HOW TO RUN SELF-HOSTED AGENTS WITH YOUR AZURE DEVOPS PIPELINE



Microsoft Azure Devops

Microsoft Azure—Azure for short—is the Microsoft cloud services platform spanning <u>laaS, PaaS, and</u> <u>SaaS services</u> from simple virtualized infrastructure to data warehousing, ML, and AI platforms.

The <u>Azure DevOps service</u> is one such SaaS offering that offers a fully featured <u>DevOps platform</u> consisting of:

- Azure Boards (Planning and Management of the Project)
- Azure Pipelines (CI/CD Pipeline)
- Azure Repos (Cloud-hosted private Git Repositories)
- Azure Test Plans (Manual and Exploratory testing tools)
- Azure Artifacts (Artifact Storage)

These platforms are augmented by a vast collection of extensions to integrate third-party tools and platforms and extend the functionality. <u>CI/CD pipeline</u> is one of the core components to power a software development process provided by the Azure Pipelines service. Azure Pipelines provides the option to use Microsoft-hosted or self-hosted agents to run CI/CD jobs.

In this article, we will look at how to configure self-hosted Azure agents to be used in a pipeline.

Advantages of using self-hosted agents with Azure DevOps

While it may seem a bit strange to use a self-hosted agent with Azure DevOps since it is a cloudbased service, but there are some significant benefits of opting to go with a self-hosted agent.

One reason is cost. Microsoft does offer:

- One free Microsoft-hosted job with 1,800 minutes
- One self-hosted job with unlimited minutes

Though sufficient for small-scale development, most users inevitably need more flexibility to run multiple concurrent builds and releases.

At the time of this article's writing, a Microsoft-hosted <u>agent cost \$40 USD per agent</u> while a self-hosted agent costs \$15, both with unlimited minutes. The self-hosted option provides cost savings when you need to scale up, even when youe add management overhead costs.

The second reason for choosing self-hosting is customizability. You have the freedom to run the agent on any supported operating system, including Windows, Linux, and macOS. Microsoft hosted Azure agents allow you to select a specific image type, but are limited to only what is available from Microsoft.

You have the flexibility to configure agentsand can run multiple agents on a single host to maximize resource usage.

Setting up a self-hosted agent for Azure DevOps

Setting up and running a self-hosted Azure agent is a relatively simple process, with the primary requirement being running the correct agent for the specified operating system and underlying architecture. In this section, we will see how to run agents on a Windows and a Linux VM.

Steps for setting up a self-hosted agent in Azure DevOps

Verify Prerequisites, Hardware and Account Permissions

Before you start the set-up process, ensure your machine is ready. You'll need the following prerequisites:

- The right operating system and version:
 - Client OS
 - Windows 7 SP1 ESU
 - Windows 8.1
 - Windows 10
 - Windows 11
 - Server OS
 - Windows Server 2012 or higher
- <u>PowerShell</u> 3.0 or higher. If you are building from a Subversion repo, you must install the Subversion client on your machine.
- We recommend you also install Visual Studio build tools (2015 or higher)

- Don't worry about .NET, as the agent software will install its own version as part of the set-up process.
- The hardware you use depends on your team size and needs. Most Azure DevOps code is built on 24-core server-class machines, each running four self-hosted agents.

Creating a Personal Access Token (PAT)

The first step before setting up an agent is to create a personal access token which will be used to connect the agent to the Azure Pipeline.

Step 1. Login to Azure DevOps organization, open user settings, and select "Personal access tokens"



Open up your terminal

window and change directory to the folder containing the downloaded file. Inflate the file to view the contents.

First, you need to generate the agent configuration file using the interacting configuration generator script.

Step 2. In the Personal Access Tokens screen, click on "New Token" to create a token.

User settings	Personal Access Tokens These can be used instead of a passwo	ord for applications like Git or can be passed in
	+ New Token	
Account	Token name ↓	Status
^R ≡ Profile	Git: https://	
Sime and Locale	Code (Read & write); Packaging (Read)	Active
O Permissions		
Preferences		
Notifications		
😵 Theme		
Db Usage		
Security		
℅ Personal access tokens		
윢 SSH public keys		
Alternate credentials		
Authorizations		

name, expiration date, and the necessary permissions and click on Create to create the PAT.

Create a new personal access token	×
Name	
self-hosted-agent-token	
Organization	
	~
Expiration (UTC)	
30 days ~ 12/30/2021	
Scopes Authorize the scope of access associated with this token Scopes Full access Custom defined	
Work Items Work items, queries, backlogs, plans, and metadata Read Read & write Read, write, & manage	
Code Source code, repositories, pull requests, and notifications	
🗹 Read 🖉 🧖 Read, write, & 🔽 Full 🗹 Status write	
Build Artifacts, definitions, requests, queue a build, and update build properties Read Read Read & execute	
Show all scopes (28 more) Create Cancel	•

Note: Ensure that

all the correct permissions are granted. Otherwise, you will not be able to initialize the connection. If required, you can configure the agent to have Full access to Azure DevOps.

Create a new personal access token			×
Name			
self-hosted-agent-token			
Organization			
			~
Expiration (UTC)			
30 days	~	12/31/2021	ä
Scopes			
Authorize the scope of access associated with this token			
Scopes O Full access			
Custom defined			

Step 4. Once the

Success!	×
You have successfully added a new persona self-hosted-agent-token token	al access token. Copy the token now!
Marning - Make sure you We don't store it and you again.	copy the above token now. will not be able to see it

token is generated, securely store it as it will not be accessible later.

Installing & configuring the self-hosted agents

Since we have created the token, we can now move into setting up the agent. Any agent configuration can be obtained via the Pipelines Agent pools section in the organizational settings in the Azure DevOps dashboard.

Obtaining Agent Configuration Instructions

Step 1. Navigate to the Organization Settings and select Agent pools from the Pipeline section.



Step 2. Select the Default agent pool. (If needed, select the agent to other available pools or create a new pool and add the agent.)

Agent pools	Ŷ	Security Add pool
Name	Queued jobs	Running jobs
Azure Pipelines Azure Pipelines		
Default		

Step 3. Click on

the New Agent option to obtain the agent installation instructions.



Step 4. Select the

desired operating system and system architecture and follow the instructions provided.

Get the agent			×
w	/indows	macOS	Linux
x64 x86	Configure you Configure your ac Download the Download Create the age	System prerequisites r account count by following the steps outlined <u>here</u> . agent P	5
	PS C:\> mkdir PS C:\agent> [System.IO.Co agent-win-x64 Configure the	r agent ; cd agent Add-Type -AssemblyName System.IO.C ompression.ZipFile]::ExtractToDirec 4-2.195.1.zip", "\$PWD") agent Detailed instructions ≌	ompression.FileSystem ; tory("\$HOME\Downloads\vsts-
	PS C:\agent>	.\config.cmd	
	Optionally run	the agent interactively s a service above:	
	PS C:\agent>	.\run.cmd	
	That's it! More Information		*

Windows Installation of Self-Hosted Agent for Azure

Let's see how to install the Azure agent in Windows 10 on X64 architecture. Please refer to Microsoft's official <u>Windows agent guide</u> for a complete list of prerequisites and specifications.

Step 1. Download the agent. (It will be downloaded as a zip file.)

```
Invoke-WebRequest -Uri
https://vstsagentpackage.azureedge.net/agent/2.195.1/vsts-agent-win-x64-2.195
.1.zip -OutFile vsts-agent-win-x64-2.195.1.zip
```



Step 2. Extract the

downloaded agent to the desired destination. It is recommended that the agent is extracted to a

folder named agents in the root of the C drive (C:agents).

Create directory and navigate to the directory New-Item -Path "C:" -Name "agents" -ItemType "directory" Set-Location -Path "C:agents"

Extract the downloaded zip file Add-Type -AssemblyName System.IO.Compression.FileSystem ; ::ExtractToDirectory("C:vsts-agent-win-x64-2.195.1.zip", "\$PWD")

Verify the extraction
Get-ChildItem



Step 3. Start the

agent configuration by running the following command. (It is recommended to use Elevated Powershell prompt.)

.config.cmd



You will be

required to enter configuration details such as:

- Server URL (Azure organizational URL)
- Authentication type (Here, we have used the previously created authentication token)
- Agent details, including agent-pool and agent name

Finally, specify whether to configure the agent as a Windows service.

You will be able to see the configured Azure Pipelines Agent if you navigate to the Services section on Windows (services.msc).

Q Services		-	o x
File Action View	Help		
🗢 🔿 📅 🖾 🖉	à 🗟 🛛 🖬 🕨 🖩 🖬 🕨		
🔍 Services (Local)	Services (Local)		
	Azure Pipelines Agent	Name	Descriptic ^
	agent-01)	Microsoft Edge Update Service (edgeupdate)	Keeps you Keeps you
	Stop the service	AarSvc_8e3be	Runtime 1
	Restart the service	ActiveX Installer (AxInstSV)	Provides I
		🖾 AllJoyn Router Service	Routes Al
		🚇 App Readiness	Gets apps
		Application Identity	Determin
		Application Information	Facilitates
		Application Layer Gateway Service	Provides :
		Application Management	Processes
		AppX Deployment Service (AppXSVC)	Provides i
		AssignedAccessManager Service	Assigned,
		🖏 ASUS Com Service	ASUS Cor
		AsusCertService	
		🏟 Auto Time Zone Updater	Automati
		AVCTP service	This is Au
		Azure Pipelines Agent (Default.windows-10-agent-01)	
		AzureAttestService	
		Background Intelligent Transfer Service	Transfers :
		Background Tasks Infrastructure Service	Windows
		A Barrier	Manages 🗸
		<	>
	Extended Standard		

Step 4. Navigate

back to the Agent pools in the Organizational settings, and you can see the newly configured agent as the Default pool in the Agents tab.

	Defau	lt						Update all agents	New agent
Jobs	Agents	Details	Security	Settings	Maintenance History	Analytics			
Nar	me			Las	st run	Current status	Agent version	Enabled	
wir • C	ndows-10- Online	agent-01				Idle	2.195.1	•	On

Linux Installation of Self-Hosted Agent for Azure

Installing and configuring the pipeline agent in Linux is similar to Windows. So, in this section, let's see how to install the agent in an Ubuntu environment. Full configuration details are available in the <u>Microsoft documentation</u>.

Step 1. Download the agent

```
wget
https://vstsagentpackage.azureedge.net/agent/2.195.1/vsts-agent-linux-x64-2.1
95.1.tar.gz
```



./config.sh

ubuntu@oc-ubser02-dok:~/Downloads/agent\$./config.sh
Image: 1 Image
>> End User License Agreements:
Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreem ent. This step is not required for building sources from Git repositories.
A copy of the Team Explorer Everywhere license agreement can be found at: /home/ubuntu/Downloads/agent/externals/tee/license.html
Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (press enter for N) > Y
>> Connect:
Enter server URL > <u>https://dev.azure.com/</u> Enter authentication type (press enter for PAT) > PAT Enter personal access token > ***********************************
>> Register Agent:
Enter agent pool (press enter for default) > Enter agent name (press enter for oc-ubser02-dok) > ubuntu-agent-01 Scanning for tool capabilities. Connecting to the server. Successfully added the agent Testing agent connection. Enter work folder (press enter for _work) > 2021-11-30 19:33:23Z: Settings Saved. ubuntu@oc-ubser02-dok:~/Downloads/agent\$

Similar to

Windows configuration, the users will be asked to enter the server details, authentication type, and the authentication token we created earlier. Then configure the agent details, and finally, the user

can start the agent by running the run.sh script.

Step 4 (Optional). You can configure the agent to run as a system service using the svc.sh script located in the agent directory. Specify the user and use the install command to configure the service.

sudo ./svc.sh install ubuntu
sudo ./svc.sh start



back to the Agent pools in the Organizational settings and then to the Default pool of the Agents tab to verify that the new Ubuntu agent is added as a self-hosted agent.

Jobs Agents Details Security Settings Maintenance History Analytics Name Last run Current status Agent version Enabled ubuntu-agent-01 Online Idle 2.195.1 On 	Default			Update al	l agents New agent
Name Last run Current status Agent version Enabled ubuntu-agent-01 • Online Idle 2.195.1 On	lobs Agents Details Sec 	curity Settings Mainte	enance History Analytics		
ubuntu-agent-01 Idle 2.195.1 On • Online	Name	Last run	Current status	Agent version	Enabled
	ubuntu-agent-01 Online		Idle	2.195.1	On

Running your self-hosted agent in Docker

Running the agent as a <u>container</u> is another option we can use to run the agent. Both Windows and Linux are supported as container hosts.

In the following section, let's look at how to create a container image with the Azure pipeline agent and spin up the image as a container. We will be utilizing the Docker Desktop in a Windows environment to create a Linux (Ubuntu) based agent container.

Step 1. Create a folder named dockeragent and then create a Dockerfile within the folder with ubuntu:18.04 as the base image with the required configurations. (The configuration is available via <u>Microsoft documentation</u>.)

FROM ubuntu:18.04

```
# To make it easier for build and release pipelines to run apt-get,
# configure apt to not require confirmation (assume the -y argument by
```

```
default)
ENV DEBIAN FRONTEND=noninteractive
RUN echo "APT::Get::Assume-Yes "true";" > /etc/apt/apt.conf.d/90assumeyes
RUN apt-get update && apt-get install -y --no-install-recommends
ca-certificates
curl
jq
git
iputils-ping
libcurl4
libicu60
libunwind8
netcat
libssl1.0
&& rm -rf /var/lib/apt/lists/*
RUN curl -LsS https://aka.ms/InstallAzureCLIDeb | bash
&& rm -rf /var/lib/apt/lists/*
ARG TARGETARCH=amd64¬¬
ARG AGENT VERSION=2.194.0
WORKDIR /azp
RUN if ; then
AZP AGENTPACKAGE URL=https://vstsagentpackage.azureedge.net/agent/${AGENT VER
SION}/vsts-agent-linux-x64-${AGENT VERSION}.tar.gz;
else
AZP AGENTPACKAGE URL=https://vstsagentpackage.azureedge.net/agent/${AGENT_VER
SION}/vsts-agent-linux-${TARGETARCH}-${AGENT VERSION}.tar.gz;
fi;
curl -Ls¬S "$AZP AGENTPACKAGE URL" | tar -xz
```

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

ENTRYPOINT

Step 2. Create the startup script (start.sh) and put it within the same folder. Ensure that the line endings are configured as Unix-style (LF) line endings.

fi

```
AZP_TOKEN_FILE=/azp/.token
echo -n $AZP_TOKEN > "$AZP_TOKEN_FILE"
fi
```

```
unset AZP_TOKEN
if ; then
mkdir -p "$AZP_WORK"
fi
export AGENT ALLOW RUNASROOT="1"
cleanup() {
if ; then
print header "Cleanup. Removing Azure Pipelines agent..."
# If the agent has some running jobs, the configuration removal process will
fail.
# So, give it some time to finish the job.
while true; do
./config.sh remove --unattended --auth PAT --token $(cat "$AZP TOKEN FILE")
&& break
echo "Retrying in 30 seconds..."
sleep 30
done
fi
}
print_header() {¬
lightcyan='@33[1;36m'
nocolor='@33[Om'
echo -e "${lightcyan}$1${nocolor}"
}
# Let the agent ignore the token env variables
export VS0_AGENT_IGNORE=AZP_TOKEN,AZP_TOKEN_FILE
source ./env.sh
print_header "1. Configuring Azure Pipelines agent..."
./config.sh --unattended
--agent "${AZP_AGENT_NAME:-$(hostname)}"
--url "$AZP URL"
--auth PAT
--token $(cat "$AZP_TOKEN_FILE")
--pool "${AZP POOL:-Default}"
--work "${AZP_WORK:-_work}"
--replace
```

```
--acceptTeeEula & wait $!
print_header "2. Running Azure Pipelines agent..."
trap 'cleanup; exit 0' EXIT
trap 'cleanup; exit 130' INT
trap 'cleanup; exit 143' TERM
# To be aware of TERM and INT signals call run.sh
# Running it with the --once flag at the end will shut down the agent after
the build is executed
./run.sh "$@" & wait $!
```

Step 3. Build the Image by running the following command in the dockeragent folder.

docker build -t dockeragent:latest .



container using the docker run command with the newly created docker image. We can pass <u>environment variables</u> when creating the container. In this instance, we will be passing the server URL (AZP_URL), PAT token (AZP_TOKEN), and agent name (AZP_AGENT_NAME) as variables.

```
docker run -e AZP_URL=https://dev.azure.com/ -e AZP_TOKEN= -e
AZP_AGENT_NAME=docker-agent-01 dockeragent:latest
```

- docker run -e AZP_URL=https://dev.azure.com/ -e AZP_TOKEN=f ama -e AZP_AGENT_NAME=docker-agent=01 dockeragent:latest 1. Configuring Azure Pipelines agent
/ _ \
>> End User License Agreements:
Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building sources from Git repositories.
A copy of the Team Explorer Everywhere license agreement can be found at: /azp/externals/tee/license.html
>> Connect:
Connecting to server
>> Register Agent:

Step 5. We can

verify if the container is added as an agent by looking at the Default agent pool in the Azure DevOps dashboard.

	Defau	lt					Update all agent	ts 🚺	New agent
Jobs	Agents	Details	Security	Settings	Maintenance History Analytics				
Na	me			Last run	Current status	Agent v	ersion	Enabled	
do • (cker-agen Online	t-01			Idle	2.194.0			On

Self-hosted agents for Azure DevOps

Self-hosted agents in Azure DevOps Pipelines offer cost savings and more flexibility to configure and run build and release agents in any supported environment. These pipeline agents can be used to extend the functionality of the CI/CD pipeline from running in bare-metal servers to <u>VMs</u> and even as containers.