Azure Container Service Plugin

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

A Jenkins Plugin to deploy configurations to Azure Container Service (AKS).

It also supports deployment to the legacy Azure Container Service (ACS) with the following orchestrator:

- Kubernetes
- DC/OS with Marathon
- Docker Swarm

It also supports deployments on Azure Kubernetes Service.

It provides the following main functionality:

- Integration with ACS. Allow you to select existing ACS clusters and manages the authentication credentials.
- Unified configure interface for all the supported orchestrator.
- Variable substitution for configurations, which enables dynamic deployment in CI/CD.
- Docker login credentials management for pulling images from private repository.

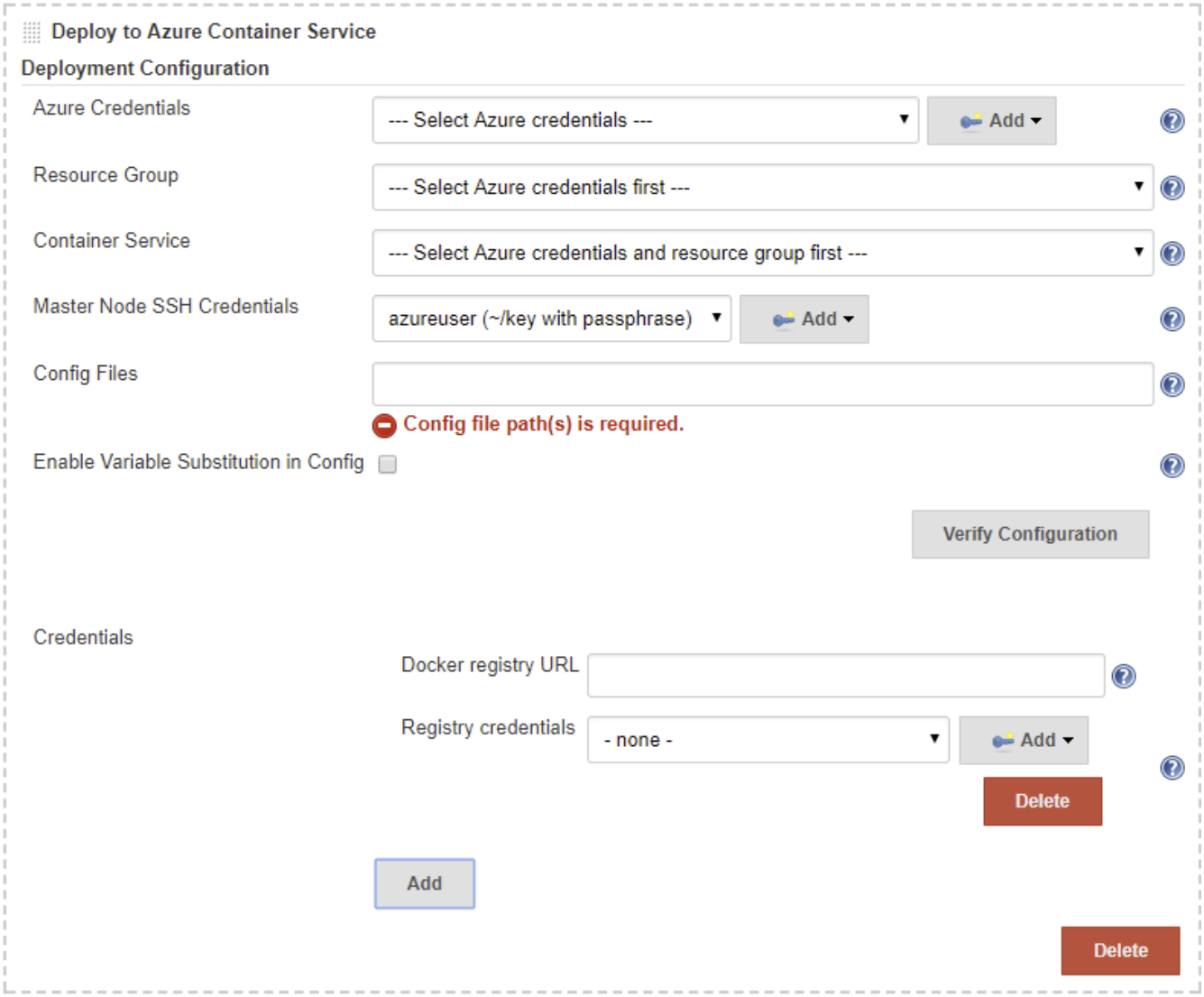
Pre-requirements

- AKS cluster, or an ACS cluster with the supported orchestrator
    - Azure Container Service with Kubernetes
    - Azure Container Service with DC/OS and Swarm

  A Azure service principal that can be used to manage the ACS cluster
    - Application and service principal objects in Azure Active Directory (Azure AD)
    - Use portal to create an Azure Active Directory application and service principal that can access resources
    - Azure Credentials Plugin
- Configurations for the target ACS cluster orchestrator, this can be
    - Kubernetes resource configurations of the following kinds:
        - Deployment
        - Replica Set
        - Replication Controller - No rolling-update support. If that's required, consider using Deployment.
        - Daemon Set
        - Pod
        - Job
        - Service
        - Ingress
        - Secret - The plugin also provides secrets configuration.
    - Marathon application configurations
    - Docker compose configurations

  In the context of container deployment, normally we should build a Docker image from the project artifacts, and push the image to a repository. This can be done with some existing plugins such as CloudBees Docker Build and Publish plugin.

Configure the plugin



1. Within the Jenkins dashboard, Select a Job then select Configure
2. Scroll to the "Add post-build action" drop down.
3. Select "Deploy to Azure Container Service"
4. Select the service principal from "Azure Credentials" dropdown. If no credentials are configured, create one.
5. All resource group names will be loaded into the "Resource Group" dropdown. Select the one containing your ACS cluster.
6. All the container service available in the selected resource group will be loaded into the "Container Service" dropdown. Select your target ACS cluster. (It's suggested that we use a standalone resource group to manage an ACS cluster, and do not add other resources or ACS clusters into the resource group.)
7. Select the "Master Node SSH Credentials". This should be the credentials of type "SSH Username with private key", where username is the login name you specified when you create the ACS cluster (`azureuser` by default), and private key is the one matching the public key you specified on the ACS cluster creation (by default that may be `$HOME/.ssh/id_rsa` on Linux and `%USERPROFILE%\.ssh\id_rsa` on Windows).
8. Enter the "Config Files" path of the configurations you want to deploy, in the form of Ant glob syntax. Use comma (,) to separate multiple patterns.
9. If you want to dynamically update the configurations on each build, for example, use the `$BUILD_NUMBER` as the tag name of the image being pulled, you can tick the "Enable Variable Substitution in Config" option and the variables (in the pattern `$VARIABLE` or `${VARIABLE}`) in the configurations will be substituted with the corresponding value if they exists in the environment variables.
10. If the configurations needs to pull images from private repository, click the "Docker Container Registry Credentials..." button and add them one by one.
    - For Kubernetes, you can enter the secret name and the namespace where the secret will be created based on the credentials configured. The credentials you provided will be consolidated into a Secret resource in your Kubernetes cluster with the name you provided. You can use that secret in your Kubernetes configuration.

      You can use variables in the secret name (e.g., `$BUILD_NUMBER`), to generate a secret specific to a build. The name will be exposed as environment variable `KUBERNETES_SECRET_NAME` and you can use that in your Kubernetes resource configurations if the "Enable Variable Substitution in Config" option is turned on.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: sample-k8s-deployment
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: sample-k8s-app
    spec:
      containers:
      - name: sample-k8s-app-container
        image: <username or registry URL>/<image_name>:<tag(maybe, $BUILD_NUMBER)>
        ports:
        - containerPort: 8080
      imagePullSecrets:
      - name: $KUBERNETES_SECRET_NAME
```

- For Marathon on DC/OS, you can specify the "DC/OS Docker Credentials Path", the credentials you provided will be archived into a file `docker.tar.gz`, and uploaded to the agent nodes in the path you filled. You can use it to construct the URI in your Marathon config:

  `file://<absolute_path_you_filled>/docker.tar.gz`

  If this path is left blank, a unique path will be generated under `$HOME/acs-plugin-dcos.docker`.

  The URI will be exported to the environment variable `MARATHON_DOCKER_CFG_ARCHIVE_URI` which you can use in your Marathon configuration.

```
{
  "id": "marathon-demo-app",
  "cmd": null,
  "cpus": 1,
  "mem": 512,
  "disk": 0,
  "instances": 1,
  "container": {
    "docker": {
      "image": "<username or registry URL>/<image_name>:<tag(maybe, $BUILD_NUMBER)>",
      "network": "BRIDGE",
      "portMappings": [
        {
          "containerPort": 8080,
          "hostPort": 80,
          "protocol": "tcp",
          "name": "80",
          "labels": null
        }
      ]
    },
    "type": "DOCKER"
  },
  "acceptedResourceRoles": [
    "slave_public"
  ],
  "uris": [
    "$MARATHON_DOCKER_CFG_ARCHIVE_URI"
  ]
}
```

- Add a credentials entry for each of the private Docker registries involved in your configuration. If it is hosted on DockerHub, you can leave the URL as empty; otherwise for other private registries, you need to specify the "Docker registry URL".

11. You may verify the static configuration by clicking "Verify Configuration". This will give you basic result of the configuration quality. You need to run a sample build to verify it works as some of the contents has to be loaded at build time.

Pipeline Support

To use the plugin in pipeline, go to the Pipeline Syntax page when you configure the pipeline job, and choose acsDeploy: Deploy to Azure Container Service from the Sample Step dropdown. You can configure it and click Generate Pipeline Scriptwhich will give you

```
acsDeploy(azureCredentialsId: '<azure-credential-id>',
          resourceGroupName: '<resource-group-name>',
          containerService: '<acs-name> | <acs-type>',
          sshCredentialsId: '<ssh-credentials-id>',
          configFilePaths: '<configuration-file-paths>',
          enableConfigSubstitution: true,

          // Kubernetes
          secretName: '<secret-name>',
          secretNamespace: '<secret-namespace>',

          // Docker Swarm
          swarmRemoveContainersFirst: true,

          // DC/OS Marathon
          dcosDockerCredentialsPath: '<dcos-credentials-path>',

          containerRegistryCredentials: [
              [credentialsId: '<credentials-id>', url: '<docker-registry-url>']
          ])
```

Data/Telemetry

Azure Container Service 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 0.2.3, 2018-04-03

- Support for credentials lookup in Folders

Version 0.2.2, 2018-02-09

- Abort the build flow if the deployment failed

Version 0.2.1, 2018-01-09

- Fix AKS deployment after AKS resource API change (#10)

Version 0.2.0, 2018-01-05

- Support MSI

Version 0.1.5, 2017-11-27

- Fix typo in AI reporting

Version 0.1.4, 2017-11-07

- Support for Azure Kubernetes Service (AKS)
- Add Third Party Notice

Version 0.1.3, 2017-10-18

- Remove EULA
- Remove "Run On" check

Version 0.1.2, 2017-09-29

- Fixed a stream closed issue when variable substitution is disabled

Version 0.1.1, 2017-09-28

- Fixed an issue that plugin crashes on fastxml load

Version 0.1.0, 2017-09-27

- Initial release