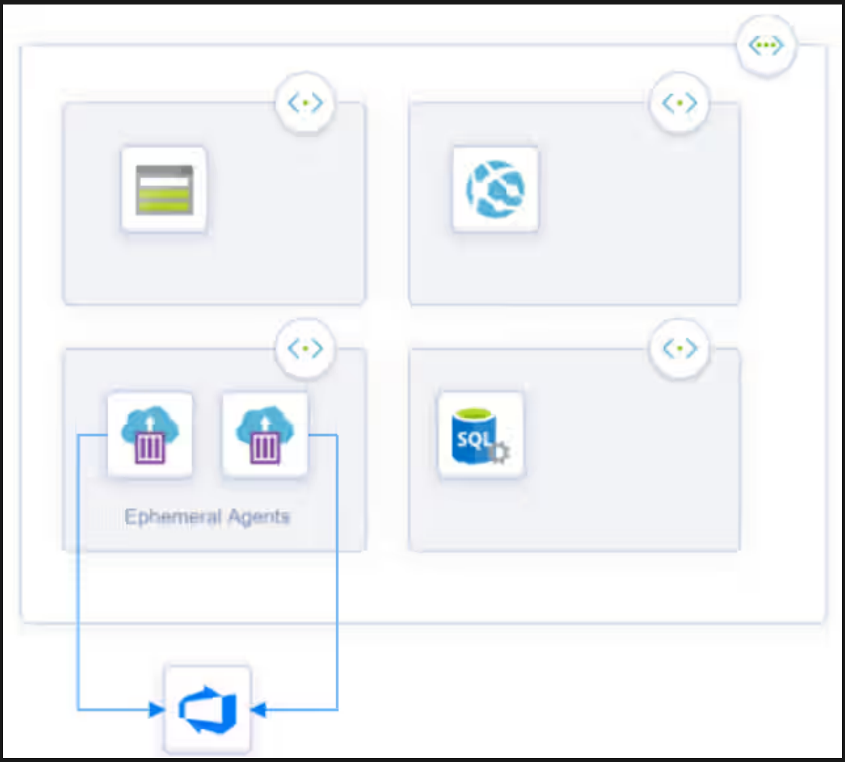
Replacing self-hosted agents with ephemeral pipelines agents in Azure DevOps

7 min read.



Schema of sample architecture using ephemeral agents

If you have Azure Resources that aren't exposed on the internet but only accessible via a private network, you can't use Microsoft-hosted agents because they can't connect to the private network. Therefore, we need to maintain a pool (or several pools) of self-hosted agents, with the associated costs and effort to maintain that pool(s).

In this case it is where **Ephemeral pipelines agents come into action**.

Overview

Ephemeral pipelines agents come to eliminate the need to use and maintain pools of self-hosted agents for deployment purposes. This type of agents are capable to deploy to private azure resources. The process is not very complex. Ephemeral pipelines agents run in an Azure Container Instance (ACI) with access to the private network where the other Azure resources are. The agents are created to run a pipeline job and then deleted to avoid extra costs and resources consumption.

This way, we can deploy to private Azure resources without having to expose them on the internet or having to maintain self-hosted agents on the same (or with access) virtual network (with their associated costs and cons).

The agent that runs in the ACI its only a basic Docker image with the necessary dependencies to run the agent and to be capable to deploy to our private resources. For example, a base Ubuntu image with the necessary dependencies and the deploy agent installed.

TL;DR: The purpose of this task is to create a short-lived Azure Pipelines Agent to run a deploy in a private virtual network so we can deploy to Azure Resources that are not internet accessible.

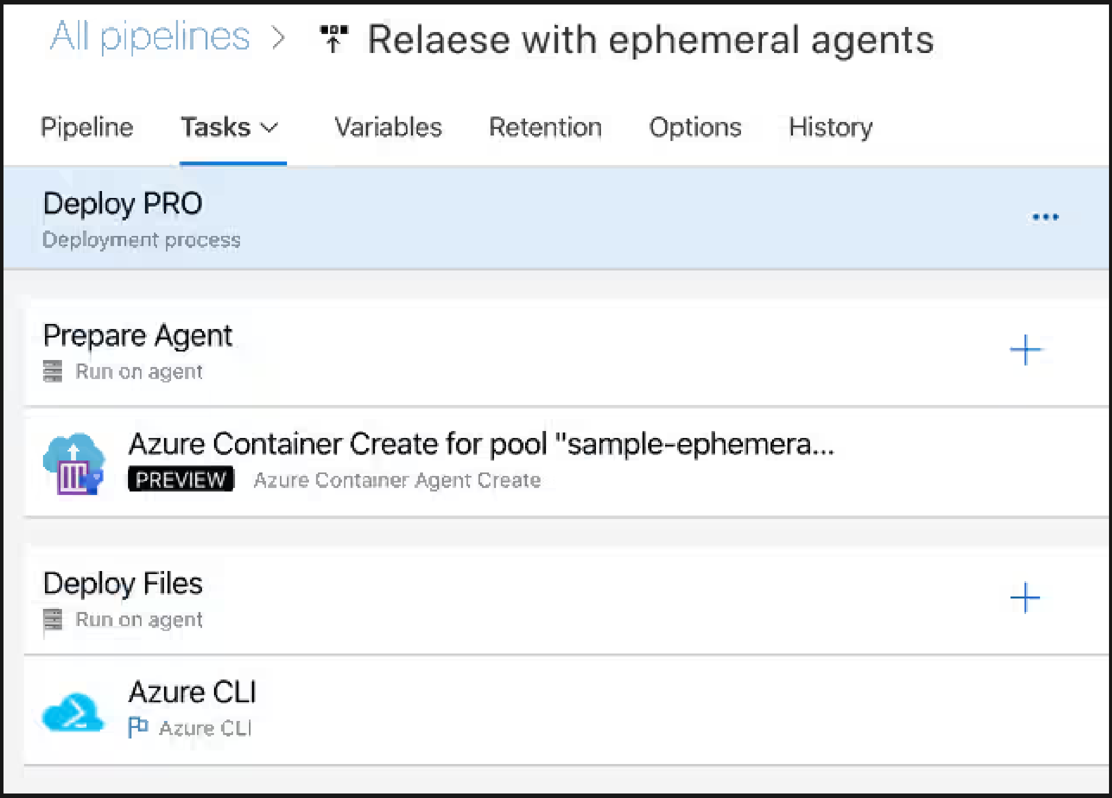
⚠ **Important:** This approach (and task) is currently in preview and has known issues and limitations. Please take this into account before proceeding to use this process.

How does the process works

Only three steps/requirements:

1. One docker image that can run a deploy agent in a container (needed to provision agents).
2. Provision one ephemeral pipeline agent to run the deployment job. This process can be done using this task. This task provisions, configures and registers an agent on an ACI using the docker image mentioned in the first step.
3. The container runs one pipeline job, and then it unregisters the agent and deletes the container (it self destructs).

So considering we have setup a docker image that can run a deploy agent in a container, our pipeline will look like this:


All pipelines > Relaese with ephemeral agents

Pipeline | Tasks | Variables | Retention | Options | History

Deploy PRO
Deployment process

Prepare Agent
Run on agent

Azure Container Create for pool "sample-ephemera...
PREVIEW Azure Container Agent Create

Deploy Files
Run on agent

Azure CLI
Azure CLI

Example pipeline with ephemeral agents

As you can see, two jobs are needed:

- The first one provisions the agent in one pool to run the deployment job.
- The second one (depending on the first being finished correctly), runs the deployment job. It runs on the agent that the first job has provisioned.

Tutorial

In this post I will guide you through a simple tutorial on how to deploy assets on a container in an Azure Storage account in a private virtual network, using Ephemeral pipeline agents.

So let's get started!

Requirements:

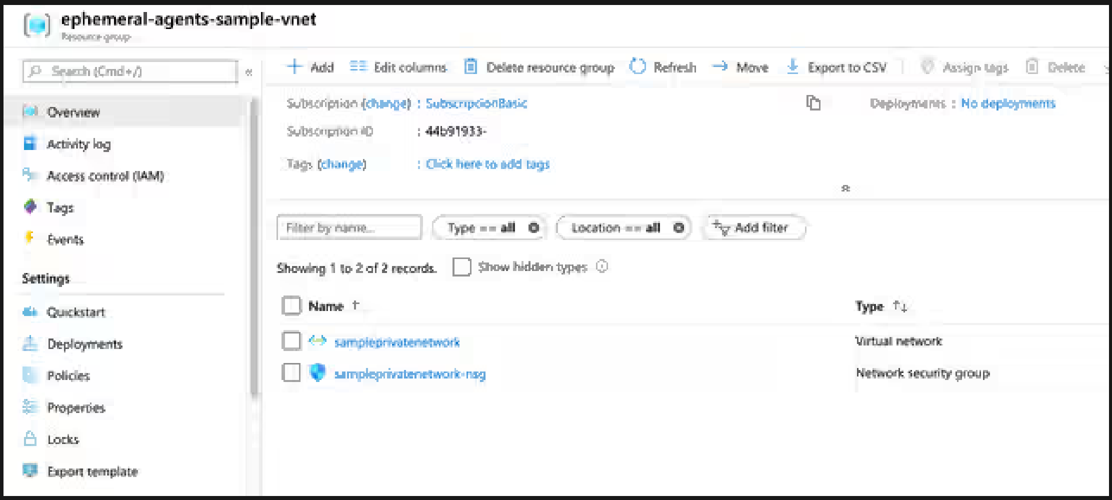
- A virtual network with a security group.

- A dedicated subnet in the virtual network to run the ephemeral agents.
- The agent must run **in the same Azure location** as the virtual network.
- All the created subnets must share the same security group.

Overview of the resources

For this tutorial I have created two resource groups:

- One for the virtual network and the security group.
- Another one for the Azure Container Registry (ACR) and the storage we will deploy to.

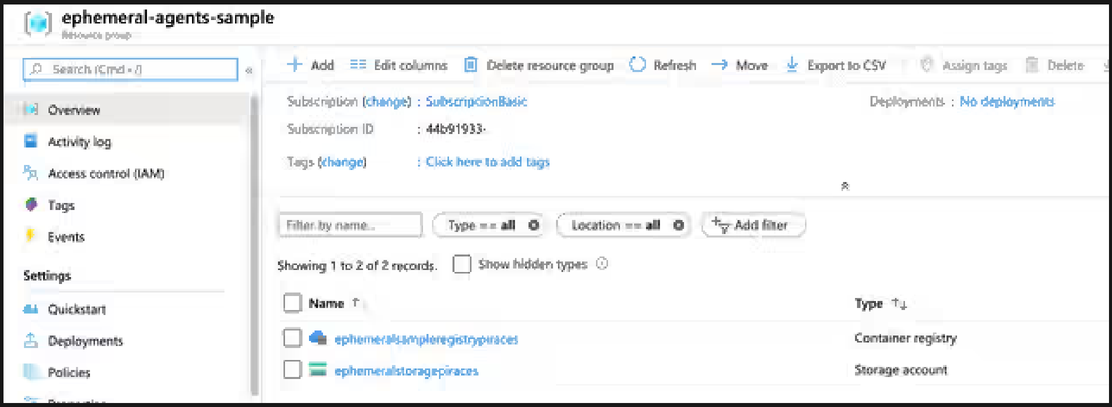

ephemeral-agents-sample-vnet
Resource group

Subscription (change) : SubscriptionBasic
Subscription ID : 44b91933-
Tags (change) : Click here to add tags
Deployments : No deployments

Showing 1 to 2 of 2 records.

Name	Type
sampleprivatenetwork	Virtual network
sampleprivatenetwork-nsg	Network security group

Resource group for the VNet and security group


ephemeral-agents-sample
Resource group

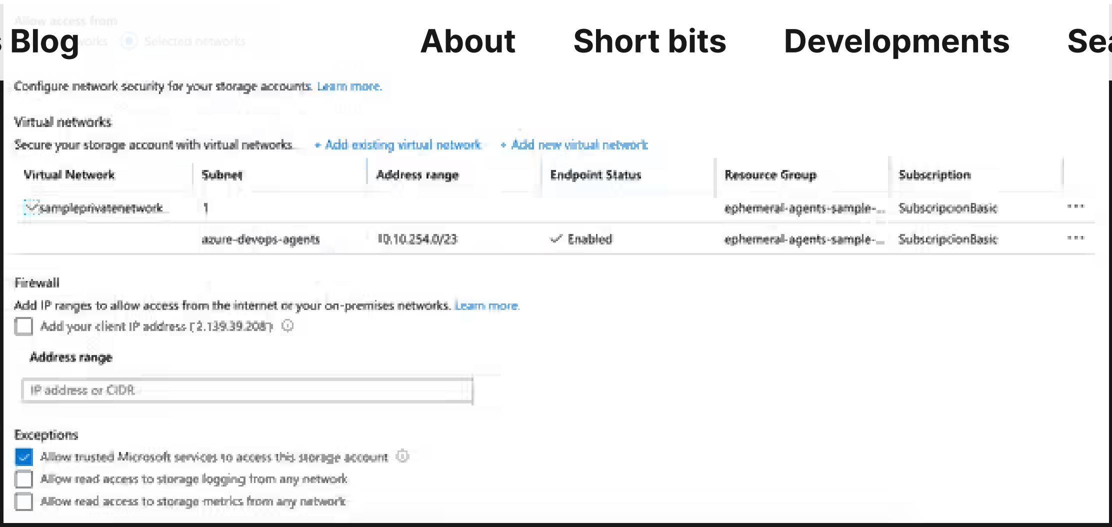
Subscription (change) : SubscriptionBasic
Subscription ID : 44b91933-
Tags (change) : Click here to add tags
Deployments : No deployments

Showing 1 to 2 of 2 records.

Name	Type
ephemeralsampleregistrypiraces	Container registry
ephemeralstoragepiraces	Storage account

Resource group for the ACR and Storage account

Also, the Azure Storage account should be in the private network, with a configuration similar to this one:



Configure network security for your storage accounts. Learn more.

Virtual networks

Secure your storage account with virtual networks. + Add existing virtual network + Add new virtual network

Virtual Network	Subnet	Address range	Endpoint Status	Resource Group	Subscription
sampleprivatenetwork	1			ephemeral-agents-sample-...	SubscripcionBasic
	azure-devops-agents	10.10.254.0/23	✓ Enabled	ephemeral-agents-sample-...	SubscripcionBasic

Firewall

Add IP ranges to allow access from the internet or your on-premises networks. Learn more.

☐ Add your client IP address ('2.139.39.208')

Address range

IP address or CIDR

Exceptions

☑ Allow trusted Microsoft services to access this storage account

☐ Allow read access to storage logging from any network

☐ Allow read access to storage metrics from any network

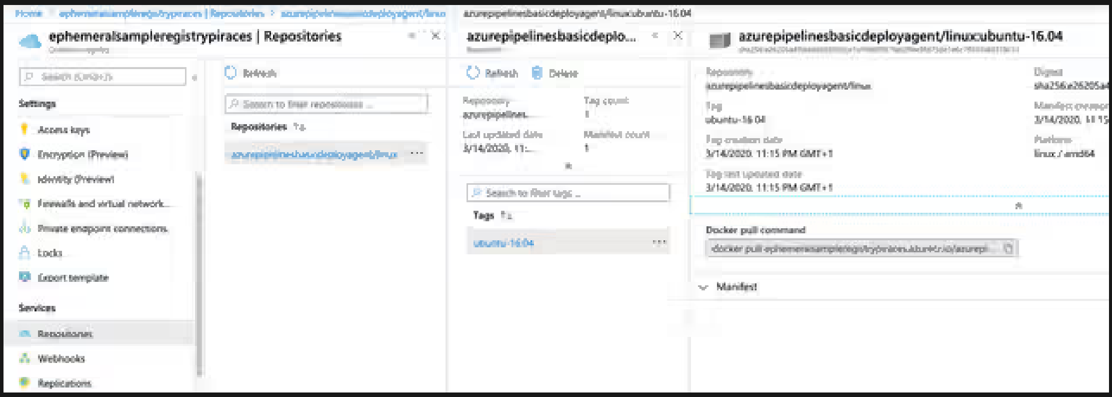
Azure Storage Account VNet configuration

If you need it, the main repo of ephemeral agents has sample scripts on how to deploy these resources.

Pushing the base image for ephemeral agents

Once finished configuring our resources in Azure, we will need the base image to use with the ephemeral agents.
In this case I have used the Agent Images available in the GitHub repo, specifically the Ubuntu one. We can use the pipeline in the GitHub repo to deploy our Agent image in the ACR (using an ACR Service connection).

After that, we should have one repository with one image in our ACR:



ephemeralsampleregistrypiraces | Repositories

Settings: Access keys, Encryption (Preview), Identity (Preview), Firewalls and virtual network..., Private endpoint connections, Locks, Export template

Services: Repositories, Webhooks, Replications

Repositories: azurepipelinesbasicdeployagent/linux...

azurepipelinesbasicdeplo...

Repository: azurepipelines... Tag count: 1

Last updated date: 3/14/2020, 11:... Manifest count: 1

Tags: ubuntu-16.04

azurepipelinesbasicdeployagent/linuxcubuntu-16.04

Repository: azurepipelinesbasicdeployagent/linux...

Tag: ubuntu-16.04

Tag creation date: 3/14/2020, 11:15 PM GMT+1

Tag last updated date: 3/14/2020, 11:15 PM GMT+1

Digest: sha256:a26205a4...

Manifest creation date: 3/14/2020, 11:15...

Platform: linux / amd64

Docker pull command: docker pull ephemeralsampleregistrypiraces.azurecr.io/azurepi...

Manifest

ACR with sample Agent Image based on Ubuntu

Setting up the pipeline in Azure DevOps

After creating the needed resources and pushing the Agent Image to the ACR, we are ready to create our pipeline and use ephemeral agents to deploy.

First of all, we will need to create an additional agent pool in our Project, and grant access to all pipelines.

Permissions:

For this process we will need to give specific permissions in order to allow the pipeline to register agents in the new pool, running jobs in these agents and unregister them.

In order to register the agent a token with sufficient permissions on the Azure DevOps Organization the agent is going to be registered is needed.

You can use two different types of tokens: a **Personal Access Token** or an **OAuth token (recommended)**.

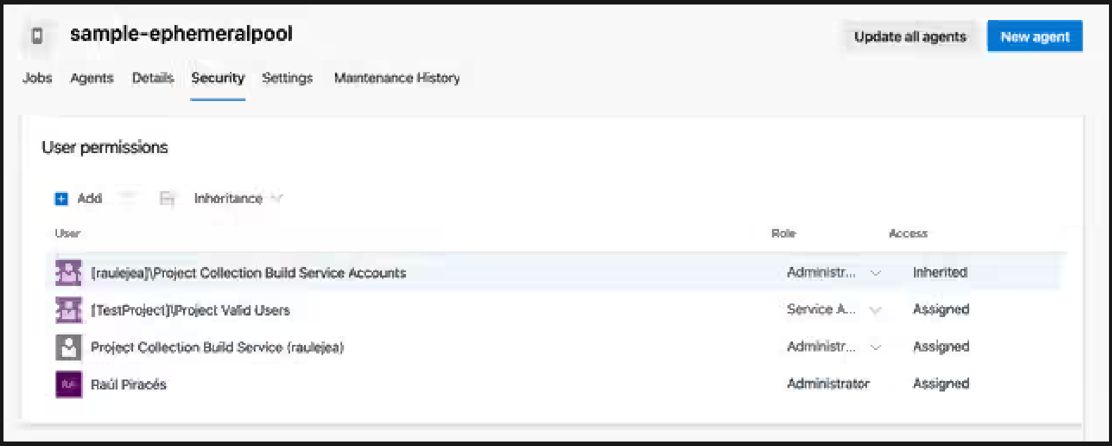
- If using an Personal Access Token (PAT), it requires the *Agent Pools Read & Manage scope* and that the user who owns the PAT has administration privileges on the agent pool you intend to register the agent.

- If using an OAuth token, the best approach is to use **System.AccessToken** which are short lived, dynamic and automatically managed by the system.

If using the OAuth token, you need to met the following conditions:

- The timeout for the job that creates the agent only expires after the deploy job in the agent is finished (because the token is used to register and unregister the agent in the pool).
- The account `Project Collection Build Service (YOUR_ORG)` needs to have administration permissions on the pool (granted at the organization level, not at team project level).

In this example, permissions on the pool (for the OAuth token approach) will look like the following:



User	Role	Access
[raulejea]\Project Collection Build Service Accounts	Administr...	Inherited
[TestProject]\Project Valid Users	Service A...	Assigned
Project Collection Build Service (raulejea)	Administr...	Assigned
Raúl Piracés	Administrator	Assigned

Pool permissions for OAuth token

Additional considerations:

You must register an Azure Resource provider for the namespace `Microsoft.ContainerInstance` in order to create container instances for the agents. This can be done easily opening a Powershell instance in the Azure portal and executing the following command:

```
Register-AzureRmResourceProvider -ProviderNamespace 'Microsoft.ContainerIns
```

This will register the needed Azure Resource provider.

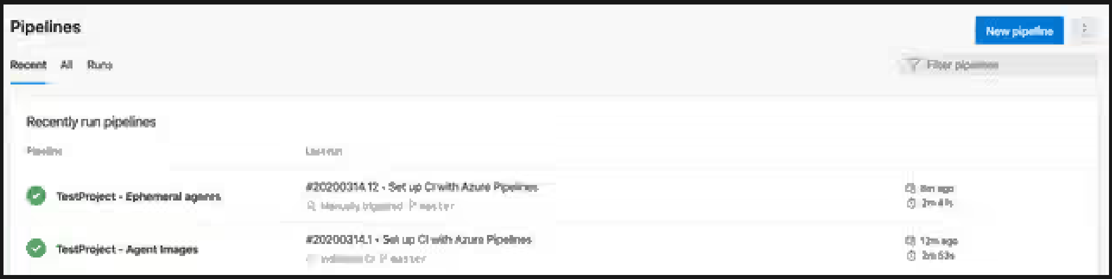
Service connections:

We will need a Service connection using a Service principal, which has to have access to the VNet resource group and the ACR and Storage Account resource group. We can set it up in the project settings in Azure DevOps, granting permissions to one resource group and then going to the Azure Portal to grant permissions to the another one. Make sure to grant permissions to all pipelines.

Also, as mentioned above we will need an ACR Service Connection to our ACR to run the ephemeral agents in the ACI from the Agent Image in the ACR (granting permissions to all pipelines).

Pipeline definition:

In this part of the process two pipelines are created for my sample project, one for creating the base image for the agents in the ACR (which we have previously created), and another one for the main deploy process of our project.



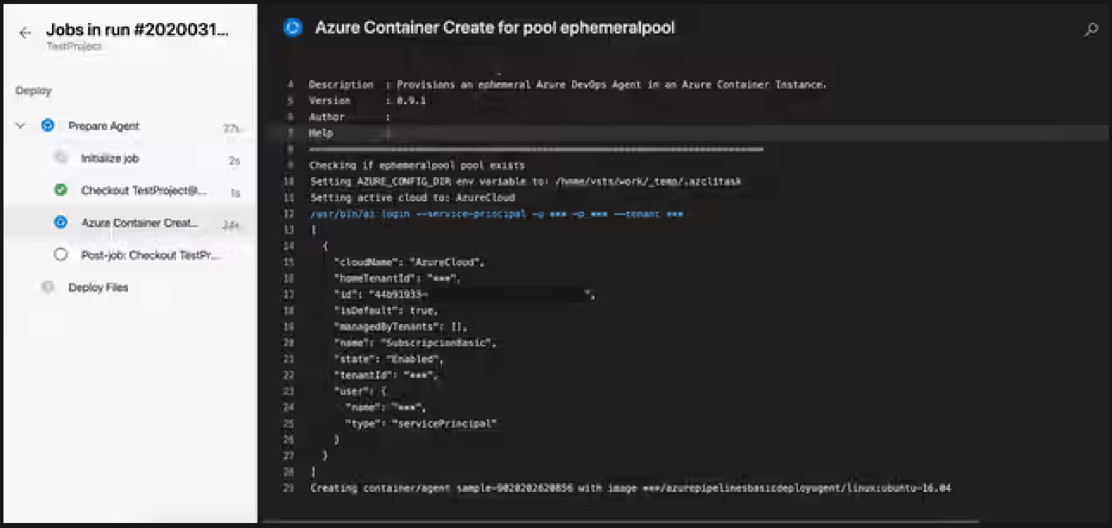
Recently run pipelines

Pipeline	Last run	
TestProject - Ephemeral agents	#20200314.12 • Set up CI with Azure Pipelines Manually triggered master	8m ago 2m 41s
TestProject - Agent Images	#20200314.1 • Set up CI with Azure Pipelines [illegible] master	12m ago 2m 53s

Pipelines defined in our projects

The main deploy pipeline used in this tutorial is the same defined in the GitHub repo sample. As we have seen in the previous stages, we have to define two jobs: one to provision the agent and another one to perform the deploy job.

Configure the variables in the sample pipeline to reference your resources correctly and then try to run it:

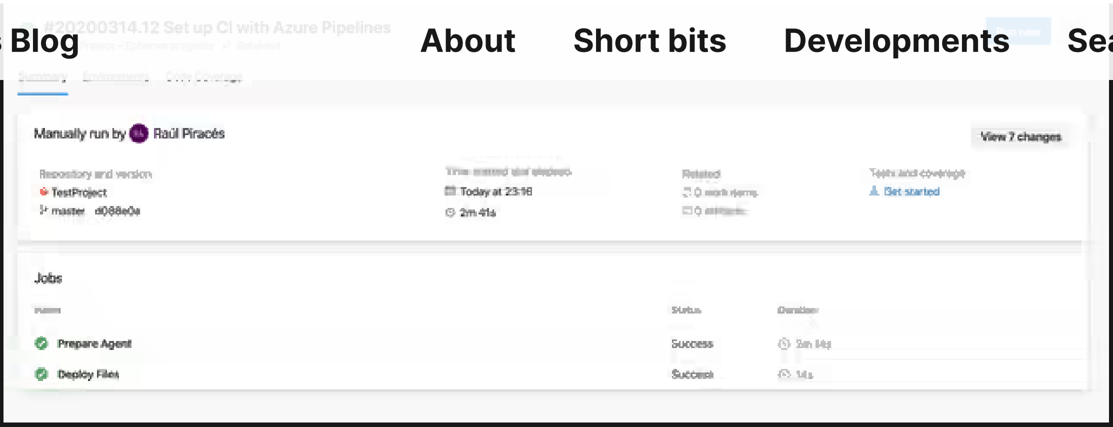


Azure Container Create for pool ephemeralpool

```
Description  : Provisions an ephemeral Azure DevOps Agent in an Azure Container Instance.
Version      : 0.0.1
Author       :
Help         :
==========================================================================
Checking if ephemeralpool pool exists
Setting AZURE_CONFIG_DIR env variable to: /home/vsts/work/_temp/.azclitask
Setting active cloud to: AzureCloud
/usr/bin/az login --service-principal -u *** -p *** --tenant ***
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "***",
    "id": "44b91933-[redacted]",
    "isDefault": true,
    "managedByTenants": [],
    "name": "SubscripcionBasic",
    "state": "Enabled",
    "tenantId": "***",
    "user": {
      "name": "***",
      "type": "servicePrincipal"
    }
  }
]
Creating container/agent sample-0020202620856 with image ***/azurepipelinesbasicdeployagent/linux:ubuntu-16.04
```

Pipeline preparing the agent in the first job

If we have configured everything correctly, the pipeline will succeed:



#20200314.12 Set up CI with Azure Pipelines

Summary Environments Code Coverage

Manually run by Raúl Piracés

View 7 changes

Repository and version
TestProject
master d088e0a

Time started and elapsed
Today at 23:18
2m 41s

Related
0 work items
0 artifacts

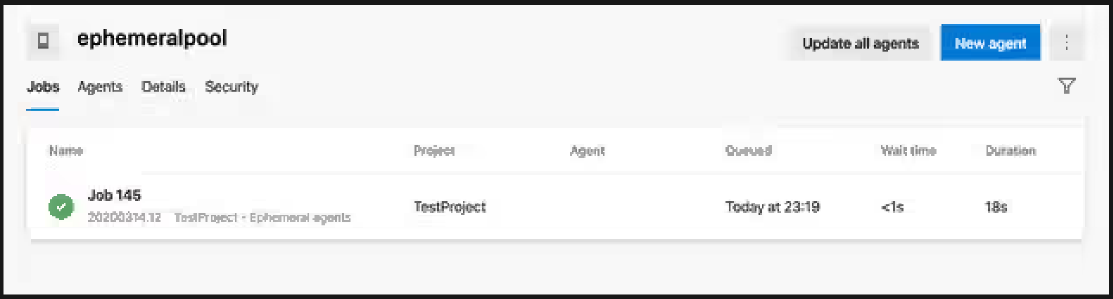
Tests and coverage
Get started

Jobs

Name	Status	Duration
Prepare Agent	Success	2m 14s
Deploy Files	Success	14s

Final pipeline result

If we inspect our created agent pool, we will see the executed job but no agents registered in the pool (which is the main purpose of this process). This is because the agent has unregistered itself when the deployment job finished.



ephemeralpool

Update all agents New agent

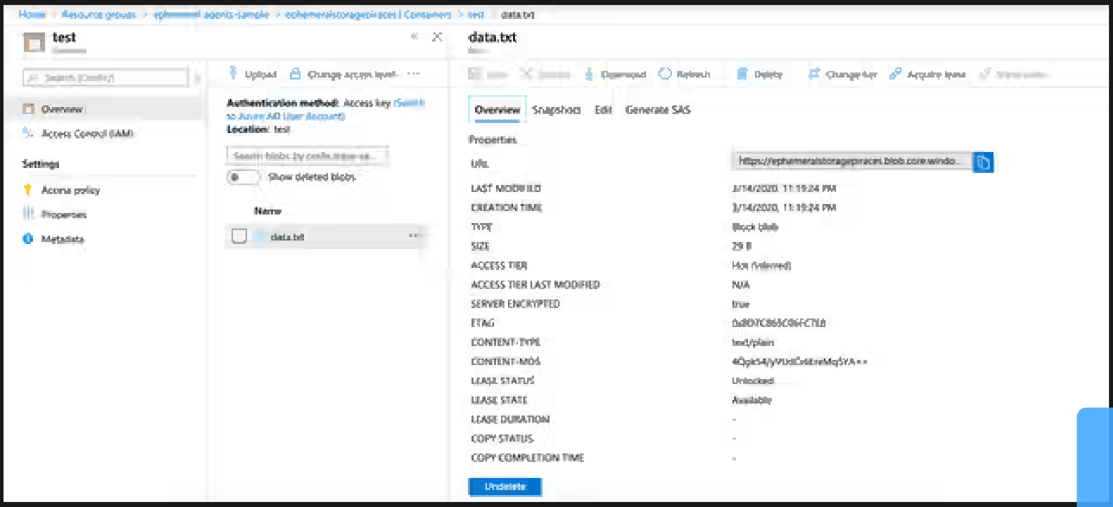
Jobs Agents Details Security

Name	Project	Agent	Queued	Wait time	Duration
Job 145 20200314.12 TestProject - Ephemeral-agents	TestProject		Today at 23:19	<1s	18s

Pool executed job

Final result:

After the deploy pipeline execution, we can check how we successfully have deployed our assets to the Azure Storage container connected to the private virtual network:



Home > Resource groups > ephemeral-agents-sample > ephemeralstorageqpiraces | Containers > test > data.txt

test

Overview
Access Control (IAM)
Settings
Access policy
Properties
Metadata

Upload Change access level

Authentication method: Access key (Switch to Azure AD User Account)
Location: test

Show deleted blobs

Name
data.txt

data.txt

Download Refresh Delete Change tier Acquire lease

Overview Snapshots Edit Generate SAS

Properties

URL	https://ephemeralstorageqpiraces.blob.core.windo...
LAST MODIFIED	3/14/2020, 11:19:24 PM
CREATION TIME	3/14/2020, 11:19:24 PM
TYPE	Block blob
SIZE	29 B
ACCESS TIER	Hot (Inferred)
ACCESS TIER LAST MODIFIED	N/A
SERVER ENCRYPTED	true
ETAG	0x8D7C865C06FC7E8
CONTENT-TYPE	text/plain
CONTENT-MD5	4Qgk54/yVUdCs6EreMq5YA==
LEASE STATUS	Unlocked
LEASE STATE	Available
LEASE DURATION	-
COPY STATUS	-
COPY COMPLETION TIME	-

Undelete

Conclusion

We have seen how ephemeral pipelines agents works, how to replace self-hosted agents with ephemeral ones and the most important part: how to reduce maintenance costs of having our own self-hosted agents pool(s).

It is important to emphasize that this process its currently in preview, so maybe some things just do not work out of the box, but I personally think its an excellent approach to avoid having self-hosted agents in the mentioned situations.

Happy deploy! 🎉🎉

Sources:

1. GitHub of Microsoft Azure Pipelines ephemeral agents. (2020, March 15)
2. Microsoft docs: Azure Pipelines agents. (2020, March 15)
3. Ephemeral Pipelines Agents extension. (2020, March 15)

🙏🙏🙏

Since you've made it this far, sharing this article on your favorite social media network would be highly appreciated 😀! For feedback, please ping me on Twitter.

Published 15 Mar 2020