Get Started with Enterprise Tier

Follow up below instructions to get started with Enterprise Tier in Azure Spring Cloud. Table of content is listed as below.

- Pre-requisite
- Provision Service Instance
- Create and Configure apps
- Use Application Configuration Service
- Use Service Registry
- Deploy Apps
- Real-time app log streaming
- Monitor Apps with Application Insights
- Clean up resources

Pre-requisite

- You’ll need an Azure subscription before you begin. If you don’t have one, create an account. For more information, see Marketplace for Enterprise tiers.
- Install the Azure CLI version 2.0.67 or higher
- Install the preview version of Azure Spring Cloud extension for Enterprise tier with command:

```
az extension remove -n spring-cloud
az extension add -s https://ascprivatecli.blob.core.windows.net/enterprise/spring_cloud-2.7.0a1-py3-none-any.whl -y
```

Provision Service Instance

1. Open the Azure portal.

2. From the top search box, search for Azure Spring Cloud.

3. Select Azure Spring Cloud from the results.

4. On the Azure Spring Cloud page, click + Create.
5. Go to the Azure Spring Cloud Create page. In Pricing option, click Change and choose Enterprise tier.

Check Terms checkbox to agree all legal terms and privacy statements of Enterprise tier offering in Azure Marketplace.

Terms

By selecting the checkbox and clicking “Create”, I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the Azure Marketplace Terms for additional details.

6. Click Next: VMware Tanzu settings button at the bottom right of the page to configure VMware Tanzu components.

NOTE

By default, Service Registry and Application Configuration Service is enabled. Suggest to keep the default value. Service Registry and Application Configuration Service cannot be enabled after service instance is provisioned in the current phase.
7. On the Application Insights tab, check **Enable Application Insights** checkbox. Application Insights can also be enabled after service instance is provisioned.

   - Choose an existing Application Insights instance or create a new Application Insights instance.
   - Give a **Sampling Rate** with range 0-100 or use default value 10.

**Note**

   - Please take note of the instance name of Application Insights, which is not shown in portal after provisioning.
   - You will pay for the usage from the instance of Application Insights integrated with Azure Spring Cloud. Check more details about Application Insights pricing information at [Manage usage and costs for Application Insights](#).
8. Click **Review and create** button at the bottom left of the page. After validation is completed successfully, click **Create** button to start provisioning service instance.
9. It takes about 5 minutes to finish the resource provisioning.

Create and Configure Apps

Create apps on Azure Spring Cloud
1. If you didn't run the following commands in the previous quickstarts, set the CLI defaults.

```bash
az configure --defaults group=<resource group name> spring-cloud=<serviceName>
```

2. Create the 2 core microservices for PetClinic: API gateway and customers-service.

```bash
az spring-cloud app create --name api-gateway --instance-count 1 --memory 2Gi --assign-endpoint
az spring-cloud app create --name customers-service --instance-count 1 --memory 2Gi
```

Use Application Configuration Service

**Precondition**

Application Configuration Service should be enabled when service instance is provisioned. It cannot be enabled after provisioning.

1. Open **Application Configuration Service** blade to see the **Overview** tab, which shows the running state and resources allocated to Application Configuration Service.

2. Open **Settings** blade, and add a new entry with below information in **Repositories** section.
   - Name: `default`
   - Patterns: `api-gateway,customers-service`
   - URI: `https://github.com/leonard520/spring-petclinic-microservices-config`
   - Label: `master`

3. Click **Validate** button to validate access to the target URI. After validation is completed successfully, click **Apply** button to update configuration settings.
4. Open **App binding** tab to bind/unbind app to the Application Configuration Service.

   a. Click **Bind app** button and choose one app in the dropdown. Click **Apply** button to bind.
b. The list shows the bound apps with Application Configuration Service:

```
5. Go to Apps blade and choose the pattern(s) to be used by the apps.

a. Open Apps blade to list all the apps.

b. Click api-gateway.

c. Open Configuration menu and General settings tab. Under Config file patterns property, choose one or more patterns from the dropdown list. Click Save button to save the changes.
d. Apply step b & c to **customers-service** and choose the pattern **customers-service**.

### Use Service Registry

**Precondition**

Service Registry should be enabled when service instance is provisioned. It cannot be enabled after provisioning.

1. Open **Service Registry** blade to see the **Overview** tab, which shows the running state and resources allocated to Service Registry.

2. Open **App binding** tab to bind/unbind app to the Service Registry.
a. Click **Bind app** button and choose one app in the dropdown. Click **Apply** button to bind.

b. The list shows the bound apps with Service Registry:

---

**Deploy Apps**
Build the microservices apps locally

1. Clone the sample app repository to your Azure Cloud account. Change the directory, and build the project.

   ```
   git clone -b xiading/enterprise https://github.com/leonard520/spring-petclinic-microservices.git
   cd spring-petclinic-microservices
   mvn clean package -DskipTests
   ```

   Compiling the project takes 5 - 10 minutes. Once compilation is completed, you will have individual JAR files for each service in their respective folders.

   ```
   <dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-config</artifactId>
   </dependency>
   ```

2. Deploy the JAR files built in the previous step.

   ```
   az spring-cloud app deploy --name api-gateway --artifact-path spring-petclinic-api-gateway/target/spring-petclinic-api-gateway-2.3.6.jar
   az spring-cloud app deploy --name customers-service --artifact-path spring-petclinic-customers-service/target/spring-petclinic-customers-service-2.3.6.jar
   ```

3. Query app status after deployments with the following command.

   ```
   az spring-cloud app list -o table
   ```

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>ResourceGroup</th>
<th>Public Url</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production Deployment</td>
<td>eastus</td>
<td>&lt;resource group&gt;</td>
<td>https://&lt;service name&gt;</td>
</tr>
<tr>
<td>Instance</td>
<td></td>
<td></td>
<td>Succeeded</td>
</tr>
<tr>
<td>Registered Instance</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Provisioning State</td>
<td></td>
<td></td>
<td>2Gi</td>
</tr>
<tr>
<td>CPU</td>
<td></td>
<td></td>
<td>1/1</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td></td>
<td>1/1</td>
</tr>
<tr>
<td>Running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bind Application</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration Service</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Succeeded</td>
<td></td>
<td>default</td>
<td></td>
</tr>
</tbody>
</table>
Verify the microservices

Access the app gateway and customers service from browser with the Public Url shown above, in the format of https://<service name>-api-gateway.azuremicroservices.io.

Real-time app log streaming

Use the following command to get real-time logs from the app.

```bash
az spring-cloud app logs -n <app-name> -s <service-instance-name> -g <resource-group> --lines 100 -f
```

This will return logs:

```
2021-07-15 01:54:40.481  INFO [auth-service,,,] 1 --- [main]
o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-07-15 01:54:40.482  INFO [auth-service,,,] 1 --- [main]
org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.22]
```
Monitor Apps with Application Insights

Read more in Monitor Apps with Application Insights

Clean up resources

1. Open the Azure portal. Delete the service instance as below screenshot.

![Image of Azure portal with service instance DELETE option]

2. Run below command to remove the preview version of CLI extension.

```bash
az extension remove -n spring-cloud
```

Azure Marketplace

Overview

Azure Spring Cloud Enterprise Tier is optimized for the needs of enterprise Spring developers through advanced configurability, flexibility, portability, and enterprise-ready VMware Spring Runtime 24x7 support. Developers also benefit from proprietary Tanzu components such as Tanzu Build Service, Tanzu Application Configuration Service and Tanzu Service Registry; and access to Spring experts. Customers obtain and pay for a license to Tanzu components through an Azure Marketplace offer. Azure Spring Cloud manages the license acquisition so that you won’t have to do it separately. To purchase in the Azure Marketplace, you must meet the following prerequisites:

- Your Azure subscription has a valid payment instrument. Azure credits or free MSDN subscriptions aren’t supported.
- Your organization allows Azure Marketplace purchases.
- Your private Azure Marketplace must contain the Azure Spring Cloud Enterprise Tier w/VMware Tanzu offer.

This document guides you how to include Azure Spring Cloud Enterprise Tier w/VMware Tanzu offer to your private Azure Marketplace and how to redirect to Azure Spring Cloud Enterprise tier creation page from Azure...
Marketplace.

View Azure Spring Cloud Enterprise Tier w/VMware Tanzu offer from your private Azure Marketplace

Visit [Azure Spring Cloud Enterprise Tier w/VMware Tanzu](#), you can see the offer and read detailed description of the offer.

Click "Plans + Pricing", then you will see the supported plans in your market.

If you see "No plans are available for market '<Location>'", that means none of your Azure subscription can purchase the SaaS offer. See "No plans are available for market '<Location>'" for more details.

Click "Set up + subscribe", it will redirect you to the Enterprise tier creation page.

---

Build Service
Overview

In Azure Spring Cloud, the existing Standard tier already supports compiling user source code into OCI images through kpack - a Kubernetes (K8s) implementation of Cloud Native Buildpacks (CNB) provided by VMware. In Enterprise tier, more functionalities and configurations are exposed. Read more details in the following sections.

Build Agent Pool

Build Service in Enterprise tier will play as the entry point to containerize user application from both source codes and artifacts. There is a dedicated build agent pool that reserves compute resources for a given number of concurrent build tasks. So that there will not be resource contention with your running apps. For now, 2 vCPU and 4 Gi memory are allocated to the build agent pool. In the coming releases after Private Preview, you will be able to configure the amount of the resources allocated to build agent pool to accommodate number of concurrent build tasks you want.

Tanzu Buildpacks

On top of the open-source Paketo Buildpacks provided in Standard tier, a list of proprietary Tanzu Buildpacks are available by default in Enterprise tier. Tanzu Buildpacks make it easier to integrate with other software like New Relic etc.. They are configured as optional and will only run with proper configuration. More details are explained in Buildpacks Bindings section.

Below is the complete list of all proprietary Tanzu Buildpacks.

- tanzu-buildpacks/apache-skywalking
- tanzu-buildpacks/appdynamics
- tanzu-buildpacks/aspectj
- tanzu-buildpacks/checkmarx
- tanzu-buildpacks/contrast-security
- tanzu-buildpacks/dynatrace
- tanzu-buildpacks/elasticsearch
- tanzu-buildpacks/icu
- tanzu-buildpacks/jacoco
- tanzu-buildpacks/jprofiler
- tanzu-buildpacks/jrebel
- tanzu-buildpacks/new-relic
- tanzu-buildpacks/node-engine
- tanzu-buildpacks/overops
- tanzu-buildpacks/snyk
- tanzu-buildpacks/synopsys
- tanzu-buildpacks/yourkit

You can find details about each of them in https://docs.pivotal.io/tanzu-buildpacks/partner-integrations/partner-integration-buildpacks.html

**Real-time Build Logs**

A build task will be triggered when an app is deployed from Azure CLI command. Build logs are streamed out in real-time as part of the CLI command output. Each build task consists of the following phases: **prepare, detect, restore, analyze, build, export** and **completion**.

Here is a screenshot of sample build logs.

![Sample Build Logs](image)

**Buildpacks Bindings**

Kpack Images can be configured with Service Bindings as described in the [Cloud Native Buildpacks Bindings specification](https://docs.pivotal.io/tanzu-buildpacks/cloud-native-bindings.html). We leverage this Binding to integrate with [Tanzu Partner Buildpacks](https://docs.pivotal.io/tanzu-buildpacks/partner-integrations/partner-integration-buildpacks.html). For example, we use Binding to integrate [Azure Application Insights](https://docs.pivotal.io/tanzu-buildpacks/cloud-native-bindings.html) by paketo-buildpacks/azure-application-insights.

For integration with [Azure Application Insights](https://docs.pivotal.io/tanzu-buildpacks/cloud-native-bindings.html), see [Monitor Apps with Application Insights](https://docs.pivotal.io/tanzu-buildpacks/cloud-native-bindings.html).

**Application Configuration Service**

**Overview**
Application Configuration Service is one of the proprietary VMware Tanzu components. It enables the management of Kubernetes-native ConfigMap resources that are populated from properties defined in one or more Git repositories. With Application Configuration Service, you have a central place to manage external properties for applications across all environments.

Prerequisites

- An already provisioned Azure Spring Cloud Enterprise tier service instance with Application Configuration Service enabled. See [get-started](#).

Manage Application Configuration Service settings

Application Configuration Service supports Azure DevOps, GitHub, GitLab, and Bitbucket for storing your config file.

1. Open **Settings** blade, and add a new entry with below information in **Repositories** section.

Terminology

<table>
<thead>
<tr>
<th>Property</th>
<th>Required</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>Unique name to label each git repository.</td>
</tr>
</tbody>
</table>
| Patterns | Yes      | Use patterns to search in Git repositories. For each pattern, use format like 
application or application/profile instead of [application]-[profile].yml, and separate them by comma. See [Patterns](#) for more detail explanation. |
| URI      | Yes      | Git URI (e.g. https://github.com/Azure-Samples/piggymetrics-config,  
git@https://github.com:Azure-Samples/piggymetrics-config) |
| Label    | Yes      | branch name to search in the Git repository. |
### Pattern

Configuration will be pulled from git backends using what it defined in a pattern which is combined of (application)/(profile).

- **(application)** - The name of an application for which the configuration is being retrieved. If “application”, then this is considered the default application and includes configuration shared across multiple applications. Any other value specifies a specific application and will include properties for both the specified application as well as shared properties for the default application.
- **(profile)** - Optional. The name of a profile for which properties may be retrieved. If “default” or empty, then this includes properties that are shared across any all profiles. If any non-default value, then include properties for the specified profile as well as properties for the default profile.

### Authentication

Application Configuration Service supports three repository types. Please check the detail below.

- **Public repository**

You don’t need extra Authentication configuration when using a public repository but just select **Public** in the Authentication form.

- **Private repository with basic authentication**

All configurable properties used to set up private Git repository with basic authentication are listed below.

<table>
<thead>
<tr>
<th>Property</th>
<th>Required</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Yes</td>
<td>The username used to access the repository.</td>
</tr>
</tbody>
</table>
All configurable properties used to set up private Git repository with SSH are listed in the following table:

<table>
<thead>
<tr>
<th>Property</th>
<th>Required</th>
<th>Explaination</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Yes</td>
<td>The password used to access the repository.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Private repository with SSH authentication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Required</th>
<th>Explaination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private key</td>
<td>Yes</td>
<td>The private key that identifies the Git user. Passphrase-encrypted private</td>
</tr>
<tr>
<td></td>
<td></td>
<td>keys are not supported.</td>
</tr>
<tr>
<td>Host key</td>
<td>No</td>
<td>The host key of the Git server. If you have connected to the server via git</td>
</tr>
<tr>
<td></td>
<td></td>
<td>on the command line, this is in your .ssh/known_hosts. Do not include the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>algorithm prefix; this is specified in <code>Host key algorithm</code>.</td>
</tr>
<tr>
<td>Host key algorithm</td>
<td>No</td>
<td>The algorithm of hostKey: one of &quot;ssh-dss&quot;, &quot;ssh-rsa&quot;, &quot;ecdsa-sha2-nistp256&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;ecdsa-sha2-nistp384&quot;, and &quot;ecdsa-sha2-nistp521&quot;. (Required if supplying</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>Host key</code>).</td>
</tr>
<tr>
<td>Strict host key</td>
<td>No</td>
<td>Whether or not the backend should be ignored it encounters an error</td>
</tr>
<tr>
<td>checking</td>
<td></td>
<td>when using the provided <code>Host key</code>. (Optional.) Valid values are true and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false. Default is true.</td>
</tr>
</tbody>
</table>

2. Click **Validate** button to validate access to the target URI. After validation is completed successfully, click **Apply** button to update configuration settings.

Use Application Configuration Service with Apps

**Restriction**

There are some restrictions when you use Application Configuration Service with a Git back end.
⚠️ Please DON'T include `spring-cloud-starter-config` in the app's pom.xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
</dependency>

You have to bind the app to Application Configuration Service to claim that the app will use the centralized configurations. After that, you will need to configure which pattern to be used by the app.

1. Open **App binding** tab to bind/unbind app to the Application Configuration Service.

2. Click **Bind app** button and choose one app in the dropdown. Click **Apply** button to bind.

3. Go to **Apps** blade and choose the **pattern(s)** to be used by the apps.

**Note**

When you change the bind/unbind status, you have to restart or redeploy the app to take effective.
a. Open **Apps** blade to list all the apps.

![Apps Blade](image)

b. Click target app to configure patterns.

c. Open **Configuration** menu and **General settings** tab. Under **Config file patterns** property, choose one or more patterns from the dropdown list. Click **Save** button to save the changes.

![Configuration Page](image)

---

**Service Registry**

**Overview**

**Service Registry** is one of the proprietary VMware Tanzu components. It provides your apps with an implementation of the Service Discovery pattern, one of the key tenets of a microservice-based architecture. Trying to hand-configure each client of a service or adopt some form of access convention can be difficult and prove to be brittle in production. Instead, your apps can use the Service Registry to dynamically discover and call registered services.

**Prerequisites**

- An already provisioned Azure Spring Cloud Enterprise tier service instance with Service Registry enabled. See [get-started](#)
Use Service Registry with Apps

Before your application can manage service registration and discovery using Service Registry, several dependencies must be included in the application's pom.xml file.

```xml
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

Finally, add an annotation to the top level class of your application

```java
@SpringBootApplication
@EnableEurekaClient
public class DemoApplication {
  public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
  }
}
```

1. Open App binding tab to bind/unbind app to the Service Registry.
2. Click **Bind app** button and choose one app in the dropdown. Click **Apply** button to bind.

---

**Deploy Azure Spring Cloud in a virtual network**

This tutorial explains how to deploy an Azure Spring Cloud instance in your virtual network. Please refer to [Deploy Azure Spring Cloud in a virtual network](#) for details.

**Note:** Please go to this [Azure portal](#) to deploy an Azure Spring Cloud Enterprise instance.

The **Networking** tab of Azure Spring Cloud **Create** page is as below:
App and Deployment management

- **Portal**
  - Create Apps
  - App overview and instance list
  - App configuration
  - Scale up and scale out
- **Azure CLI**

**Portal**

**Create Apps**

1. Open the [Azure portal](https://portal.azure.com).

2. Go to the Azure Spring Cloud Apps page. Click **Create App** button to create apps.
3. Add a new entry in **Create App** blade with app name, vCPU, memory and instance count. You can add more entries by clicking **+ Add app** link to create more apps at one time. Then click **Create** button.

App overview and instance list

1. Once the app is created, open the app overview page by clicking the app name in **Apps** blade.
In the app overview page, you can find a set of information of the app, including resource basic information, provisioning state, public endpoint and test endpoint.

You can also make some operations on the app in the overview page, such as stop, start and restart the app, assign public endpoint to the app.

Some general metrics are listed at bottom of the page, you can learn the app health state from the charts.

2. To see all instance of the app, go to **App Instance** page.
App configuration

**Configuration** blade enables you to update config file patterns, JVM options (if your app is built with Java), environment variables. You can also view temporary storage information in this page.

1. General settings
2. Environment variables

Environment variables are encrypted at rest and transmitted over an encrypted channel. The app must be restarted after setting new environment variables.

- Reveal all values
- Hide all values

Key | Value
--- | ---

3. Temporary storage

Temporary storage is automatically assigned to your app. Don't use temporary storage for persistent data.

Path | /tmp
--- | ---
Size | 5 GB

Scale up and scale out

1. To scale up/down your app by updating vCPU and memory, open **Scale up** blade.
2. Open **Scale out** blade to change instance count of the app.

---

**Azure CLI**

**Monitor Apps with Application Insights**
This article explains how to monitor apps and microservices by using the Application Insights Java agent in Azure Spring Cloud.

With this feature you can:

- Search tracing data with different filters.
- View dependency map of microservices.
- Check request performance.
- Monitor real-time live metrics.
- Check request failures.
- Check app metrics.
- Check app log.

Application Insights provide many observable perspectives, including:

- Application map
- Performance
- Failures
- Metrics
- Live Metrics
- Log
- Availability

Using the Application Insights feature

When the **Application Insights** feature is enabled, you can:

Go to the **Overview** page of Application Insights.

- You can see overview of all apps.

- Click the **Application Map** to see the status of calls between apps.
• Click the testdb to see all the apps connecting to database.

• Click an endpoint to see all apps requesting to the endpoint.
• In the left navigation pane, click **Live Metrics** to see the real time metrics of all apps' in last 60 seconds.

• In the left navigation pane, click **Failures** to see if something unexpected of dependencies from your apps.
• In the left navigation pane, click **Failures** to see if something unexpected or exceptions from your apps.

• In the left navigation pane, click **Performance** to see the performance data of all apps’ operations.
In the left navigation pane, click **Performance** to see the performance data of all apps’ dependencies.

In the left navigation pane, click Metrics and select the namespace, you will see both Spring Boot metrics and custom metrics, if any.
In the left navigation pane, click **Log** to check logs of one(filter by cloud RoleName) or all apps.

Use Azure CLI to manage settings of Application Insights

We use **Buildpacks Binding** to integrate **Azure Application Insights** with type **ApplicationInsights**.

- Create Application Insights buildpacks binding.

```bash
az spring-cloud build-service buildpacks-binding create --name {your-binding-name} \
    --type ApplicationInsights \
    --properties sampling-rate={your-sampling-rate} \
    --secrets connection-string={your-connection-string} \
    -s ${asc-resource-name} -g ${resource-group-name}
```
- Replace Application Insights buildpacks binding.

```bash
az spring-cloud build-service buildpacks-binding set --name {your-binding-name} \
  --type ApplicationInsights \n  --properties sampling-rate={your-sampling-rate} \n  --secrets connection-string={your-connection-string} \n  -s ${asc-resource-name} -g ${resource-group-name}
```

- Get Application Insights buildpacks binding.

```bash
az spring-cloud build-service buildpacks-binding show --name {your-binding-name} \
  -s ${asc-resource-name} -g ${resource-group-name}
```

- Delete Application Insights buildpacks binding.

```bash
az spring-cloud build-service buildpacks-binding delete --name {your-binding-name} \
  -s ${asc-resource-name} -g ${resource-group-name}
```

## Concept mapping between Azure Spring Cloud and Application Insights

<table>
<thead>
<tr>
<th>Azure Spring Cloud</th>
<th>Application Insights</th>
</tr>
</thead>
<tbody>
<tr>
<td>App</td>
<td></td>
</tr>
<tr>
<td>* Application Map/Role</td>
<td>* Application Map/Role Instance</td>
</tr>
<tr>
<td>* Live Metrics/Role</td>
<td>* Live Metrics/Service Name</td>
</tr>
<tr>
<td>* Failures/Roles/Cloud Role</td>
<td>* Failures/Roles/Cloud Instance</td>
</tr>
<tr>
<td>* Performance/Roles/Cloud Role</td>
<td>* Performance/Roles/Cloud Instance</td>
</tr>
</tbody>
</table>

The name **App Instance** from Azure Spring Cloud will be changed or generated in the following scenarios:

- You create a new app.
- You deploy a JAR file or source code to an existing app.
- You initiate a blue/green deployment.
- You restart the app.
- You stop the deployment of an app, and then restart it.

When data is stored in Application Insights, it contains the history of Azure Spring Cloud app instances created or deployed since the Java agent was enabled. This means that, in the Application Insights portal, you
can see app data created yesterday, but then deleted within a specific time range, like the last 24 hours. The following scenarios show how this works:

- You created an app around 8:00 AM today from Azure Spring Cloud with the Java agent enabled, and then you deployed a JAR file to this app around 8:10 AM today. After some testing, you change the code and deploy a new JAR file to this app at 8:30 AM today. Then, you take a break, and when you come back around 11:00 AM, you check some data from Application Insights. You will see:
  - Three instances in Application Map with time ranges in the last 24 hours, as well as Failures, Performance, and Metrics.
  - One instance in Application Map with a time range in the last hour, as well as Failures, Performance, and Metrics.
  - One instance in Live Metrics.
- You created an app around 8:00 AM today from Azure Spring Cloud with the Java agent enabled, and then you deployed a JAR file to this app around 8:10 AM today. Around 8:30 AM today, you try a blue/green deployment with another JAR file. Currently, you have two deployments for this app. After a break around 11:00 AM today, you want to check some data from Application Insights. You will see:
  - Three instances in Application Map with time ranges in the last 24 hours, as well as Failures, Performance, and Metrics.
  - Two instances in Application Map with time ranges in last hour, as well as Failures, Performance, and Metrics.
  - Two instances in Live Metrics.

**Frequently Asked Questions**

**Fail to deploy apps**

```bash
az spring-cloud app deploy --name <app name> <app.jar>
```

112404: Failed to wait for deployment instances to be ready. Please check the application log (see https://aka.ms/azure-spring-cloud-doc-log ), and try again later.

Please check if the below error messages in the application log.

1. Spring Boot 2.4 and higher versions

```
Application failed to start due to an exception
org.springframework.cloud.commons.ConfigDataMissingEnvironmentPostProcessor$ImportException: No spring.config.import set
```

2. Spring Boot 2.3 and earlier versions

```
WARN c.c.c.ConfigServicePropertySourceLocator : Could not locate PropertySource: I/O error on GET request for
```
It is probably caused by adding spring-cloud-starter-config starter by mistake. Please remove the dependency and try again.

112034: Failed to provision resource.

When failed to create Azure Spring Cloud Enterprise tier instance with error message "112034: Failed to provision resource."
. Check whether your Azure subscription's billing account address is in the supported location. See "No plans are available for market '<Location>'" for more details.

If that doesn't help, you can contact support team with the following info.

```
AZURE_TENANT_ID=<your Azure Tenant Id which host the Azure subscription>
AZURE_SUBSCRIPTION_ID=<your Azure subscription Id which is used to create Spring Cloud instance>
SPRING_CLOUD_NAME=<the failed instance name>
```

No plans are available for market '<Location>'

When you visit SaaS offer Azure Spring Cloud Enterprise Tier w/VMware Tanzu in Marketplace, it may say "No plans are available for market '<Location>'" like the following image.

![Azure Spring Cloud Enterprise Tier w/VMware Tanzu](https://example.com)

Azure Spring Cloud Enterprise tier needs customers to pay for a license to Tanzu components through an Azure Marketplace offer. To purchase in Azure Marketplace, the billing account's country or region for your Azure subscription should be in the SaaS offer's support geographic locations.

Azure Spring Cloud Enterprise Tier w/VMware Tanzu now supports all geographic locations that Azure Marketplace supports. See Supported geographic locations.

You can view your billing account for your subscription if you have admin access, see view billing accounts.

Azure Spring Cloud Enterprise tier is free for private preview.

Azure CLI Command Reference
This reference is part of the spring-cloud extension for Azure CLI and requires version 2.0.67 or higher. Install the extension by running:

```bash
az extension add -s https://ascprivatecli.blob.core.windows.net/enterprise/spring_cloud-2.7.0a1-py3-none-any.whl -y
```

Commands to manage Azure Spring Cloud Enterprise tier.

## Commands

### az spring-cloud

- **az spring-cloud list** - List all Azure Spring Cloud in the given resource group, otherwise list the subscription's.
- **az spring-cloud show** - Show the details for an Azure Spring Cloud.
- **az spring-cloud delete** - Delete an Azure Spring Cloud.

### az spring-cloud app

- **az spring-cloud app create** - Create a new app with a default deployment in the Azure Spring Cloud.
- **az spring-cloud app delete** - Delete an app in the Azure Spring Cloud.
- **az spring-cloud app deploy** - Deploy source code or pre-built binary to an app and update related configurations.
- **az spring-cloud app list** - List all apps in the Azure Spring Cloud.
- **az spring-cloud app logs** - Show logs of an app instance, logs will be streamed when setting `-f/--follow`.
- **az spring-cloud app restart** - Restart instances of the app, default to production deployment.
- **az spring-cloud app scale** - Manually scale an app or its deployments.
- **az spring-cloud app show** - Show the details of an app in the Azure Spring Cloud.
- **az spring-cloud app start** - Start instances of the app, default to production deployment.
- **az spring-cloud app stop** - Stop instances of the app, default to production deployment.
- **az spring-cloud app update** - Update configurations of an app.

### az spring-cloud test-endpoint

- **az spring-cloud test-endpoint disable** - Disable test endpoint of the Azure Spring Cloud.
- **az spring-cloud test-endpoint enable** - Enable test endpoint of the Azure Spring Cloud.
- **az spring-cloud test-endpoint list** - List test endpoint keys of the Azure Spring Cloud.
- **az spring-cloud test-endpoint renew-key** - Regenerate a test-endpoint key for the Azure Spring Cloud.

### az spring-cloud build-service buildpacks-binding

- **az spring-cloud build-service buildpacks-binding create** - Create a buildpacks binding.
- **az spring-cloud build-service buildpacks-binding set** - Set a buildpacks binding.
- **az spring-cloud build-service buildpacks-binding show** - Show a buildpacks binding, the secrets will be masked.
- **az spring-cloud build-service buildpacks-binding delete** - Delete a buildpacks binding.
• **az spring-cloud application-configuration-service clear** - Reset all Application Configuration Service settings.

• **az spring-cloud application-configuration-service git repo add** - Add an item of git property to Application Configuration Service settings.

• **az spring-cloud application-configuration-service git repo update** - Update an existing item of git property to Application Configuration Service settings.

• **az spring-cloud application-configuration-service git repo list** - List all git settings of Application Configuration Service.

• **az spring-cloud application-configuration-service git repo remove** - Delete an existing item of git property to Application Configuration Service settings.

• **az spring-cloud application-configuration-service show** - Show provisioning status, runtime status and settings of Application Configuration Service.

• **az spring-cloud application-configuration-service bind** - Bind app to Application Configuration Service.

• **az spring-cloud application-configuration-service unbind** - Unbind app to Application Configuration Service.

```
az spring-cloud service-registry
```

• **az spring-cloud service-registry show** - Show provisioning status and runtime status of Service Registry.

• **az spring-cloud service-registry bind** - Bind app to Service Registry.

• **az spring-cloud service-registry unbind** - Unbind app to Service Registry.