Ubuntu on Azure

Canonical Group Ltd

Nov 01, 2023
## CONTENTS

1 Canonical’s offerings 3

2 Project and community 5
   2.1 How-to guides .................................................. 5
   2.2 Understanding Ubuntu on Azure ............................. 30
   2.3 Confidential computing on Azure ............................. 31
Ubuntu on Azure is a set of customised Ubuntu images that allow easy access to a wide range of products and services - offered by both Microsoft Azure and Canonical. These images have an optimised kernel that boots faster, has a smaller footprint and includes Azure-specific drivers.

These images provide a foundation for deploying cloud-based software solutions, specifically for software built on Ubuntu and running on Azure. They focus on providing the optimal tools and features needed to run specific workloads.

The images create a stable and secure cloud platform that is ideal for scaling development work done on Ubuntu-based systems. Since Ubuntu is one of the most favoured operating systems amongst developers, using an Ubuntu-based image for the corresponding cloud deployment becomes the simplest option.

Everyone from individual developers to large enterprises use these images for developing and deploying their softwares. For highly regulated industries from the government, medical and finance sectors, various security-certified images are also available.
Integration with Azure systems - Ubuntu on Azure cloud integrates with the Systems Manager, ensuring that system management tools work natively for instances on the platform. This includes everything from Azure Update Manager and Security Center, to Azure Policy, to using Azure AD to manage your SSH logins. A number of Microsoft products are built on Ubuntu, such as Azure Kubernetes Service, Databricks, and SQL Server on Ubuntu Pro, which includes end-to-end joint support. Furthermore Canonical is working with Microsoft to bring confidential VMs on the cloud on Ubuntu Advantage and Pro. You can find more information on the public preview of AMD-based Confidential VMs.

Customised kernel - The linux-azure kernel enables accelerated networking for the InfiniBand capable instances, as well as consistent support for the Single Root I/O Virtualisation (SR-IOV) on the present hardware, enabling network traffic to bypass the virtualisation stack and achieve almost native performance. It comes with FPGA support out of the box, taking advantage of project catapult to provide performance without the cost and complexity of a custom ASIC.

Some other offerings include:

1. Ubuntu server and Ubuntu Pro for all supported LTS versions across all Azure architectures, including ARM
2. Minimal Ubuntu images for Independent Software Vendors and others who want to build on Azure
3. Ubuntu Pro FIPS images that allow US government Ubuntu users to easily move their workloads to Azure
4. Pre-hardened (CIS) Ubuntu minimal images, for those who want best security practices for Ubuntu out of the box
5. The only commercial Linux launch partner for Azure’s Confidential Computing platform (both AMD SEV-SNP and TDX)
6. SQL Server on Ubuntu Pro images that include specific SQL-related OS optimisations and joint support from Canonical and Microsoft
7. Anbox on Azure, that allows users to run Android apps on Azure at scale
8. Collaboration with Azure’s AKS team to support the Azure Kubernetes worker node image, as these worker nodes nearly always run Ubuntu
9. Collaboration with the Azure Guest Patching Service and Update Manager teams to ensure simple security patch management for users
10. Collaboration with the .Net team on Chiselled .Net images that have a smaller size and security cross-section
PROJECT AND COMMUNITY

Ubuntu on Azure is a member of the Ubuntu family and the project warmly welcomes community projects, contributions, suggestions, fixes and constructive feedback.

- Code of conduct
- Get support
- Join our online chat
- Talk to us about Ubuntu on Azure

2.1 How-to guides

Step-by-step guides for some of the common tasks related to Ubuntu on Azure are available here:

2.1.1 Install Azure CLI on Ubuntu

This documentation is based on the official Azure documentation.

Install a few pre-requisites:

```bash
sudo apt-get update
sudo apt-get install ca-certificates curl apt-transport-https lsb-release gnupg
```

Download the key for the Microsoft archive:

```
mkdir -p /etc/apt/keyrings
curl -sL https://packages.microsoft.com/keys/microsoft.asc |
gpg --dearmor |
sudo tee /etc/apt/keyrings/microsoft.gpg > /dev/null
```

Add the repository to the sources list:

```
SUITE=$(lsb_release -cs)
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/microsoft.gpg] https://packages.microsoft.com/repos/azure-cli/ $SUITE main" |
sudo tee /etc/apt/sources.list.d/microsoft.list
```

Pin a few rules to ensure that the Azure CLI is fetched only from Microsoft’s archive:

```
cat << EOF | sudo tee /etc/apt/preferences.d/99-microsoft
EOF
```

```bash
Pin a few rules to ensure that the Azure CLI is fetched only from Microsoft’s archive:

```
cat << EOF | sudo tee /etc/apt/preferences.d/99-microsoft
EOF
```
Never prefer packages from the Microsoft repository:

```
Package: *
Pin-Priority: 1
```

...except if it is the Azure CLI:

```
Package: azure-cli
Pin-Priority: 500
EOF
```

Finally, install the CLI:

```
sudo apt-get update && \
sudo apt-get install -y azure-cli
```

## 2.1.2 Find Ubuntu images on Azure

### Latest images

### Free long term support offers

To learn more about LTS versions of Ubuntu read this article.

### Ubuntu 22.04 LTS - Jammy Jellyfish

Quick start: 22.04 LTS on Azure
Quick start (China): 22.04 LTS on Azure China

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-server-jammy:22_04-lts:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen1</td>
<td></td>
</tr>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-server-jammy:22_04-lts-gen2:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen2</td>
<td></td>
</tr>
<tr>
<td>Architecture: Arm64</td>
<td>Canonical:0001-com-ubuntu-server-jammy:22_04-lts-arm64:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen2</td>
<td></td>
</tr>
</tbody>
</table>
## Ubuntu 20.04 LTS - Focal Fossa

Quick start: 20.04 LTS on Azure  
Quick start (China): 20.04 LTS on Azure China

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
</table>
| Architecture: AMD64  
Hyper-V Generation: Gen1 | Canonical:0001-com-ubuntu-server-focal:20_04-lts:latest |
| Architecture: AMD64  
Hyper-V Generation: Gen2 | Canonical:0001-com-ubuntu-server-focal:20_04-lts-gen2:latest |
| Architecture: Arm64  
Hyper-V Generation: Gen2 | Canonical:0001-com-ubuntu-server-focal:20_04-lts-arm64:latest |

## Interim releases

To learn more about the difference between LTS releases and interim releases, see this page.

## Ubuntu 23.04 - Lunar Lobster

Quick start: 23.04 on Azure  
Quick start (China): 23.04 on Azure China

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
</table>
| Architecture: AMD64  
Hyper-V Generation: Gen1 | Canonical:0001-com-ubuntu-server-lunar:23_04:latest |
| Architecture: AMD64  
Hyper-V Generation: Gen2 | Canonical:0001-com-ubuntu-server-lunar:23_04-gen2:latest |
| Architecture: Arm64  
Hyper-V Generation: Gen2 | Canonical:0001-com-ubuntu-server-lunar:23_04-arm64:latest |

## Ubuntu 22.10 - Kinetic Kudu

Quick start: 22.10 on Azure  
Quick start (China): 22.10 on Azure China

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
</table>
| Architecture: AMD64  
Hyper-V Generation: Gen1 | Canonical:0001-com-ubuntu-server-kinetic:22_10:latest |
| Architecture: AMD64  
Hyper-V Generation: Gen2 | Canonical:0001-com-ubuntu-server-kinetic:22_10-gen2:latest |
| Architecture: Arm64  
Hyper-V Generation: Gen2 | Canonical:0001-com-ubuntu-server-kinetic:22_10-arm64:latest |

## 2.1. How-to guides
## Ubuntu on Azure

### Ubuntu Pro Offers

To learn more about Ubuntu Pro on Azure: [Ubuntu Pro for Azure](#)

### Ubuntu Pro 22.04 LTS - Jammy Jellyfish

Quick start: Ubuntu Pro 22.04 LTS on Azure

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-jammy:pro-22_04-lts:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen1</td>
<td></td>
</tr>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-jammy:pro-22_04-lts-gen2:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen2</td>
<td></td>
</tr>
</tbody>
</table>

### Ubuntu Pro 20.04 LTS - Focal Fossa

Quick start: Ubuntu Pro 20.04 LTS on Azure

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-focal:pro-20_04-lts:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen1</td>
<td></td>
</tr>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-focal:pro-20_04-lts-gen2:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen2</td>
<td></td>
</tr>
</tbody>
</table>

### Ubuntu Pro 18.04 LTS - Bionic Beaver

Quick start: Ubuntu Pro 18.04 LTS on Azure

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-bionic:pro-18_04-lts:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen1</td>
<td></td>
</tr>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-bionic:pro-18_04-lts-gen2:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen2</td>
<td></td>
</tr>
</tbody>
</table>

### Confidential compute capable offer

To learn more about Confidential Compute: [Azure confidential computing](#)
Ubuntu on Azure

Ubuntu CVM 22.04 LTS - Jammy Jellyfish

Quick start: Ubuntu CVM 22.04 LTS on Azure
Quick start (China): Ubuntu CVM 22.04 LTS on Azure China

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyper-V Generation:</td>
<td>Gen2</td>
</tr>
</tbody>
</table>

Ubuntu CVM 20.04 LTS - Focal Fossa

Quick start: Ubuntu CVM 20.04 LTS on Azure
Quick start (China): Ubuntu CVM 20.04 LTS on Azure China

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture:</td>
<td>AMD64</td>
</tr>
<tr>
<td>Hyper-V Generation:</td>
<td>Gen2</td>
</tr>
</tbody>
</table>

FIPS compliant offers

To learn more about FIPS: FIPS for Ubuntu

Ubuntu Pro FIPS 20.04 LTS - Focal Fossa

Quick start: Ubuntu Pro FIPS 20.04 LTS on Azure

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture:</td>
<td>AMD64</td>
</tr>
<tr>
<td>Hyper-V Generation:</td>
<td>Gen1</td>
</tr>
<tr>
<td>Architecture:</td>
<td>AMD64</td>
</tr>
<tr>
<td>Hyper-V Generation:</td>
<td>Gen2</td>
</tr>
</tbody>
</table>

Ubuntu Pro FIPS 18.04 LTS - Bionic Beaver

Quick start: Ubuntu Pro FIPS 18.04 LTS on Azure

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture:</td>
<td>AMD64</td>
</tr>
<tr>
<td>Hyper-V Generation:</td>
<td>Gen1</td>
</tr>
<tr>
<td>Architecture:</td>
<td>AMD64</td>
</tr>
<tr>
<td>Hyper-V Generation:</td>
<td>Gen2</td>
</tr>
</tbody>
</table>
Ubuntu on Azure

CIS Hardened offer

To learn more about CIS compliance: CIS compliance with Ubuntu LTS

Ubuntu Minimal Pro CIS 20.04 LTS - Focal Fossa

Quick start: Ubuntu Pro CIS 20.04 LTS on Azure

<table>
<thead>
<tr>
<th>Kind</th>
<th>URN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-minimal-cis-focal:pro-cis-minimal-20_04:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen1</td>
<td>Canonical:0001-com-ubuntu-pro-minimal-cis-focal:pro-cis-minimal-20_04-gen1:latest</td>
</tr>
<tr>
<td>Architecture: AMD64</td>
<td>Canonical:0001-com-ubuntu-pro-minimal-cis-focal:pro-cis-minimal-20_04-gen2:latest</td>
</tr>
<tr>
<td>Hyper-V Generation: Gen2</td>
<td>Canonical:0001-com-ubuntu-pro-minimal-cis-focal:pro-cis-minimal-20_04-gen2:latest</td>
</tr>
</tbody>
</table>

List all images published by Canonical

 Canonical publishes Ubuntu images under the Canonical publisher ID. You can find all our images using this az command:

```
az vm image list -p Canonical --all -o table
```

Via the portal, make sure to look for Canonical rather than Ubuntu to find the official Ubuntu images. Also, always make sure the offer is published by Canonical. Non-Pro LTS offers are always FREE.

2.1.3 Get Ubuntu Pro on Azure

What is Ubuntu Pro?

Ubuntu Pro is an additional stream of security updates and packages that meet compliance requirements such as FIPS or HIPAA, on top of an Ubuntu LTS. To learn more about Ubuntu Pro you can read this FAQ.

How to get Ubuntu Pro on Azure?

New instances

For new instances or instances that are easy to redeploy (e.g. launched programatically in a CI/CD pipeline), the best option is to redeploy onto a new Azure Ubuntu Pro instance. These Ubuntu Pro instances on Azure attach to their entitlements automatically and will receive all the Pro features by default without further steps. Pro offers on Azure are listed at: Ubuntu Pro Offers.
Running instances

It is possible to upgrade an Ubuntu Server LTS instance to receive Ubuntu Pro entitlements by buying a token and attaching this with `sudo pro attach [YOUR_TOKEN]` on the relevant instance. The best way to purchase tokens for Azure instances is to contact Canonical.

### 2.1.4 Create Ubuntu Pro FIPS golden image with Azure Image Builder

This guide will provide instructions for using the Azure Image Builder (AIB) to create an Ubuntu Pro 20.04 FIPS “golden” image in an Azure Compute Gallery, (formerly Shared Image Gallery). In the process, you’ll:

- Set up an environment with an Azure Compute Gallery (ACG) and the necessary resources to distribute an image from it
- Create an image definition for Ubuntu Pro 20.04 FIPS
- Create a build configuration template to add optional applications
- Create the golden image using the AIB service
- Create a VM from the golden image in the ACG

**Note:** We are using a pre-enabled FIPS image, but you can also use the standard Ubuntu Pro if it better suits your needs.

### What you’ll need

- A Microsoft Azure account
- *Azure Command-Line Interface*

### Set up your Azure Compute Gallery

To set up the compute gallery, you’ll need to create a resource group, a user-identity, a gallery and finally the image definition itself. To simplify the process, we start by creating some variables for values that’ll be used repeatedly.

### Set up variables

A new resource group with an unused name will have to be created. It can be deleted after use.

**Note:** To create a custom image, the AIB must be in the same resource group as the source-managed image.

```bash
# Resource group name
sigResourceGroup=ibUbuntuFIPSGalleryRG

# Datacenter location
location=westus2

# Additional region for image replication
additionalregion=eastus
```

Create variables for gallery name and image definition name. The image will be displayed in the Azure Portal as `sigName/imageDefName`.

2.1. How-to guides
Ubuntu on Azure

# Name of the ACG
sigName=myIbGallery
# Name of the image definition to be created
imageDefName=myIbImageDef
# Image distribution metadata reference name
runOutputName=aibUbuntuSIG

Create a variable for your subscription ID:

subscriptionID=$(az account show --query id --output tsv)

Set up variables for the Ubuntu Pro plan to be used. If you have an Ubuntu Pro private offer with Canonical that includes 24x7 technical support with SLAs, you’ll have a custom offer and sku, which can be used here. If not, as seen in the example below, you can use the details from the Ubuntu Pro 20.04 FIPS image that is publicly available at the Azure Marketplace.

# Set the 'Publisher' field
ProPlanPublisher=canonical
# Set the 'Offer' field
ProPlanOffer=0001-com-ubuntu-pro-focal-fips
# ProPlanSku the 'Sku'
ProPlanSku=pro-fips-20_04-gen2

Create required resources, identities and permissions

Create the resource group:

az group create -n $sigResourceGroup -l $location --subscription $subscriptionID

AIB needs a user-identity to inject an image into ACG. So first create an Azure role definition with actions needed to distribute an image to ACG, and then assign that role definition to the user-identity.

# Create a user-identity
identityName=aibBuiUserId$(date +'%s')
az identity create -g $sigResourceGroup -n $identityName --subscription $subscriptionID

# Get the user-identity ID
imgBuilderCliId=$(az identity show -g $sigResourceGroup -n $identityName --subscription $subscriptionID -o json | grep "clientId" | cut -c16- | tr -d ",")

# Get the user-identity URI
imgBuilderId=/subscriptions/$subscriptionID/resourcegroups/$sigResourceGroup/providers/Microsoft.ManagedIdentity/userAssignedIdentities/$identityName

# Download an Azure role definition template
curl https://raw.githubusercontent.com/Azure/azvmimagebuilder/master/solutions/12_Creating_AIB_Security_Roles/aibRoleImageCreation.json -o aibRoleImageCreation.json

imageRoleDefName="Azure Image Builder Image Def"$(date +'%s')

# Update the role definition template with the correct subscription ID, resource group, and role definition name

(continues on next page)
Create an image definition and gallery

To use AIB with ACG, you’ll need to have an existing gallery and an image definition.

First, create a gallery:

```
az sig create \
  -g $sigResourceGroup \n  --gallery-name $sigName \n  --subscription $subscriptionID
```

Then, create an image definition. Ensure that the “hyper-v-generation” flag is set to the same generation as the base image that you plan to use.

```
az sig image-definition create \
  -g $sigResourceGroup \n  --gallery-name $sigName \n  --gallery-image-definition $imageDefName \n  --publisher $ProPlanPublisher \n  --offer $ProPlanOffer \n  --sku $ProPlanSku \n  --os-type Linux \n  --plan-name $ProPlanSku \n  --plan-product $ProPlanOffer \n  --plan-publisher $ProPlanPublisher \n  --hyper-v-generation V2 \n  --subscription $subscriptionID
```
Create a configuration template

We’ll be using a sample JSON template to configure the image. It can be customised to include build instructions that are specifically needed for your golden image. Download a template:

```
curl https://pastebin.com/raw/fCkQAgAc -o UbuntuProFips2004SIGTemplate.json
```

Customise it to use the values set above:

```
sed -i -e "s/<subscriptionID>/$subscriptionID/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<rgName>/$sigResourceGroup/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<imageDefName>/$imageDefName/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<sharedImageGalName>/$sigName/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<region1>/$location/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<region2>/$additionalregion/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<runOutputName>/$runOutputName/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s%<imgBuilderId>%$imgBuilderId%g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<ProPlanPublisher>/$ProPlanPublisher/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<ProPlanOffer>/$ProPlanOffer/g" UbuntuProFips2004SIGTemplate.json
sed -i -e "s/<ProPlanSku>/$ProPlanSku/g" UbuntuProFips2004SIGTemplate.json
```

Review the template content

In case you want to change something or add your own actions, some of the following sections might help. The plan details of the VM image being used as a starting point for your golden image are under ‘source’:

```
"source": {
  "type": "PlatformImage",
  "publisher": "canonical",
  "offer": "0001-com-ubuntu-pro-focal-fips",
  "sku": "pro-fips-20_04-gen2",
  "version": "latest",
  "planInfo": {
    "planName": "pro-fips-20_04-gen2",
    "planProduct": "0001-com-ubuntu-pro-focal-fips",
    "planPublisher": "canonical"
  }
},
```

The `customize` section allows you to run commands as part of the image building process. The command seen here is used to include a wait until Ubuntu’s `ua` client is attached to its subscription.

```
"customize": [
  {
    "type": "Shell",
    "name": "WaitForUAtokenAutoAttach",
    "inline": [
      "sudo ua status --wait"
    ]
  },
```

Within this section you can add your own actions, for say hardening the image or installing specific software.
The following commands deregister the golden image from Ubuntu Pro and remove the machine-id. This will ensure that VMs generated from the golden image will generate their own unique IDs.

```json
{
  "type": "Shell",
  "name": "DetachUA -- images created from this will auto attach themselves with new credentials",
  "inline": [
    "sudo ua detach --assume-yes && sudo rm -rf /var/log/ubuntu-advantage.log"
  ]
},
{
  "type": "Shell",
  "name": "Replace /etc/machine-id with empty file to ensure UA client does not see clones as duplicates",
  "inline": [
    "sudo rm -f /etc/machine-id && sudo touch /etc/machine-id"
  ]
}
```

### Create the golden image

To create the image in ACG, submit the image configuration to the AIB service:

```
az resource create \
  --resource-group $sigResourceGroup \
  --subscription $subscriptionID \
  --properties @UbuntuProFips2004SIGTemplate.json \
  --is-full-object \
  --resource-type Microsoft.VirtualMachineImages/imageTemplates \
  -n UbuntuProFips2004SIG01
```

Accept the legal terms of the image:

```
az vm image terms accept --plan $ProPlanSku --offer $ProPlanOffer --publisher $ProPlanPublisher --subscription $subscriptionID
```

Start the image build process:

```
az resource invoke-action  \
  --resource-group $sigResourceGroup \
  --subscription $subscriptionID \
  --resource-type Microsoft.VirtualMachineImages/imageTemplates
```

(continues on next page)
This step can take some time (~25 minutes) as Azure will actually launch a VM and run the steps that you have defined. While you are waiting for the AIB build process to complete, you can view the corresponding logs by going to the storage account inside the resource group created by AIB. (i.e. Go to Azure Portal > Resource groups > IT_ibUbuntuFIPSGalleryRG_*** > Random ID of the storage account > Containers > packerlogs > Random ID of the container > customization.log > Download)

Once the build process is completed, the status will change from “Running” to “Succeeded”, to show something like:

```json
{
   "endTime": "2022-09-10T23:13:25.9008064Z",
   "name": "37962BEF-34DC-45B1-A1C6-E827CE20F89B",
   "startTime": "2022-09-10T22:48:19.7520483Z",
   "status": "Succeeded"
}
```

### Create a VM - using the Portal

To create a VM based on the golden image, in the portal:

1. Go to Azure services > Virtual Machines > Create > Virtual machine
2. Open the See all images link located below the Image field drop down
3. Select Shared Images from the column on the left
4. Choose your golden image and it should now be the selected image in the Image field
5. Complete the remaining fields as per your requirements and select Review + Create

### Create a VM - using the CLI

To create a VM from the command line, you’ll need to use all the variables created earlier. If you already have an SSH key use the following commands to launch the VM:

```
SSHPublicKeyPath=<path to your id_rsa.pub>

az vm create \
   --resource-group $sigResourceGroup \ 
   --subscription $subscriptionID \ 
   --name myAibGalleryVM \ 
   --admin-username aibuser \ 
   --location $location \ 
   --image "/subscriptions/$subscriptionID/resourceGroups/$sigResourceGroup/providers/Microsoft.Compute/galleries/$sigName/images/$imageDefName/versions/latest" \ 
   --ssh-key-values $SSHPublicKeyPath \ 
   --plan-name $ProPlanSku \ 
   --plan-product $ProPlanOffer \ 
   --public-ip-sku Standard \ 
   --plan-publisher $ProPlanPublisher
```
Alternatively, if you do not have an SSH key, replace the \(--ssh-key-values \$SSHPublicKeyPath\) with \(--generate-ssh-keys\) as shown below. However this may overwrite the ssh keypair \texttt{id_rsa} and \texttt{id_rsa.pub} located in .ssh in your home directory.

```
az vm create \
  --resource-group $sigResourceGroup \
  --subscription $subscriptionID \
  --name myAibGalleryVM \
  --admin-username aibuser \
  --location $location \
  --image "/subscriptions/$subscriptionID/resourceGroups/$sigResourceGroup/providers/Microsoft.Compute/galleries/$sigName/images/$imageDefName/versions/latest" \
  --generate-ssh-keys \
  --plan-name $ProPlanSku \
  --plan-product $ProPlanOffer \
  --public-ip-sku Standard \
  --plan-publisher $ProPlanPublisher
```

Once the command completes, you should see something like:

```
{
  "fqdns": "",
  "id": "/subscriptions/50a71625-6dba-43a2-87ad-9eb26e52c9c4/resourceGroups/ibUbuntuFIPSGalleryRG/providers/Microsoft.Compute/virtualMachines/myAibGalleryVM",
  "identity": {
    "principalId": "632b2bcf9-9d93-46da-bbd1-3b32e85f92eb",
    "tenantId": "40a524d9-f848-46d4-a96f-be6df491fe15",
    "type": "SystemAssigned",
    "userAssignedIdentities": null
  },
  "location": "westus2",
  "macAddress": "00-0D-3A-F5-29-B8",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "51.143.126.x",
  "resourceGroup": "ibUbuntuFIPSGalleryRG",
  "zones": ""
}
```

You can use the \texttt{publicIpAddress} (51.143.126.x in this case) to ssh into the machine. To check that the VM is attached to an Ubuntu Pro subscription and is running a FIPS kernel, run:

```
sudo ua status --wait
```
Ubuntu on Azure

Post creation cleanup

You now have an Azure Compute Gallery with an Ubuntu Pro 20.04 FIPS image inside. You have also launched and tested a VM based on this golden image. So you can go ahead with the deletion of the resource groups that were created. You should be able to see the created resource groups with:

```bash
az group list --query [].name --output table --subscription $subscriptionID | grep $sigResourceGroup
```

This command returns something like:

```bash
ibUbuntuFIPSGalleryRG
IT_ibUbuntuFIPSGalleryRG_UbuntuProFips2004S_02ecb26b-21f4-4450-b207-e86c7fd6853e
```

If you want to delete these resource groups, use the following command on each of them. You may find that deleting the first one automatically deletes the second.

```bash
az group delete --name [the name from above] --subscription $subscriptionID
```

2.1.5 Install Kubeflow on AKS

Kubeflow is a novel open-source end-to-end Machine Learning tool that runs on Kubernetes. It is composed of 30+ microservices, and can be challenging to deploy and operate. However, the process is greatly simplified using Juju and this guide describes:

- How to deploy Kubeflow on AKS
- How to observe the state of your deployment using Juju and
- How to access your Kubeflow dashboard from your local machine

Basic requirements

- Access to an AKS Kubernetes cluster via kubectl
- A minimum of 4 CPUs, 16GB RAM and 50GB Disk space should be available in your cluster

Install the Juju client

Juju provides a simple installation of Kubeflow across Kubernetes platforms, with a good level of customisation, as well as easy maintenance. For further details check out Charmed Kubeflow. To use the Juju CLI, install the Juju client. On Linux, install it via snap using:

```bash
sudo snap install juju --classic
```

Alternatively, download the Windows installer or use brew install juju on macOS
**Connect Juju to your AKS cluster**

To operate workloads in your Kubernetes cluster with Juju, you have to add your cluster to the list of clouds in Juju via the `add-k8s` command. If your Kubernetes config file is in the standard location (`~/.kube/config` on Linux), and you only have one cluster, you can simply run:

```
juju add-k8s myk8s
```

If your `kubectl` config file contains multiple clusters, you can specify the appropriate one by name:

```
juju add-k8s myk8s --cluster-name=foo
```

Finally, if your config file is in a different location, you can set the `KUBECONFIG` environment variable to point to the relevant file. For example:

```
KUBECONFIG=path/to/file juju add-k8s myk8s
```

**Create a controller**

To operate workloads on your Kubernetes cluster, Juju uses controllers. You can create a controller with the `bootstrap` command:

```
juju bootstrap myk8s my-controller
```

This command creates a couple of pods under the my-controller namespace. You can see your controllers with the `juju controllers` command.

**Create a model**

A model in Juju is a blank canvas where your operators are deployed, and it holds a 1:1 relationship with a Kubernetes namespace. You can create a model and give it a name, e.g. `kubeflow`, with the `add-model` command. In the process you are also creating a Kubernetes namespace of the same name:

```
juju add-model kubeflow
```

You can list your models with the `juju models` command.

**Deploy Kubeflow**

*Note:* To deploy Kubeflow you need at least 50GB of disk space, 14GB of RAM and 2 CPUs on your cluster. If you have fewer resources, deploy `kubeflow-lite` or `kubeflow-edge`.

Once you have a model, you can simply `juju deploy` any of the provided Kubeflow bundles into your cluster. For the full Kubeflow bundle, run:

```
juju deploy kubeflow --trust
```

You can observe your Kubeflow deployment process with the command:

```
watch -c juju status --color
```

To customise your deployment, use the [docs on customisation](#).
Final deployment steps

There are currently a couple of additional steps required to effectively deploy Kubeflow.

Add an RBAC role for Istio

To setup Kubeflow with Istio correctly, you need to provide the `istio-ingressgateway` operator access to Kubernetes resources. This is done by creating an appropriate Role Based Access Control (RBAC) role:

```bash
kubectl patch role -n kubeflow istio-ingressgateway-operator -p '{"apiVersion":"rbac.authorization.k8s.io/v1","kind":"Role","metadata":{"name":"istio-ingressgateway-operator"},"rules":[{"apiGroups":[]","resources":[]","verbs":[]}]}
```

Find the external IP address of your Kubeflow dashboard

To get the IP address of your Kubeflow dashboard run:

```bash
kubectl get svc/istio-ingressgateway -n kubeflow
```

where `kubeflow` is the name that you specified for your Juju model, and is in turn the namespace of your Kubeflow deployment. Save the returned IP address as `EXTERNAL-IP` for use in the next step.

Provide the external IP to authentication services

To enable access to your dashboard, provide its public IP to `dex-auth` and `oidc-gatekeeper`:

```bash
juju config dex-auth public-url=http://<EXTERNAL-IP>:80
juju config oidc-gatekeeper public-url=http://<EXTERNAL-IP>:80
```

Access the Kubeflow dashboard

To view your authentication credentials, run:

```bash
juju config dex-auth static-username
juju config dex-auth static-password
```

By default, these are both empty. If you wish to set them, add the relevant string to the end of the command, e.g.

```bash
juju config dex-auth static-username=admin
juju config dex-auth static-password=AxWiJjk2hu4ffga7
```

Assuming you have configured your virtual network’s firewall to allow you to connect, you should be able to access your Kubeflow dashboard URL. The AKS cluster’s ingress will take you to the login page of your Charmed Kubeflow MLOps platform.
2.1.6 Deploy Kubeflow pipelines with AKS spot instances

Charmed Kubeflow is an MLOps platform that delivers an end-to-end solution for AI/ML applications. It includes Kubeflow Pipelines, an engine for orchestrating different MLOps workflows.

Kubeflow pipelines can be created using Azure spot instances on an AKS cluster. This can save costs as long as appropriate tasks are chosen for the spot instances and their eviction is handled gracefully.

- **Use spot instances for:**
  - data processing
  - distributed training & hyperparameter tuning
  - model training (with checkpointing)
  - batch inference

- **Don’t use spot instances for:**
  - Kubernetes control plane
  - notebooks and dashboards
  - datastores or databases
  - a model serving as an online inference

To create Kubeflow pipelines with spot instances, start with an AKS cluster containing an existing deployment of Charmed Kubeflow. Refer to Install Kubeflow on AKS for instructions.

**Add spot instances**

In the cluster details screen of your Kubeflow deployment on AKS, go to Settings > Node pools > Add node pool:

Select **Enable Azure Spot instances**:

---

2.1. How-to guides
Ubuntu on Azure

Specify when and how your node should be evicted - Eviction type and Eviction policy. If you have a maximum price over which it is not worth running your workload, then specify that:

Create the pool after specifying all required fields. Once the VMs are up, check if all the nodes are available in the portal’s Node pools tab or on the command line using kubectl get nodes.
Update pipeline to allow task execution on spot instances

To enable tasks to execute on spot instances, use the Kubernetes *toleration* mechanism. Add an appropriate toleration to each task and if the tasks have to run on a certain type of node (say GPU required), include an *affinity* configuration to enforce the type of pods allowed.

**Sample code**

To try out the changes needed, you can clone this Kubeflow-examples repository and use the notebooks from the aks-spot-instances folder to create new notebooks in your Kubeflow deployment. Run the first notebook (01-base-workload) and use the Kubeflow Pipelines UI to see the workflow run details:

![Kubeflow pipeline](image)

To check where the workflow steps were executed run `kubectl get pods -n admin -o wide` and in the *NODE* column, you’ll see that all the tasks were executed on agentpool:

```
NAME            READY   STATUS      RESTARTS   AGE            NODE             NOMINATED NODE   READINESS GATES
base_pipeline-1   0/2     Completed 8 4h25m   18.244.4.41         agentpool-1   <none>           <none>
```

The code used to create the pipeline was:

```python
@dsl.pipeline(
    name="base_pipeline",
    description="my base pipeline",
)
def my_pipeline(url):
    web_downloader_task = web_downloader_op(url=url)
    merge_csv_task = create_step_merge_csv(file=web_downloader_task.outputs['data'])

client = kfp.Client()

client.create_run_from_pipeline_func(
    my_pipeline,
    arguments={
        'url': 'https://storage.googleapis.com/ml-pipeline-playground/iris-csv-files.tar.gz'
    },
)
```

Now, run the second notebook (02-spot-workload) to use spot instances. The pipeline definition in this code includes *toleration* and *affinity*. This configuration is done at the pipeline level and is applicable to each task, but the tasks themselves are unaware of it.
from kubernetes.client.models import V1Toleration, V1Affinity, V1NodeAffinity,
V1NodeSelector, V1NodeSelectorTerm, V1NodeSelectorRequirement

@dsl.pipeline(
    name="base_pipeline",
    description="my base pipeline",
)
def my_pipeline(url):
    web_downloader_task = web_downloader_op(url=url)
    merge_csv_task = create_step_merge_csv(file=web_downloader_task.outputs['data'])

    toleration = V1Toleration(
        key="kubernetes.azure.com/scalesetpriority",
        operator="Equal",
        value="spot",
        effect="NoSchedule",
    )
    web_downloader_task.add_toleration(toleration)
    merge_csv_task.add_toleration(toleration)

    aff = V1Affinity(node_affinity=V1NodeAffinity(
        required_during_scheduling_ignored_during_execution=V1NodeSelector(
            node_selector_terms=[V1NodeSelectorTerm(
                match_expressions=[V1NodeSelectorRequirement(
                    key='kubernetes.azure.com/scalesetpriority',
                    operator='In',
                    values=['spot']))]))
    )
    web_downloader_task.add_affinity(aff)
    merge_csv_task.add_affinity(aff)

client = kfp.Client()

client.create_run_from_pipeline_func(
    my_pipeline,
    arguments={
        'url': 'https://storage.googleapis.com/ml-pipeline-playground/iris-csv-files.tar.gz'
    })

The results seen will be the same as before except for the node names. Run kubectl get pods -n admin -o wide |
grep "spot-pipeline" to see that the tasks were executed on the spotpool.

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Reason</th>
<th>&quot;aks-spotpool-24665937-vvss000000&quot;</th>
<th>&quot;aks-spotpool-24665937-vvss000000&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/2</td>
<td>Completed</td>
<td>0</td>
<td>38s 18.244.1.5  &quot;aks-spotpool-24665937-vvss000000&quot;</td>
<td>&quot;aks-spotpool-24665937-vvss000000&quot;</td>
</tr>
</tbody>
</table>

Chapter 2. Project and community
Handle eviction gracefully

To handle spot eviction gracefully, set the retry policy for all tasks that are to be executed on a spot instance. This ensures that when the task fails, the pipeline will retry the task after waiting for a while (backoff_duration). It’ll keep doing this until a specified count of maximum retries (num_retries) is reached.

You can use the third notebook (03-spot-retries-workload) to try this out. Here the backoff duration is set to 5 minutes to give enough time for a new node to spawn and the max retry count is set to 5:

```python
from kubernetes.client.models import V1Toleration, V1Affinity, V1NodeAffinity, 
    V1NodeSelector, V1NodeSelectorTerm, V1NodeSelectorRequirement

@dsl.pipeline(
    name="base_pipeline",
    description="my base pipeline",
)
def my_pipeline(url):
    web_downloader_task = web_downloader_op(url=url)
    merge_csv_task = create_step_merge_csv(file=web_downloader_task.outputs['data'])

    toleration = V1Toleration(
        key="kubernetes.azure.com/scalesetpriority",
        operator="Equal",
        value="spot",
        effect="NoSchedule",
    )
    web_downloader_task.add_toleration(toleration)
    merge_csv_task.add_toleration(toleration)

    aff = V1Affinity(node_affinity=V1NodeAffinity(
        required_during_scheduling_ignored_during_execution=V1NodeSelector(
            node_selector_terms=[V1NodeSelectorTerm(
                match_expressions=[V1NodeSelectorRequirement(
                    key='kubernetes.azure.com/scalesetpriority',
                    operator='In',
                    values=['spot']))])
        )
    )
    web_downloader_task.add_affinity(aff)
    merge_csv_task.add_affinity(aff)

    web_downloader_task.set_retry(num_retries=5, backoff_dureation="5m")
    merge_csv_task.set_retry(num_retries=5, backoff_dureation="5m")

client = kfp.Client()

client.create_run_from_pipeline_func(
    my_pipeline,
    arguments={
        'url': 'https://storage.googleapis.com/ml-pipeline-playground/iris-csv-files.tar.gz',
    })
```
2.1.7 Upgrade from Focal to Jammy

General Advice

Once you have decided to upgrade your system, the next question is how? There are two options depending on whether your system is setup/deployed with automation or whether it requires manual configuration.

For fully automated system deployments it is recommended to redeploy with new Jammy instances instead of upgrading from Focal.

For systems that cannot be easily created or destroyed and require manual configuration, running `do-release-upgrade` is a good option. However this option requires some manual intervention as explained below.

Manual intervention steps

While upgrading from Focal to Jammy, manual decision making will be needed for the following options that are presented.

Additional SSH daemon

When upgrading in a session over SSH there is an inherent risk of losing access if something goes wrong with the SSH daemon. To mitigate this risk an additional SSH daemon is started on a different port as a backup.

The prompt notifies you that an additional SSH daemon will be started and you can either continue or cancel the upgrade.

Optional firewall rules for additional SSH daemon

If you are using a firewall there is a chance that the port used by the backup SSHD is not open. Opening this port is not done automatically since it could be security risk. An optional command to open the port is provided and you are prompted to press enter to continue.
Start upgrade

A final prompt is provided before starting the upgrade. It gives information about the number of changes and the estimated time to complete because once started, the upgrade process cannot be cancelled. At this stage you can continue, cancel or see additional details.
Restart services automatically

During the upgrade of certain libraries, some services have to be restarted. You have the option of allowing the services to be restarted automatically during the upgrade. If you select ‘no’ here, you’ll be asked about the services that you want to restart after each library upgrade.

Chrony configuration modified

Canonical makes changes to /etc/chrony/chrony.conf for Azure images. As a result, during upgrade you’ll see a prompt notifying you about the availability of a newer version of the sshd_config file. You’ll be asked if you want to keep the existing modified version, use the default one from the new upgrade or take some other action.
**SSHD configuration modified**

Canonical makes changes to `/etc/ssh/sshd_config` for AWS EC2 images. As a result, during upgrade you’ll see a prompt notifying you about the availability of a newer version of the sshd_config file. You’ll be asked if you want to keep the existing modified version, use the default one from the new upgrade or take some other action.

![Configuring openssh-server](image)

**Remove obsolete packages**

An obsolete package is a package which is no longer available in any of the sources for apt. Usually it is safe and recommended to remove obsolete packages. But before doing so you’ll be asked if you wish to remove them and you’ll have the option to select from yes, no and more details.

![Searching for obsolete software](image)

**Restart to finish upgrade**

Finally, a restart will be necessary for some parts of the upgrade to be applied. If you select no, you can use `/var/run/reboot-required.pkgs` to check for the packages that need a reboot.

![System upgrade is complete](image)
2.2 Understanding Ubuntu on Azure

2.2.1 Is Ubuntu available on Azure?

Yes, all the supported versions of Ubuntu are available for free on Azure. See: Find Ubuntu images on Azure.

2.2.2 Why are there multiple offers from Canonical on Azure?

For technical reasons related to the publication process, Canonical publishes different versions of Ubuntu under dedicated offers. This is to ensure that the publication of a given version of Ubuntu cannot block or impact the publication of another version.

2.2.3 How often are Ubuntu images refreshed?

Canonical publishes a new version of an image every time the kernel for this image is updated. On an average, this happens once every three weeks. Important security or bug fixes might also trigger an image refresh. However, running virtual machines (VMs) are not affected by these changes. Use `apt` to keep your VM up to date and reboot your VM regularly to update the running kernel.

When using the CLI or any automated process, use the keyword `latest` in place of the image version. This ensures that you will always launch the latest image available for the given offer/SKU.

2.2.4 How often are images deprecated?

Microsoft Azure Partner Center has a hard limit of 100 images. To comply with this policy, Canonical has the following deprecation policy

- No more than 90 active images will be kept for a publication
- When the limit of 90 images is reached, Canonical will deprecate the oldest version to allow publishing of the latest version
- Images will have a 90 day deprecation time. During this time, users will receive warnings from Microsoft about the deprecation
- After 90 days, the deprecated images will be fully removed, and users will no longer have access.

2.2.5 Is Ubuntu on Azure different from Ubuntu on other clouds?

Yes, Ubuntu on Azure is customised to make it better for Azure. This customisation includes:

- A custom kernel `linux-azure` developed by Canonical for Azure
- Extra configuration files that allow packages to work better with the platform
- A few extra pre-installed packages that ensure built-in support for all features of Azure
2.2.6 Why are there so many publishers of Ubuntu on Azure?

Ubuntu is a collection of free and open source software, and redistribution is permitted in accordance with Ubuntu’s intellectual property policy.

We do, however, advise caution when installing third-party Ubuntu images. You need to ensure that you trust the third party and the security of their image building pipeline. If you are paying for a third-party Ubuntu image, make sure that you not only trust it, but that you are receiving value beyond what you could obtain for free from Canonical or as part of Ubuntu Pro.

2.3 Confidential computing on Azure

Confidential computing addresses the question of trust between cloud providers and their users. The idea is to hide and protect sensitive workloads. Users should be allowed to run programs on untrusted systems with the technical assurance that the cloud provider cannot read nor modify the program’s data and memory. While it might not be entirely possible to satisfy these requirements, the solutions at least ensure that modification of data is detected.

Data can be thought of as being in one of three states:

- in-transit - being transmitted from one location to another
- at-rest - stored somewhere and
- in-use - being used by a CPU to perform some operation

Both data in-transit and data at-rest can be encrypted using well-known techniques, but securing data in-use needs confidential computing.

Confidential computing is the protection of data in-use by performing computation in a hardware-based trusted execution environment. These are secure and isolated environments that prevent unauthorised access or modification of applications and data while they are in use. In effect, they allow the encryption of data while it is in the system memory. This requires the support of both hardware and the OS. For example, your hardware could be based on Intel’s TDX (Trust Domain Extensions) processors or AMD’s SEV (Secure Encrypted Virtualisation) architecture and the OS could be Ubuntu.

Azure provides two types of security:

- Trusted launch which is a set of features including virtual Trusted Platform Module (vTPM) and secure boot
- Confidential virtual machine with support for AMD Secure Encrypted Virtualisation-Secure Nested Paging (SEV-SNP), along with measured boot using a vTPM

2.3.1 Trusted launch

All Ubuntu images from 20.04 (Focal Fossa) support trusted launch and secure boot on Hyper-V Gen2 instances. To start an Ubuntu instance with vTPM and secure boot enabled, use the following flags from the Azure CLI:

```
--security-type TrustedLaunch --enable-secure-boot true --enable-vtpm
```
2.3.2 Confidential VM

What are confidential VMs?

Check out our technical blog post about Confidential VMs on Azure.

In short, a confidential VM is a combination of two features:

- **Memory encryption**: The virtual machine’s memory is encrypted and decrypted on the fly by the CPU. This is done using AMD SEV-SNP technology. This ensures that the host machine cannot read the memory of the guest since the encryption is done at the hardware level.

- **vTPM backed full-disk encryption (FDE)**: The file system of the guest OS is encrypted at rest and the key is stored in an enclave. The key cannot be retrieved unless the boot process has stayed untouched since the last sealing of the key. Re-sealing happens when a new kernel is installed as replacing the kernel binary changes the boot sequence. The vTPM is part of the guest VM and its address space. So it benefits from the same run-time security guarantees as the guest VM memory.

It’s important to note that memory encryption is always enabled with a confidential VM, but FDE is optional and requires explicit activation after the VM is provisioned.

Using Ubuntu on confidential VMs

Confidential VMs require the use of special instance sizes and a special version of Ubuntu.

- A list of instance sizes that can be used for confidential VMs is given in Azure’s documentation.

- Only this specific offer of Ubuntu supports confidential VMs.