerized application of the Magento e-commerce platform. The Magento e-commerce website consists of a frontend component and a MySQL database. This section walks you through the steps of quickly deploying the platform by creating a stack from a modified public template **Magento E-commerce Application**. The modified template allows you to specify different request and limit usage of CPU and memory for containers every time you create stacks from this template.

In this section, you will complete the following operations:

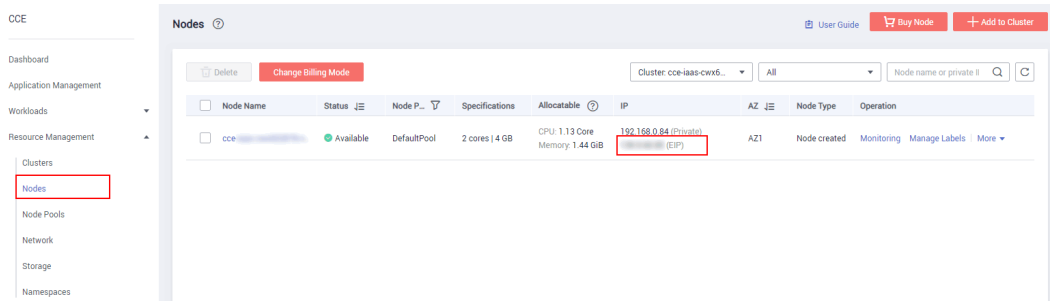
1. **Understanding Basic Concepts**: Before deployment, you are advised to understand the basic concepts of the elements that are involved.
2. **Modifying a Public Template**: Modify the public template **Magento E-commerce Application**. Specifically, add the CPU and memory request and limit parameters for the Magento frontend application.
3. **Creating a Magento E-Commerce Application Stack**: Use the modified template to create a containerized application. Then, deploy the created application on a cluster.
4. **Deleting Resources**: You are advised to delete unnecessary stack resources to avoid unwanted charges.

Prerequisites

You have a HUAWEI CLOUD account. Before deploying a container, ensure that at least one cluster and one node with 2 vCPUs and 4 GB of memory are available.

The node is in the **Available** state and bound to an elastic IP address. To query the node status, log in to the **CCE console** and choose **Resource Management** > **Nodes**. Then, obtain an elastic IP address.

Figure 1-1 Checking whether a node is available



Understanding Basic Concepts

Before modifying a template, understand the basic composition of the **Magento E-commerce Application** template.

```
# Application template information
tosca_definitions_version: huaweicloud_tosca_version_1_0
# Definitions of input parameters
inputs:
  app-name:
    default: magento
    description: 'application name'
    label: magento
  magento-EIP:
    description: 'external access address of the Magento service'
    label: magento
  magento-EPORT:
    default: 32080
    description: 'external listening port of the Magento service'
    label: magento
    type: integer
    ...
# Definition of the mapping table
mappings:
  region_map: # Defines the mapping between images and specifications of different regions.
    cn-east-2:
      magento-image: '10.125.17.64:20202/aos-samples/magento:1.9.1.0'
      mysql-image: '10.125.17.64:20202/aos-samples/mysql:latest'
    ... ...
# Definition of the application topology
node_templates:
  magento: #element name
    metadata:
      Designer:
        id: e66e332a-3466-4638-9896-f7d2e93a1ae3
    properties: #element properties
      k8sManifest:
        apiVersion: apps/v1
        kind: Deployment
        metadata:
          labels:
            app:
              get_input: app-name
          name:
            get_input: app-name
        ...
    requirements: #element dependencies
      - dependency:
          node: mysql-service
      - dependency:
          node: mysql-conf
      - dependency:
          node: magento-config
    type: HuaweiCloud.CCE.Deployment #element type
```



```
  ...
# Definitions of output parameters
outputs:
  ingress-admin_password:
    description: Password of super user.
    value: magentorocks1
  magento-addr:
    description: Access URL for magento service.
    value:
      concat:
        - 'http://'
        - get_input: magento-EIP
        - ':'
        - get_input: magento-EPORT
  magento-admin_username:
    description: Super user name.
    value: admin
```

Description of template properties:

- inputs

  This section is optional and is used to define the variables of a stack created based on a template. A maximum of 60 input parameters can be defined in a template. Each input parameter must have a unique name so that the value can be obtained by using the built-in **get_input** function. If an input parameter is defined more than once, the latest definition will overwrite the previous ones.

  Function scope: **node_templates** and **outputs** sections. That is, input parameters can be transferred in the properties of **node_templates** and value of **outputs**.

  Format of the **inputs** section:

  ```
  <Input parameter name>:
    type: <Type>
    default: <Default value>
    constraints: <Constraints>
    description: <Description>
    label: <Label>
    invisible: <Whether command outputs are visible>
  ```

- mappings

  This section is optional and is used to define a mapping table. When creating a stack based on a template, you can use the **get_in_map** function to extract the content corresponding to a specific input variable. A maximum of 10 mappings can be defined in a template.

  Format of the **mappings** section:

  ```
  <Mapping name>:
    <Mapping object name>:
      <Mapping object property name>: <Mapping object property value>
      <Mapping object property name>: <Mapping object property value>
      ...
    ...
  ...
  ```

- node_templates

  This section is mandatory and is used to define the set of element objects orchestrated in the template. All objects are elements. An element can be an application or a cloud service resource.

  Format of the **node_templates** section:

  ```
  <Element name>:
    type: <Element type>
  ```



```
properties: <Element properties>
requirements: <Element dependency>
condition: <Condition name>
```

  - **Element name**: Each element name must be unique and contains 1 to 48 characters. Only lowercase letters, digits, and hyphens (-) are allowed.
  - **type**: used to specify the type of an orchestration object. The type must be included in the element type list.
  - **properties**: Attribute information is expanded based on element types. Each element type has its corresponding properties. The variable of a property can be obtained from the **inputs** section or from the runtime information displayed after you run the **get_attribute** command. If an element does not require a special property, you do not need to define **properties**.
  - **requirements**: Optional. If there is no relationship between elements, you do not need to define this parameter. The dependency between elements is based on the defined element type. Related dependencies can be defined for specific types.
  - **condition**: Optional. This field determines whether to create elements in the **node_templates**. For details, see the description of conditions.

- outputs

  This section is optional and is used to define the output parameters during the runtime of the stack generated using a template. Each output parameter must have a unique name.

  Format of the **outputs** section:

  ```
  <Output parameter name>:
    description: <Description>
    value: <Value>
  ```

- tosca_definitions_version

  This section is mandatory and is used to specify the version of a template.

For more information about templates, see **Templates (Cloud-based Automation Scripts)**.

Modifying a Public Template

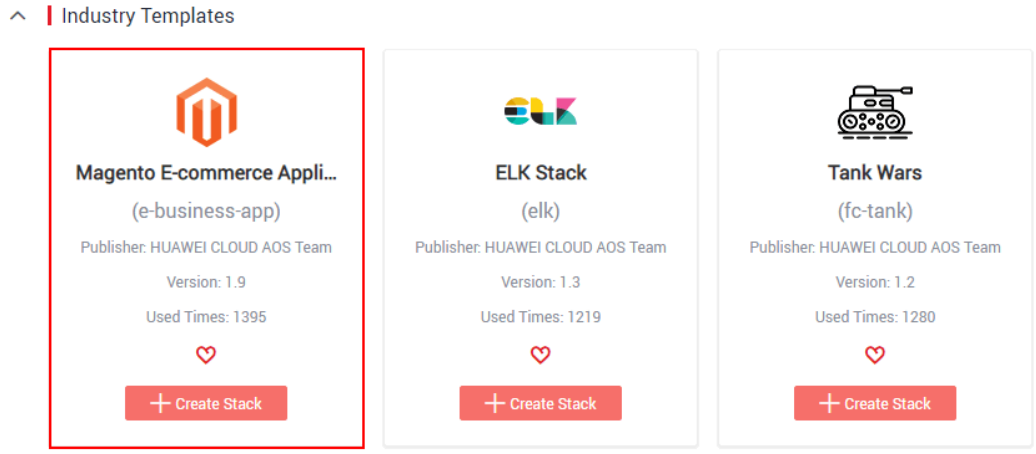
To save container running resources and prevent container overload, you can apply for and limit the CPU and memory used by a container. This section describes how to add the CPU and memory request and limit parameters for the Magento frontend application and MySQL database application by modifying a public template.

- Request: indicates the minimum amount of resources required for running a container.
- Limit: indicates the maximum amount of resources for running a container. You can set the limit to prevent system faults caused by application overload.

Step 1 Log in to the AOS console. Then, in the navigation tree, choose **Template Market** > **Public Templates**.

Step 2 In the **Industry Templates** area, click **Magento E-commerce Application** to view details.

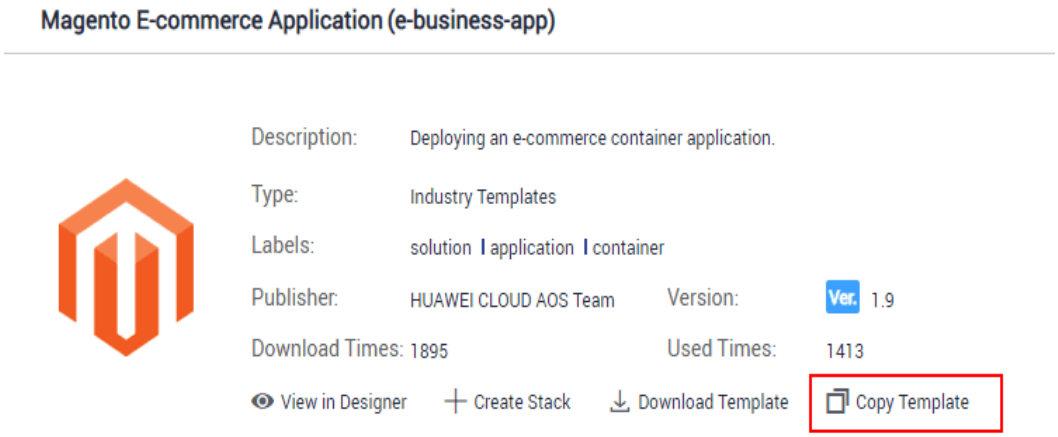
Figure 1-2 Selecting a public template



The template details contain brief information and diagram of the template. A Magento application group contains a Magento frontend application and a MySQL database application. In addition, the Magento application depends on the MySQL application because the former needs to save data to the later.

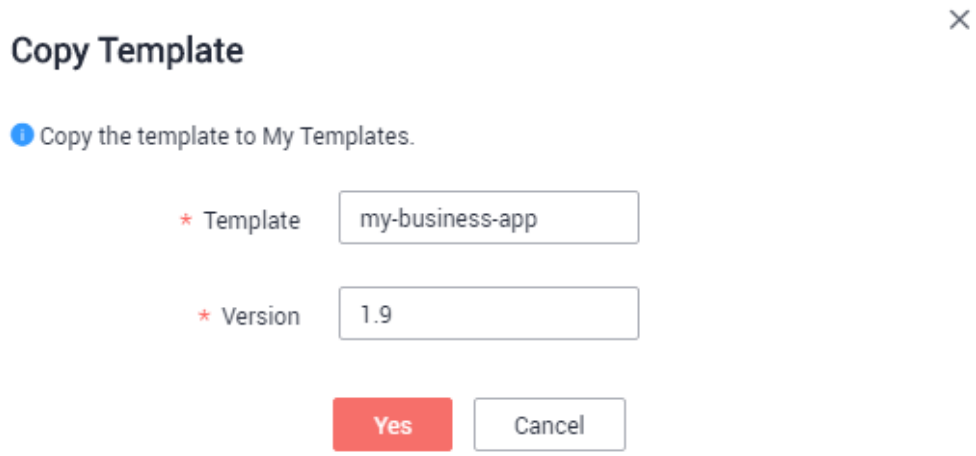
Step 3 Click **Copy Template** to copy the template to **My Templates**.

Figure 1-3 Copying the template



Step 4 Change the template name to **my-business-app** and click **Yes**. The **my-business-app** template details page is displayed.

Figure 1-4 Changing the template name



Step 5 Click **Download** in the **Operation** column and modify the template information based on service requirements. The following describes how to add the CPU and memory request and limit parameters for the Magento frontend application.

```
name: magento-container
resources:
  requests:
    cpu:
      get_input: requestMagentoCPU
    memory:
      get_input: requestMagentoMemory
  limits:
    cpu:
      get_input: limitMagentoCPU
    memory:
      get_input: limitMagentoMemory
ports:
  - containerPort: 80
    protocol: TCP
```

Description of the properties:

- **resources**: indicates container resource specifications.
- **requests**: indicates the quotas of resources allocated to a container.
- **limits**: indicates the maximum amount of resources available for a container.
- **cpu**: indicates the number of CPU cores used by a container.
- **memory**: indicates the memory amount required by a container.
- **get_input**: used to obtain the value of input parameters defined in the **inputs** section of the template

In the **inputs** section, add description and labels to the input parameters.

```
inputs:
  app-name:
    default: magento
    description: application name
    label: magento
  mysql-service-name:
    default: magento-mysql
    description: name of the MySQL service
    label: mysql
  requestMagentoCPU:
    description: CPU request of the Magento service
```



```
  label: magento
requestMagentoMemory:
  description: memory request of the Magento service
  label: magento
limitMagentoCPU:
  description: CPU limit of the Magento service
  label: magento
limitMagentoMemory:
  description: memory limit of the Magento service
  label: magento
magento-EIP:
   description: external access address of the Magento service
  label: magento
```

Description of the properties:

- **inputs**: defines the variables of a stack created based on a template.
- **requestMagentoCPU**: defines the CPU request in the input parameters of the Magento application.
- **requestMagentoMemory**: defines the memory request in the input parameters of the Magento application.
- **limitMagentoCPU**: defines the CPU limit in the input parameters of the Magento application.
- **limitMagentoMemory**: defines the memory limit in the input parameters of the Magento application.
- **description**: indicates the parameter description information.
- **label**: indicates the label of a parameter. The label defined here can be displayed by category during stack creation.

Step 6 Save the template file.

Step 7 On the template details page of **my-business-app**, click **Add Version** to upload the modified template file and set the version number to **2.0**.

----End

Creating a Magento E-Commerce Application Stack

The modified template allows you to set resource limits when creating a stack. You can apply for and limit the CPU and memory of the Magento frontend application and MySQL database application respectively.

- CPU quotas:

  Table 1-1 Description of CPU quotas

Parameter	Description
CPU request	Minimum number of CPU cores required by a container. Resources are scheduled for the container based on this value. However, the requested value does not limit the maximum number of CPU cores available for the container.
CPU limit	Maximum number of CPU cores available for a container.

You are advised to configure the CPU quotas as follows: Actual number of CPU cores available for a node ≥ Sum of CPU limits for all containers of the current instance ≥ Sum of CPU requests for all containers of the current instance. For details about the actual number of CPU cores available for a node, log in to the **CCE console**, choose **Resource Management** > **Nodes**, and obtain **Allocatable CPUs (Cores)** of the corresponding node.

- Memory quotas:

  Table 1-2 Description of memory quotas

Parameter	Description
Memory request	Minimum amount of memory required by a container. Resources are scheduled for the container based on this value.
Memory limit	Maximum amount of memory available for a container. When the memory usage exceeds the configured memory limit, the instance may be restarted, which affects the normal use of applications.

  You are advised to configure the memory quotas as follows: Actual amount of memory available for a node ≥ Sum of memory limits for all containers of the current instance ≥ Sum of memory requests for all containers of the current instance. For details about the actual amount of memory available for a node, log in to the **CCE console**, choose **Resource Management** > **Nodes**, and obtain **Allocatable Memory (GiB)** of the corresponding node.

Step 1 In the navigation pane, choose **My Templates**. You can view the **my-business-app** template in the template list.

Click **Create Stack** next to the **my-business-app** template to create a stack.

Step 2 Set the stack information.

- **Stack Name**: Enter a stack name, for example, **my-magento**.
- **Description**: You can leave it blank.
- **Cluster**: Select a created cluster.
- **Namespace**: Retain the default value.
- On the **magento** tab page, set inputs parameters for the Magento application, as shown in **Figure 1-5**. The parameters in the red box in the following figure are added in the **Modifying a Public Template** part.

Figure 1-5 Input parameters of the Magento application

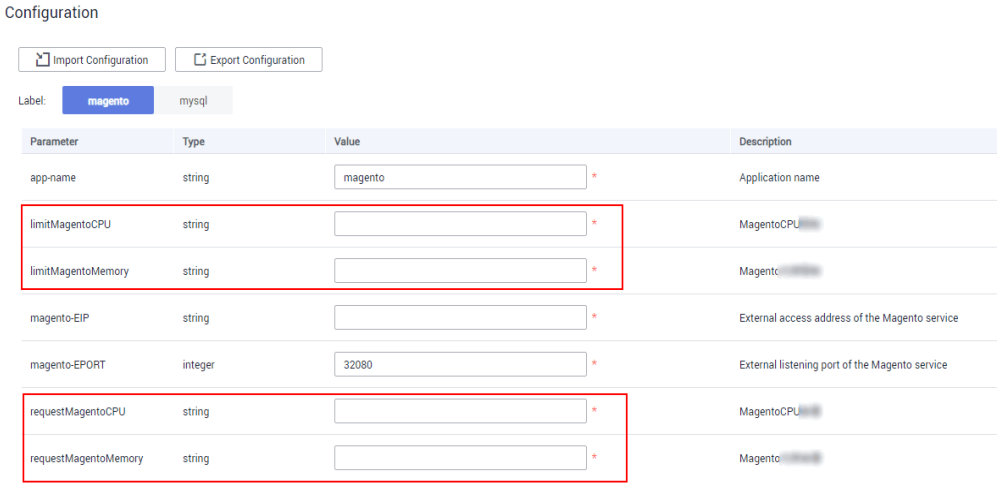


Table 1-3 Setting input parameters of the Magento application

Parameter	Description	Value
limitMagentoCPU	CPU limit of the Magento application. This parameter is added to the input parameters by modifying the public template.	Set this parameter based on actual conditions. The default unit is core and does not need to be added after the value. For example, **1**.
limitMagentoMemory	Memory limit of the Magento application. This parameter is added to the input parameters by modifying the public template.	Set this parameter based on actual conditions. The default unit is M and needs to be added after the value. For example, **2048M**.
magento-EIP	Elastic IP address of a node.	Obtain the elastic IP address from **Prerequisites**, for example, **10.0.0.0**.
magento-EPORT	Port number of a node.	Enter an integer ranging from 30000 to 32767. Ensure that the port number is unique in the cluster. You can retain the default value **32080**.

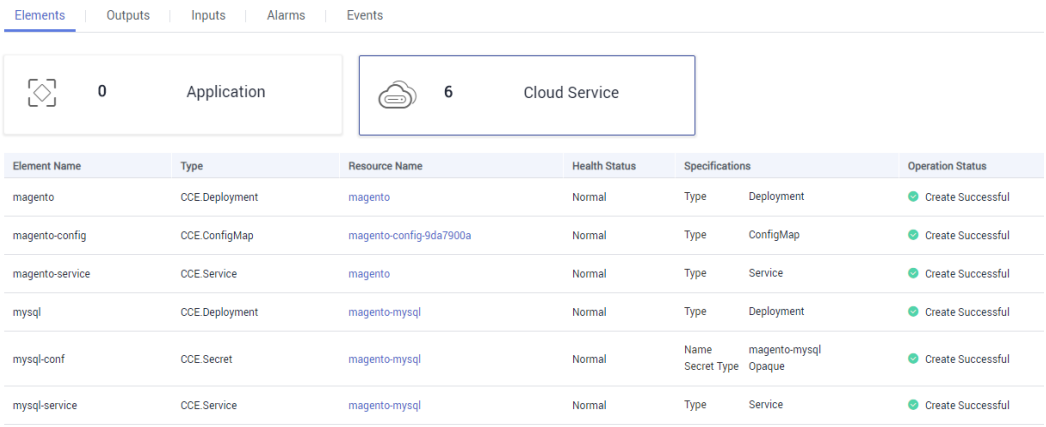
Parameter	Description	Value
requestMagentoCPU	CPU request of the Magento application. This parameter is added to the input parameters by modifying the public template.	Set this parameter based on actual conditions. The default unit is G and does not need to be added after the value. For example, **0.5**.
requestMagentoMemory	Memory request of the Magento application. This parameter is added to the input parameters by modifying the public template.	Set this parameter based on actual conditions. The default unit is M and needs to be added after the value. For example, **1024M**.

Step 3 Click **Next** and check the stack information. If the stack information is correct, click **Create Stack**.

It takes 5 minutes to create the stack.

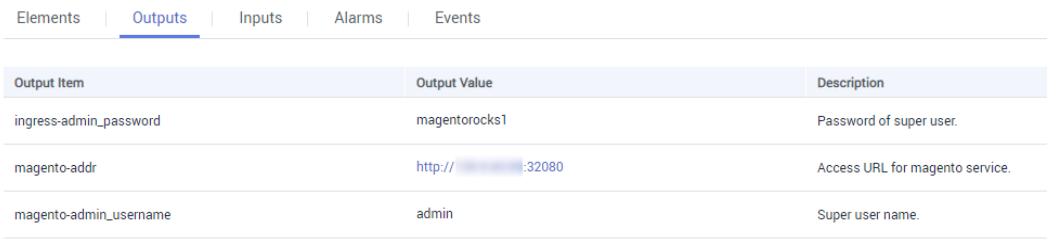
Step 4 After the stack is created, click **Stack Details**. You can view that the stack status is **Normal** and six cloud services exist in **Elements**.

Figure 1-6 Stack created successfully



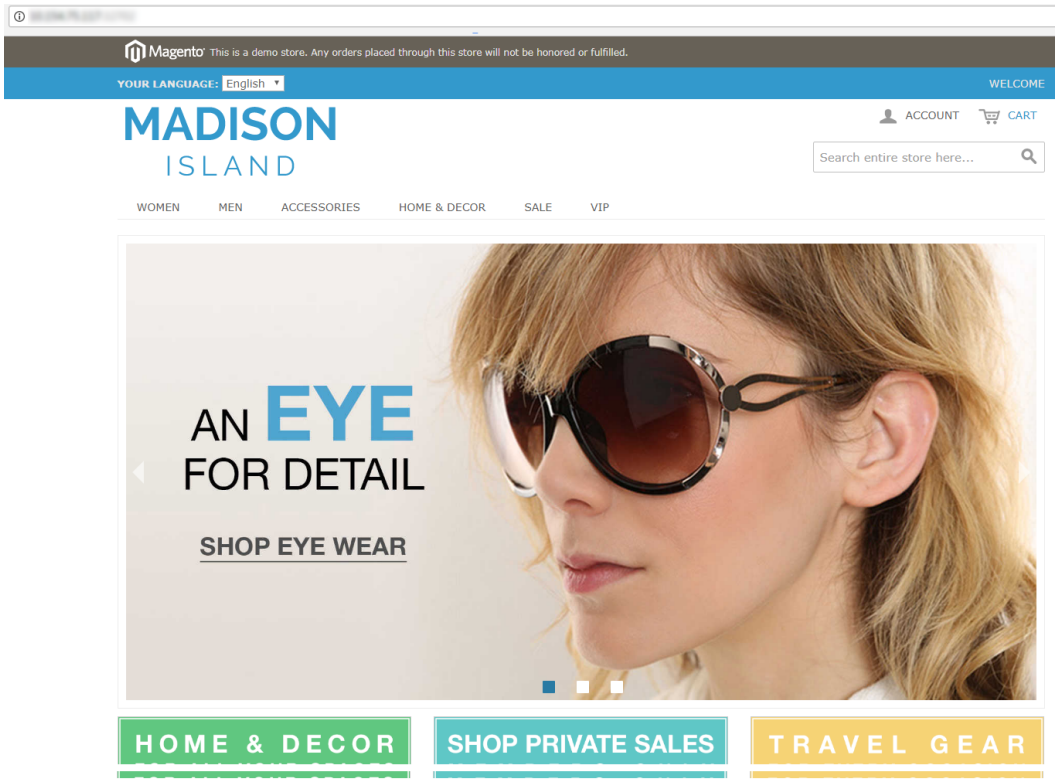
Step 5 On the **Outputs** tab page, view output parameters.

Figure 1-7 Output parameters



Step 6 In the address box of the browser, enter the value of **magento-addr** obtained from the output parameters, for example, **http://10.0.0.0:32080**. Then you can access the Magento application.

Figure 1-8 Accessing the Magento application



----End

Deleting Resources

Delete unnecessary stack resources to avoid unwanted charges.

Step 1 Log in to the AOS console.

Step 2 In the navigation pane, click **My Stacks**.

Step 3 Select the created stack, and click **Delete** to delete the stack as prompted.

----End

