Azure Container Agents Plugin

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

Azure Container Agents Plugin can help you to run a container as an agent in Jenkins

We have two different orchestrators:

- Azure Container Service (Kubernetes)
- Azure Container Instance

> **Tutorial: run your build in Azure using Azure Container Instances (ACI)**

How to install

You can install/update this plugin in Jenkins update center (Manage Jenkins -> Manage Plugins, search Azure Container Agents Plugin).

You can also manually install the plugin if you want to try the latest feature before it's officially released. To manually install the plugin:

1. Clone the repo and build:

   ```
   mvn package
   ```

2. Open your Jenkins dashboard, go to Manage Jenkins -> Manage Plugins
3. Go to Advanced tab, under Upload Plugin section, click Choose File.
4. Select `azure-container-agents.hpi` in target folder of your repo, click Upload.
5. Restart your Jenkins instance after install is completed.

Pre-requirements

- Service Principal: Create Service Principal via Azure CLI 2.0

Azure Container Service (Kubernetes)

With Azure Container Service (Kubernetes), you can create a container in you Kubernetes Cluster as agent.

You need to create your own Kubernetes Cluster in Azure and maintain the number of nodes.

You can also create containers in ACI using aci-connector-k8s (experimental).

Pre-requirements

If using Azure Container Service (Kubernetes), you need to create a Kubernetes Cluster on Azure.

Configure the plugin

1. Jenkins -> Manage Jenkins -> Configure System
2. Press `Add a new cloud` and choose `Azure Container Service(Kubernetes)`
3. Specify `Cloud Name` and it should be unique.
4. Choose an existing `Azure Service Principal` or create a new credential.
5. Choose `Resource Group` and `Container Service Name`.
6. Specify `Namespace`
7. Choose an existing `ACS Credential` or create a new one. You can choose one of two different kinds of credentials:
   - SSH Username with private key
   - Microsoft Azure Container Service
8. Press `Test Connection` to make sure the configurations above are correct.

Configure the Pod Template

Although Kubernetes supports multi-containers in a Pod, but we only support one container per pod now.

Please ensure JenkinsURL, secret and nodeName passed to container via arguments or environment variables.

1. Specify `Name` and `Labels`
2. Choose a `Docker image`. Please note that the slave will connect with master via JNLP, so make sure JNLP installed in image. Default image is `jenkins/jnlp-slave` and you can also use it as base image.
3. If you use a private registry, you need to specify a credential and you have two choices:
    - Use a Private Registry Secret. You need to create a Secret in your Kubernetes cluster in advance and then fill in the Secret name.
    - Use a Private Registry Credential. You just need to fill in the credential and we will create a Secret for you.
4. Specify a `Command` to override the ENTRYPOINT or leave it blank.
5. Specify the `Arguments`. `${rootUrl}`, `${secret}` and `${nodeName}` will be replace with JenkinsUrl, Secret and ComputerNodeName automatically.
6. Specify the `Working Dir`. It's the root dir of you job. You must ensure login user have the write permission to this directory.
7. Add Environment Variables and Volumes. Please find details in help and you may need some manual operation to use specific Volumes.
8. Choose a retention strategy. You can get details in help.
9. Specify node where the container create on. If using Azure Disk or using aci-connector-k8s, you need to specify a node.
10. Check whether to run container in privileged mode.
11. Specify Request / Limit of the resource Cpu / Memory. Find details in Managing Compute Resources for Containers

Use Aci-Connector-k8s(experimental)

Please note this software is experimental and should not be used for anything resembling a production workload.

1. Clone and Install aci-connector-k8s.
2. Specify `aci-connector` in `Specify Node`. Then all the container of this Pod Template will be created in ACI.

Only very few configurations are supported now.

Configure Azure Container Service (Kubernetes) via Groovy Script

If you want to configure Azure Container Service (Kubernetes) via script rather than manually configure it in UI. You can use the sample below in Manage Jenkins -> Script Console. The sample only contains a few of arguments. Find all the arguments in folder builders.

```
import com.microsoft.jenkins.containeragents.builders.*

def myCloud = new KubernetesCloudBuilder()
    .withCloudName("mycloud")
    .withAzureCredentialsId("<Azure Credentials Id>")
    .withResourceGroup("myResourceGroup")
    .withServiceName("myServiceName")
    .withAcsCredentialsId("<ACS Credentials Id>")
    .addNewTemplate()
        .withName("mytemplate")
        .withLabel("k8s")
    .endTemplate()
    .build();

Jenkins.getInstance().clouds.add(myCloud);

//inherit template from existing template
import com.microsoft.jenkins.containeragents.builders.*

def baseTemplate = new PodTemplateBuilder()
    .withImage("privateImage")
    .addNewImagePullSecret("yourSecret")
    .addNewEnvVar("key", "value")
    .build();

def myCloud = new KubernetesCloudBuilder()
    .withCloudName("mycloud")
    .withAzureCredentialsId("<Azure Credentials Id>")
    .withResourceGroup("myResourceGroup")
    .withServiceName("myServiceName")
    .withAcsCredentialsId("<ACS Credentials Id>")
    .addNewTemplateLike(baseTemplate)
        .withName("mytemplate")
        .withLabel("k8s")
    .endTemplate()
    .build();

Jenkins.getInstance().clouds.add(myCloud);
```

Azure Container Instance

Azure Container Instances offers the fastest and simplest way to run a container in Azure, without having to provision any virtual machines and without having to adopt a higher-level service.

Pre-requirements

- Resource Group in West US or East US (ACI only support these two regions now)

Configure the plugin

1. Jenkins -> Manage Jenkins -> Configure System
2. Press `Add a new cloud` and choose `Azure Container Instance`
3. Specify `Cloud Name` and it should be unique.
4. Choose an existing `Azure Service Principal` or create a new credential.
5. Choose `Resource Group`.

Configure the Container Template

1. Specify `Name` and `Labels`
2. Set `Startup Timeout`.
3. Select `Image OS Type`, Windows or Linux.
4. Fill in `Docker Image`. Please note that the slave will connect with master via JNLP, so make sure JNLP installed in image. Default image is `jenkins/jnlp-slave` and you can also use it as base image.
5. If you use a private registry, you need to specify a credential. Please note the `URL` should not contain protocol (e.g. index.docker.io).
6. Specify a `Command`. Now the `Command` will override the ENTRYPOINT. `Arguments`. `${rootUrl}`, `${secret}` and `${nodeName}` will be replace with JenkinsUrl, Secret and ComputerNodeName automatically.
7. Specify the `Working Dir`. You must ensure login user have the write permission to this directory.
8. Add `Ports`, `Environment Variables` and `Volumes`
9. Choose a retention strategy. You can get details in help.
10. Specify `Cpu Requirement` and `Memory Requirement`, ACI containers costs per second. Find more detail in Price Details.

Configure Azure Container Instance via Groovy Script

If you want to configure Azure Container Instance via script rather than manually configure it in UI. You can use the sample below in Manage Jenkins -> Script Console. The sample only contains a few of arguments. Find all the arguments in folder builders.

```
import com.microsoft.jenkins.containeragents.builders.*

def myCloud = new AciCloudBuilder()
    .withCloudName("mycloud")
    .withAzureCredentialsId("<Your Credentials Id>")
    .withResourceGroup("myResourceGroup")
    .addNewTemplate()
        .withName("mytemplate")
        .withLabel("aci")
        .addNewPort("80")
        .addNewEnvVar("key","value")
    .endTemplate()
    .build();

Jenkins.getInstance().clouds.add(myCloud);

//inherit template from existing template
import com.microsoft.jenkins.containeragents.builders.*

def baseTemplate = new AciContainerTemplateBuilder()
    .withImage("privateImage")
    .addNewPort("80")
    .addNewEnvVar("key", "value")
    .build();

def myCloud = new AciCloudBuilder()
    .withCloudName("mycloud")
    .withAzureCredentialsId("<Your Credentials Id>")
    .withResourceGroup("myResourceGroup")
    .addNewTemplateLike(baseTemplate)
        .withName("mytemplate")
        .withLabel("aci")
    .endTemplate()
    .build();

Jenkins.getInstance().clouds.add(myCloud);
```

Data/Telemetry

Azure Container Agents Plugin collects usage data and sends it to Microsoft to help improve our products and services. Read our privacy statement to learn more.

You can turn off usage data collection in Manage Jenkins -> Configure System -> Azure -> Help make Azure Jenkins plugins better by sending anonymous usage statistics to Azure Application Insights.

Changelog

Version 0.4.1, 2018-01-10

- Fix AKS agents after AKS resource API change

Version 0.4.0, 2018-01-02

- **Breaking Change**: No longer mount Empty Volume to Working Dir automatically. Make sure that Jenkins have permission to R/W in Working Dir or mount Empty Volume by yourself
- Add support for SSH
- UI change: Hide AcsCredential when choosing AKS
- Add more logs in provision ACI for inspecting errors conveniently

Version 0.3.0, 2017-11-29

- Add support for MSI
- Fix bugs in retention strategy

Version 0.2.0, 2017-11-3

- Support Azure Kubernetes Service
- Add Third Party Notice
- Various bugs fix

Version 0.1.2, 2017-10-18

- Remove runtime licenses

Version 0.1.1, 2017-09-29

- Fixed a guava dependency issue

Version 0.1.0, 2017-09-27

- Initial release