Azure Credentials plugin

Plugin Information
View Azure Credentials on the plugin site for more information.

Azure Credentials Plugin

Jenkins plugin to manage Azure credentials.

- General information on how to use credentials in Jenkins

It supports the following Azure credential types:

1. Azure Service Principal, with the following authentication mechanism:
    - Client secret
    - Certificate (Add the certificate to Jenkins credentials store and reference it in the Azure Service Principal configuration)
2. Azure Managed Service Identity (MSI)
3. Credentials In Azure Key Vault

Using Azure credentials in your own Jenkins plugin

1. Update your project POM file to reference `azure-credentials` plugin and necessary dependencies:

```
...
<dependencies>
   <dependency>
       <groupId>org.jenkins-ci.plugins</groupId>
       <artifactId>azure-credentials</artifactId>
       <version>${azure-credentials.version}</version>
   </dependency>
   <dependency>
       <groupId>org.jenkins-ci.plugins</groupId>
       <artifactId>azure-commons-core</artifactId>
       <version>${azure-commons.version}</version>
   </dependency>
   ...
</dependencies>
<build>
   <plugins>
       <plugin>
           <groupId>org.jenkins-ci.tools</groupId>
           <artifactId>maven-hpi-plugin</artifactId>
           <configuration>
               <maskClasses>
                   com.microsoft.jenkins.azurecommons.core.
               </maskClasses>
            </configuration>
       </plugin>
    </plugins>
    ...
</build>
```

2. Add the credential selector in the `config.jelly` and `Descriptor`

```
...
<f:entry title="${%azureCredentialsId_title}" field="
azureCredentialsId">
    <c:select expressionAllowed="false"/>
</f:entry>
...
```

```
 public ListBoxModel doFillAzureCredentialsIdItems(@AncestorInPath
Item owner) {
     StandardListBoxModel model = new StandardListBoxModel();
     model.add(Messages.
ACSDeploymentContext_selectAzureCredentials(), Constants.
INVALID_OPTION);
     model.includeAs(ACL.SYSTEM, owner, AzureBaseCredentials.
class);
     return model;
 }
```

3. Build the Azure client from the credential

```
AzureBaseCredentials credential = AzureCredentialUtil.
getCredential2(credentialsId);
// Resolve the class loader incompatibility issue. Works along
with maskClasses in the POM
TokenCredentialData token = TokenCredentialData.deserialize
(credential.serializeToTokenData());
Azure azClient = AzureClientFactory.getClient(token);
```

Getting an iterator to all SYSTEM owned Azure Credentials

```
 CredentialsProvider.lookupCredentials(AzureBaseCredentials.class,
null, ACL.SYSTEM, Collections.<DomainRequirement>emptyList());
```

Using AzureCredentials in the job (freestyle / pipeline)

Custom binding for AzureCredentials to support reading Azure service principal in both freestyle and pipeline using Credentials Binding plugin.

In freestyle jobs, click `Use secret text(s) or file(s)` in the `Build Environment` in the configuration page and add a `Microsoft Azure Service Principal` item, which allows you add credential bindings where the *Variable* value will be used as the name of the environment variable that your build can use to access the value of the credential. With the default variable names you can reference the service principal as the following:

```
echo "My client id is $AZURE_CLIENT_ID"
echo "My client secret is $AZURE_CLIENT_SECRET"
echo "My tenant id is $AZURE_TENANT_ID"
echo "My subscription id is $AZURE_SUBSCRIPTION_ID"
```

In pipelines, there're two ways to construct this binding:

1. With defaults, which will read specified service principal into four predefined environment variables: `AZURE_SUBSCRIPTION_ID`, `AZURE_CLIENT_ID`, `AZURE_CLIENT_SECRET`, `AZURE_TENANT_ID`. Sample pipeline code:

```
withCredentials([azureServicePrincipal('credentials_id')]) {
    sh 'az login --service-principal -u $AZURE_CLIENT_ID -p
$AZURE_CLIENT_SECRET -t $AZURE_TENANT_ID'
}
```

2. With custom name, where you can control the names of the variables. Sample pipeline code:

```
withCredentials([azureServicePrincipal(credentialsId:
'credentials_id',
                                       subscriptionIdVariable:
'SUBS_ID',
                                       clientIdVariable: 'CLIENT_ID',
                                       clientSecretVariable:
'CLIENT_SECRET',
                                       tenantIdVariable:
'TENANT_ID')]) {
    sh 'az login --service-principal -u $CLIENT_ID -p
$CLIENT_SECRET -t $TENANT_ID'
}
```

Changelog

Version 1.6.0, 2018-03-29

- Use scoped credentials lookup

Version 1.5.0, 2018-02-09

- Support for certificate based service principal
- Fix the configuration verification bug on non-global clouds

Version 1.4.0, 2017-12-21

- Support Environment selection for MSI credentials

Version 1.3.1, 2017-11-27

- Disable plugin first class loader to fix remote class loading issue before Jenkins 2.66.
  PluginFirstClassLoader#findResource returns null which causes ClassNotFoundException on remote class loading from slave to master.

Version 1.3, 2017-11-03

- Support for Azure KeyVault credentials
- Support for MSI credentials
- Upgrade Azure SDK dependency to version 1.3.0
- Add Third Party Notices

Version 1.2, 2017-06-19

- Improve credential creation experience:
  - Change subscription ID, client ID and OAuth Endpoint from secret to plain text.
  - Give OAuth Endpoint a default value to make it easier to compose from tenant ID.

- Add a custom binding for AzureCredentials so that you can use Azure service principal in Credentials Binding plugin. Now in a Jenkins pipeline you can retrieve Azure service principal and use it in Azure CLI using the following code:

```
withCredentials([azureServicePrincipal('my service principal')]) {
  sh 'az login --service-principal -u $AZURE_CLIENT_ID -p
$AZURE_CLIENT_SECRET -t $AZURE_TENANT_ID'
}
```

  You can also use Azure service principal in freestyle project by checking "Use secret text(s) or file(s)".

Version 1.1, 2017-06-02

- Bug fix: fix an issue that tenant is corrupted after update an existing credential.

Version 1.0, 2017-03-08

- Initial release