Create a new self-hosted build agent pool in Azure DevOps

Introduction

This is a description of a process to create a build agent pool in Azure DevOps. We are setting up a self-hosted agent in Azure Pipelines to run inside an Ubuntu container with Docker. This is useful because we can control the runtime, its dependencies and the cost when hosting our agent pool in AWS.

Note: The existing agents and the agents in this article are meant to be deployed in the **dsm-build** account - `760288302223`.

When running an agent in Docker, we need to pass a few environment variables to `docker run`, which configures the agent to connect to Azure Pipelines. We also can and need to customize the container to suit our needs.

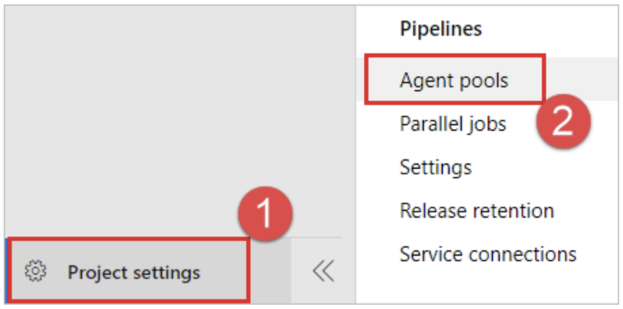
Concepts

An instance of an agent pool consists of:

- ECS cluster in **dsm-build** account - Deployed as a CloudFormation stack is hosts current and possible task definitions of a build agent
- ECS service in **dsm-build** account - Deployed as a service inside ECS cluster via CloudFormation, it hosts and maintains docker containers with the customized agent image.
- ECR repository in **dsm-build** account - This container repository hosts the customized image of an agent pool.
- Customized container image created based off the following instructions from Azure: Run a self-hosted agent in Docker - Azure Pipelines.
- Agent build pool in **Azure DevOps:** https://dsmplatform.visualstudio.com/aws-mendix-deployment/_settings/agentqueues
- Personal Access Token created for the docker container to authenticate with the Azure DevOps as per following instructions: Use personal access tokens - Azure DevOps

Create Agent Pool step by step

1. Create an Agent Pool in the team project



2. Generate Personal Access Token to be used by the Agent Pool: Use personal access tokens - Azure DevOps
3. Create an ECR repository to host the custom image in **dsm-build** account.
4. Create custom docker image: Run a self-hosted agent in Docker - Azure Pipelines.
5. Upload the image to the newly-created ECR repository.
6. Ensure that an ECS cluster in dsm-build account has enough capacity to host new agents. This is controlled via CloudFormation stack it's deployed as a part of - **dsm-ecs-cluster**. Add capacity if necessary.
7. Create a new service associated with the ECR repository/ECS cluster mentioned above. An example of a CloudFormaton template that contains all necessary resources is given below. Please make sure to update the names of the resource and double-check the template before using it.

```yaml
Description: DSM AWS ECS - VSTS Build Agent - Ubuntu 20
Parameters:
  AzpPool:
    Type: String
    Default: DSM - AWS Build Account - Ubuntu 20
  AzpUrl:
    Type: String
    Default: https://dsmplatform.visualstudio.com
  DockerImageTag:
    Type: String
    Default: 'latest'

Resources:
  VstsAgentCloudWatchLogsGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: /dsm/ecs/task/vsts-build-agent-ubuntu-20
      RetentionInDays: 7
  VstsAgentTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: dsm-vsts-build-agent-ubuntu-20
      TaskRoleArn: !Ref VstsAgentServiceIamRole
      ExecutionRoleArn: !Ref VstsAgentServiceIamRole
      Volumes:
        - Name: docker_socket
```

```yaml
          Host:
            SourcePath: /var/run/docker.sock
      ContainerDefinitions:
        - Name: dsm-vsts-build-agent-ubuntu-20
          Image: !Sub '${AWS::AccountId}.dkr.ecr.${AWS::Region}.amazonaws.com/dsm/vsts-build-agent-ubuntu-20:${DockerImageTag}'
          Memory: 2048
          MemoryReservation: 2048
          MountPoints:
            - SourceVolume: docker_socket
              ContainerPath: /var/run/docker.sock
          Environment:
            - Name: AZP_POOL
              Value: !Ref AzpPool
            - Name: AZP_URL
              Value: !Ref AzpUrl
          Secrets:
            - Name: AZP_TOKEN
              ValueFrom: /AzureDevopsTokenUbuntu20
          LogConfiguration:
            LogDriver: awslogs
            Options:
                awslogs-group: /dsm/ecs/task/vsts-build-agent-ubuntu-20
                awslogs-region: !Ref AWS::Region
  VstsAgentServiceIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Sub 'dsm-ecs-vsts-build-agent-ubuntu-20-${AWS::Region}'
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ecs-tasks.amazonaws.com
            Action:
              - sts:AssumeRole

  VstsAgentIamGoup:
    Type: AWS::IAM::Group
    Properties:
      GroupName: dsm-vsts-build-agent-ubuntu-20

  VstsAgentTaskIamPolicy:
    Type: AWS::IAM::Policy
    Properties:
      Roles:
        - !Ref VstsAgentServiceIamRole
      PolicyName: AllowAccess
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:BatchGetImage
              - ecr:CompleteLayerUpload
              - ecr:CreateRepository
              - ecr:Describe*
              - ecr:GetAuthorizationToken
              - ecr:GetDownloadUrlForLayer
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:PutLifecyclePolicy
              - ecr:PutRegistryPolicy
              - ecr:UploadLayerPart
              - ecs:CreateCluster
              - ecs:DeregisterContainerInstance
              - ecs:DiscoverPollEndpoint
              - ecs:Poll
              - ecs:RegisterContainerInstance
              - ecs:StartTelemetrySession
              - ecs:Submit*
              - logs:CreateLogStream
              - logs:PutLogEvents
            Resource: '*'
          - Effect: Allow
            Action:
              - ssm:GetParameters
              - ssm:GetParameter
            Resource: !Sub 'arn:aws:ssm:${AWS::Region}:${AWS::AccountId}:parameter/AzureDevopsTokenUbuntu20'
          - Effect: Allow
            Action:
              - kms:Decrypt
            Resource: !Sub 'arn:aws:kms:${AWS::Region}:${AWS::AccountId}:alias/aws/ssm'
          - Effect: Allow
            Action: 'sts:AssumeRole'
            Resource:
              - 'arn:aws:iam::*:role/dsm-deploy'
              - 'arn:aws:iam::*:role/dsm-*-deploy' # application-specific deployments

  VstsAgentGroupIamPolicy:
    Type: AWS::IAM::Policy
    Properties:
      Groups:
        - !Ref VstsAgentIamGoup
      PolicyName: AllowAccess
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action: 'sts:AssumeRole'
            Resource:
              - 'arn:aws:iam::767766554505:role/dsm-deploy' # dsm-billing
              - 'arn:aws:iam::991732646622:role/dsm-deploy' # dsm-iam
              - 'arn:aws:iam::759457050644:role/dsm-deploy' # dsm-audit
              - 'arn:aws:iam::760288302223:role/dsm-deploy' # dsm-build
              - 'arn:aws:iam::023254921910:role/dsm-deploy' # dsm-backup
              - 'arn:aws:iam::209242042733:role/dsm-deploy' # dsm-logging
              - 'arn:aws:iam::330650092564:role/dsm-deploy' # dsm-dx
              - 'arn:aws:iam::524908392606:role/dsm-deploy' # dss-dt
              - 'arn:aws:iam::613470590117:role/dsm-deploy' # dss-acc
              - 'arn:aws:iam::179615992533:role/dsm-deploy' # dss-prd
              - 'arn:aws:iam::574289697569:role/dsm-deploy' # dep-dt
              - 'arn:aws:iam::659213364010:role/dsm-deploy' # dep-acc
              - 'arn:aws:iam::884155879281:role/dsm-deploy' # dep-prd
              - 'arn:aws:iam::025669578742:role/dsm-deploy' # dnp-dt
              - 'arn:aws:iam::611747708708:role/dsm-deploy' # dnp-acc
              - 'arn:aws:iam::495829485656:role/dsm-deploy' # dnp-prd
              - 'arn:aws:iam::273580979641:role/dsm-deploy' # dsm-sandbox
            Condition:
              IpAddress:
                aws:SourceIp:
                  - '34.255.80.162/32'
                  - '52.215.150.61/32'
                  - '52.49.80.95/32'
                  - !ImportValue dsm-vpc-compute-subnet-1-cidr
                  - !ImportValue dsm-vpc-compute-subnet-2-cidr
                  - !ImportValue dsm-vpc-compute-subnet-3-cidr

  VstsAgentService:
    Type: AWS::ECS::Service
    Properties:
      ServiceName: dsm-vsts-build-agent-ubuntu-20
      Cluster: !ImportValue dsm-ecs-cluster-cluster-id
      DesiredCount: 3
      TaskDefinition: !Ref VstsAgentTaskDefinition
      PlacementStrategies:
        - Field: instanceId
          Type: spread
```

You can see that here a couple of parameters that need to be passed to the container as environment variables are created:

- AZP_URL - the URL of an Azure DevOps instance. For us, it's `https://dsmplatform.visualstudio.com`
- AZP_TOKEN - Personal Access Token implemented via SSM SecureString parameter that is created before the stack is provisioned
- AZP_POOL - Name of the pool created in the Step 1 of the process.

Image Customization

We can and should customize the container image to simplify the deployment process. For instance, in one of our agent pools intended to deploy cdk we pre-install cdk package to not spend the pipeline execution time on it. Please see the example of the customized image below:

```
FROM ubuntu:20.04
RUN DEBIAN_FRONTEND=noninteractive apt-get update
RUN DEBIAN_FRONTEND=noninteractive apt-get upgrade -y

RUN DEBIAN_FRONTEND=noninteractive apt-get install -y -qq --no-install-recommends \
    apt-transport-https \
    apt-utils \
    ca-certificates \
    curl \
    gnupg \
    git \
```

```dockerfile
    iputils-ping \
    jq \
    lsb-release \
    unzip \
    software-properties-common \
    libicu66 \
    amazon-ecr-credential-helper \
    subversion \
    python3-pip

RUN pip3 install boto3 botocore --upgrade

RUN mkdir -p /etc/apt/keyrings && \
    curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key | gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg && \
    echo "deb [signed-by=/etc/apt/keyrings/nodesource.gpg] https://deb.nodesource.com/node_20.x nodistro main" | tee /etc/apt/sources.list.d/nodesource.list && \
    apt-get update && \
    apt-get install nodejs -y && \
    npm install -g npm@latest

RUN npm install -g aws-cdk@2.88.0

RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip" && \
    unzip awscliv2.zip && \
    ./aws/install

RUN curl -sL https://aka.ms/InstallAzureCLIDeb | bash

# Can be 'linux-x64', 'linux-arm64', 'linux-arm', 'rhel.6-x64'.
ENV TARGETARCH=linux-x64

WORKDIR /azp

COPY ./start.sh .
RUN chmod +x start.sh

ENTRYPOINT [ "./start.sh" ]
```