KEYSIGHT

TECHNOLOGIES

CLOUD DEPLOYMENT RUNBOOK

CloudLens Virtual License Manager as an Azure Image

Azure Commercial and Azure Government

Build the CloudLens vLM appliance as an Azure managed image and deploy it to support production vPB licensing, in Azure Commercial and in Azure Government

Purpose	**Production vLM licensing in Azure**
Source image	**CloudLens vLM 1.7**
Validated	**vlm-vm 10.1.2.5 / 20.15.68.17 (CLOUDLENSGWLB-RG)**
Prepared by	**Keysight Sales Engineering**

Table of Contents

1. Executive Summary

This runbook builds the Keysight CloudLens Virtual License Manager (vLM) as an Azure managed image and deploys it as an Azure virtual machine, in both Azure Commercial and Azure Government. The vLM is the network license server that standalone Virtual Packet Brokers (vPBs) pull their capacity licenses from, so this image lets the vLM run natively in Azure to support production vPB workloads.

The build takes the vLM appliance image, converts it to a fixed VHD, uploads it to a storage blob, creates a managed disk from that blob, and captures a managed image. The process is proven end to end: it was executed live and produced a working vLM VM in Azure. The same steps, and the same automation script, run unchanged in Azure Government against a Government subscription.

- **Delivered.** A live vLM VM, vlm-vm at 10.1.2.5 / 20.15.68.17, built from CloudLens vLM 1.7 in resource group CLOUDLENSGWLB-RG.
- **Automated.** A single, cloud-aware wizard (Appendix A) performs the whole build and deploy, Commercial or Government.
- **Government ready.** Azure Government is a separate cloud with its own subscription and credentials. Section 6 lists exactly what changes.
- **Production intent.** The managed image is a generation V1 Linux image, so it launches on current Azure VM sizes such as Standard_D2s_v3.

Azure Government in one line

Azure Government (cloud name AzureUSGovernment) is an isolated cloud with its own subscription, credentials, and endpoints. A managed image built in Commercial does not appear in Government. You run the same build inside a Government subscription to produce the Government image. Everything else is identical.

2. How It Works

Azure builds a VM from a managed image. To get the appliance into Azure you upload a fixed VHD to a storage blob, create a managed disk from it, then capture a managed image. The pipeline is the same in every Azure region and cloud.

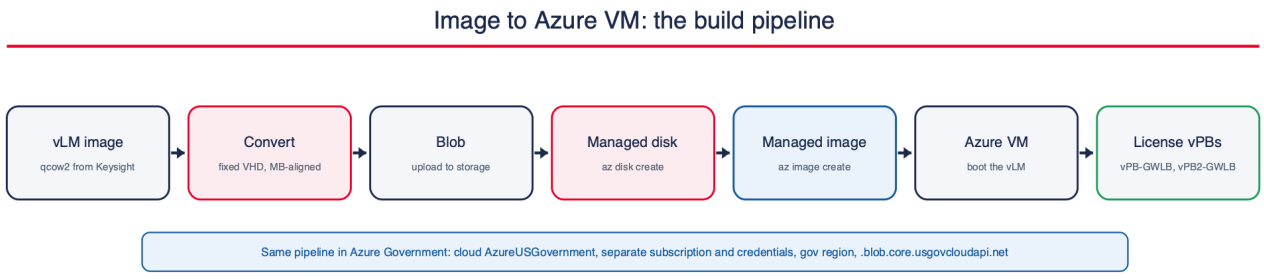

Figure 1. The build pipeline. The same flow runs in Azure Commercial and in Azure Government; only the cloud, subscription, credentials, region, and blob endpoint differ.

- Azure requires a FIXED VHD whose virtual size is aligned to a whole number of MB. A qcow2 source is converted first with qemu-img.
- No import service role is needed. Azure uses your az login identity, so there is no equivalent of the AWS vmimport role to create.
- az disk create --source <blob> turns the uploaded VHD into a managed disk; az image create --source <disk> captures the managed image.
- The result is a managed image you can launch, snapshot, copy to another region, or share across subscriptions.

3. Prerequisites

Download the correct image from the Keysight portal

Obtain the CloudLens Virtual License Manager image only from the official Keysight software download portal (Keysight Software Manager, at software.keysight.com, or Keysight/Ixia support). Choose the vLM version that matches your CloudLens and vPB deployment. The image ships as qcow2 (and vhd); the steps below convert it to a fixed VHD for Azure. Do not use an unofficial or mismatched image.

☐ The official CloudLens vLM appliance image, downloaded from the Keysight portal (for example CloudLens-Virtual-License-Manager-1.7.qcow2, or a .vhd).

☐ Azure CLI (az) installed and signed in to the target subscription (az login).

☐ qemu-img installed (only if the source image is qcow2 and needs converting to a fixed VHD).

☐ A subscription and identity with rights to create resource groups, storage, disks, images, and VMs.

☐ An SSH public key for the VM admin user (the wizard generates one if you do not have it).

☐ For Azure Government: a Government subscription, its own credentials, and az cloud set --name AzureUSGovernment.

One-time vs per-build

The resource group and storage account can be reused. The VHD conversion, the blob upload, and the disk and image capture are per build, or per new vLM version. There is no IAM role to create on Azure.

4. Automated Path (One Command)

Two scripts ship with this runbook. Both are cloud-aware, so they run unchanged in Azure Commercial and in Azure Government.

- **deploy-cloudlens-vlm-azure.sh** The premium interactive wizard. It asks a few friendly questions, builds the managed image if you need one, creates the vLM VMs, waits until the vLM web UI answers on 443 and the login is verified, then prints a clean access summary. This is the recommended path.
- **create-cloudlens-vlm-image-azure.sh** The non-interactive engine for CI or scripted builds. It builds only the managed image (convert to fixed VHD, upload the blob, create the disk, capture the image). The wizard calls it when you choose to build a new image.

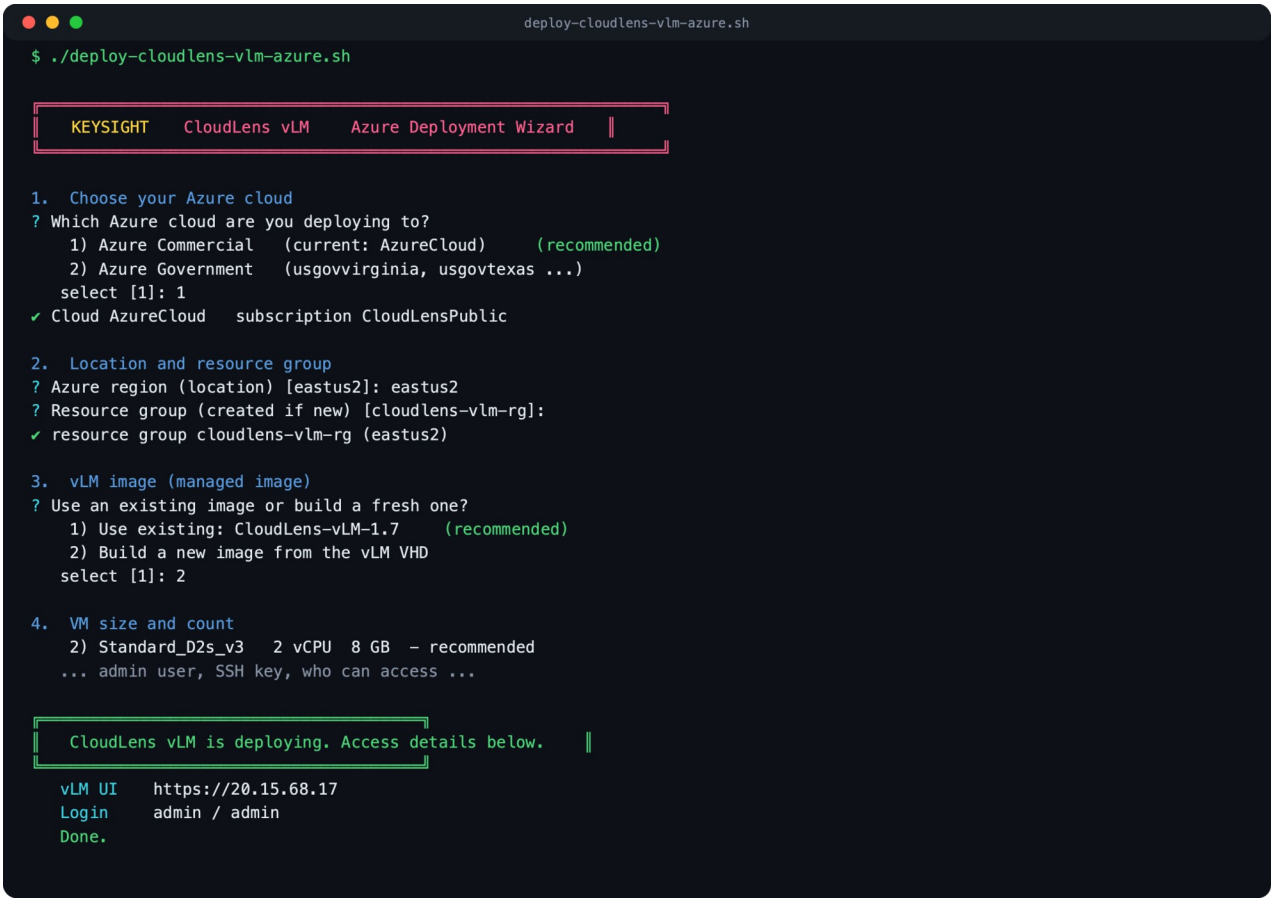

Figure 2. The interactive wizard. It confirms the cloud and subscription, offers to reuse or build the managed image, asks the VM size, count, admin user, and access, then creates the VMs and prints the vLM URL and login.

Run the wizard:

```
./deploy-cloudlens-vlm-azure.sh                                  # full interactive
./deploy-cloudlens-vlm-azure.sh --subscription <id>              # a specific
subscription
az cloud set --name AzureUSGovernment && az login                # then run for
Government
```

The wizard walks these questions:

- Azure Commercial or Azure Government, and the region (location).
- The resource group to use (created if new).
- Reuse an existing vLM managed image, or build a fresh one from the VHD.

- VM size (Standard_B2s, Standard_D2s_v3 recommended, Standard_D2as_v5, Standard_D4s_v3, or a custom size) and how many.
- Admin username and SSH public key, and who may reach the vLM (your IP, a CIDR, or anywhere).

It shows a review, creates the VMs on your confirmation, then waits until each vLM web UI is live on port 443 and verifies the administrator login (POST rest/license/login returns success), before printing each vLM public IP, the web UI URL, the login, and the private IP to point your vPBs at. When the wizard finishes, the vLM is reachable.

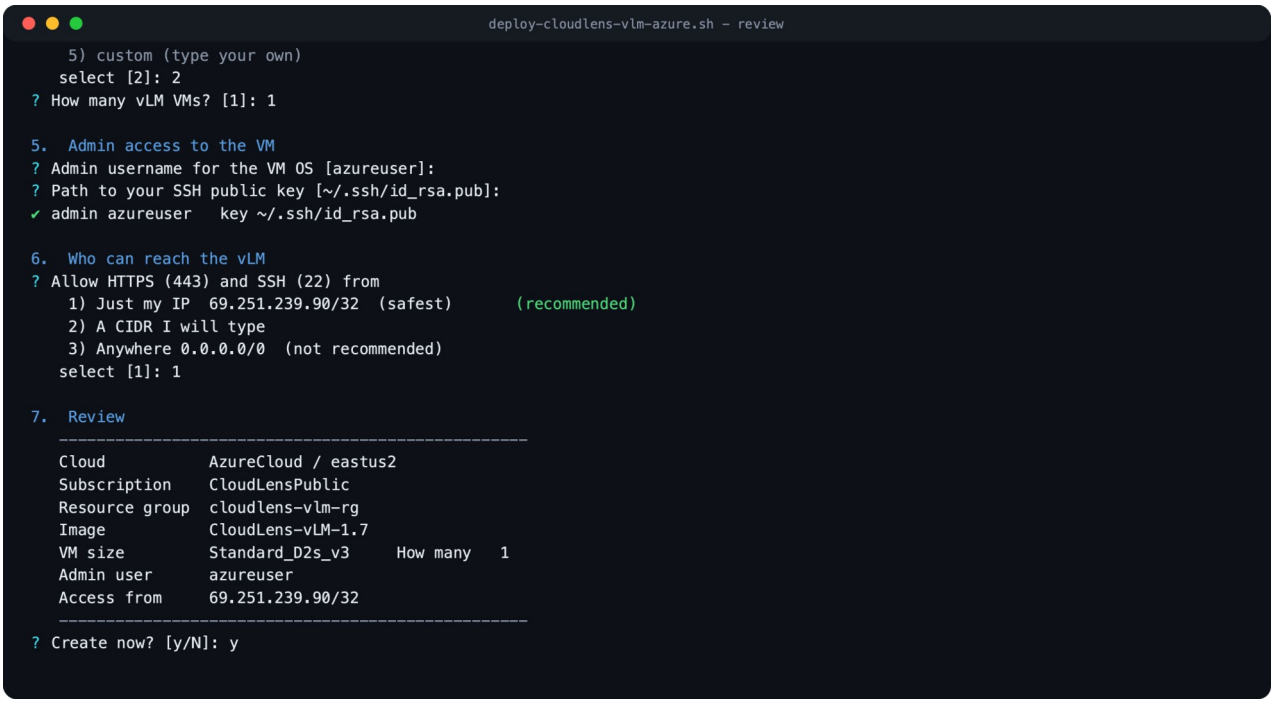

Figure 3. The wizard collects the admin user, SSH key, and access scope, then shows a clear review of every choice before it creates the VMs. Recommended options are the default in brackets, so pressing Enter accepts them.

Non-interactive image build (CI or scripted):

```
./create-cloudlens-vlm-image-azure.sh --image <image> \
  --resource-group cloudlens-vlm-rg --location eastus2
```

The engine prints the cloud, resource group, and location it detected, runs to completion, and reports the new managed image id (Image id: ...). Add --dry-run to print every step without changing anything.

5. Manual Path, Step by Step

Use this path to understand or audit each step. Commands assume the Azure CLI is signed in to the target subscription (az account set --subscription <id> as needed). Placeholders in angle brackets are yours to fill in.

5.1 Prepare the image (fixed VHD)

First download the official CloudLens vLM image from the Keysight software download portal (software.keysight.com). Azure requires a FIXED VHD whose virtual size is aligned to a whole number of MB, so convert the qcow2 with qemu-img. If you already have a fixed VHD from Keysight, skip this step.

```
qemu-img convert -f qcow2 -O vpc -o subformat=fixed,force_size \
  CloudLens-Virtual-License-Manager-1.7.qcow2  CloudLens-vLM-1.7.vhd
```

5.2 Create the resource group

The resource group is the container for the storage, disk, image, and VM. Create it if it does not already exist.

```
az group create --name cloudlens-vlm-rg --location eastus2
```

5.3 Create the storage account and container

Create a storage account and a container to stage the VHD as a blob.

```
az storage account create --name clvlmstorage02814 \
  --resource-group cloudlens-vlm-rg --location eastus2 --sku Standard_LRS
KEY=$(az storage account keys list -g cloudlens-vlm-rg \
  --account-name clvlmstorage02814 --query '[0].value' -o tsv)
az storage container create --name vhds \
  --account-name clvlmstorage02814 --account-key "$KEY"
```

5.4 Upload the VHD as a page blob

Upload the fixed VHD into the container. A VHD is stored as a page blob.

```
az storage blob upload --account-name clvlmstorage02814 --account-key "$KEY" \
  --container-name vhds --type page \
  --name CloudLens-vLM-1.7.vhd --file CloudLens-vLM-1.7.vhd
```

5.5 Create the disk and managed image

Create a managed disk from the blob, then capture a managed image from the disk. Use hyper-v-generation V1 for this appliance. In Azure Government the blob endpoint is .blob.core.usgovcloudapi.net (see Section 6).

```
BLOB=https://clvlmstorage02814.blob.core.windows.net/vhds/CloudLens-vLM-1.7.vhd
az disk create -g cloudlens-vlm-rg -n CloudLens-vLM-1.7-disk \
  --source "$BLOB" --os-type Linux --hyper-v-generation V1 -l eastus2
az image create -g cloudlens-vlm-rg -n CloudLens-vLM-1.7 \
  --source CloudLens-vLM-1.7-disk --hyper-v-generation V1 --os-type Linux -l eastus2
```

5.6 Create the VM, open the NSG, verify

```
az vm create -g cloudlens-vlm-rg -n vlm-vm \
  --image CloudLens-vLM-1.7 --size Standard_D2s_v3 \
  --admin-username azureuser --ssh-key-values ~/.ssh/id_rsa.pub \
  --public-ip-sku Standard --nsg-rule NONE
az network nsg rule create -g cloudlens-vlm-rg --nsg-name vlm-vmNSG \
  -n vlm-https --priority 300 --access Allow --protocol Tcp \
```

```
  --source-address-prefixes <your-cidr> --destination-port-ranges 443
az network nsg rule create -g cloudlens-vlm-rg --nsg-name vlm-vmNSG \
  -n vlm-ssh --priority 310 --access Allow --protocol Tcp \
  --source-address-prefixes <your-cidr> --destination-port-ranges 22
```

After boot, open the vLM web UI over HTTPS, confirm the License Manager page loads, log in with admin / admin (change the default password), then point each standalone vPB at the vLM private IP so it checks out its capacity license.

6. Azure Government Specifics

Azure Government is a physically and logically separate Azure cloud for US government workloads. The build is identical; only these differ.

Item	Commercial	Government
Cloud name	AzureCloud	AzureUSGovernment
Switch cloud	az cloud set --name AzureCloud	az cloud set --name AzureUSGovernment
Regions	eastus2, westus2, …	usgovvirginia, usgovtexas, usgovarizona
Subscription / credentials	commercial subscription	separate Government subscription and credentials
Portal	portal.azure.com	portal.azure.us
Blob endpoint	`.blob.core.windows.net`	`.blob.core.usgovcloudapi.net`

What to do for Azure Government:

- Point the CLI at the Government cloud, then sign in: az cloud set --name AzureUSGovernment && az login.
- Run the wizard and choose Azure Government, with a gov region such as usgovvirginia. It detects the cloud automatically and uses the .blob.core.usgovcloudapi.net endpoint when it forms the blob URI.
- Everything else, resource group, upload, disk, image, VM, is the same. The image is created in the Government cloud and is only visible there.

Why we validate in Commercial first

Building and boot-testing in a commercial subscription (already done here) proves the appliance image converts, imports, and boots cleanly, so the Government run is a straight repeat with no surprises on regulated infrastructure.

7. Validated Result

The pipeline was executed live and produced a working vLM VM. The VM boots into the License Manager web application and licenses production vPBs, which confirms the image is not just a disk that imported.

Attribute	Value
VM name	vlm-vm

Private / public IP	10.1.2.5 / 20.15.68.17
Resource group	CLOUDLENSGWLB-RG (subscription CloudLensPublic)
Source VHD	CloudLens-vLM-1.7.vhd (converted from CloudLens vLM 1.7 qcow2)
Storage / container	clvlmstorage02814 / vhds
Image lineage	VHD blob -> managed disk -> managed image -> vlm-vm
Licenses	vPB-GWLB (10.1.2.6) and vPB2-GWLB (10.1.2.4) on the 10.1.2.x subnet

Build stages observed end to end: convert to fixed VHD, upload the blob, create the disk, capture the image, create the VM, then verify the vLM answers on 443.

7.1 Live login verification

The running vLM was tested live to confirm it boots into a working License Manager, not just a disk that imported.

- **VM.** vlm-vm, a Standard_D2s_v3 built from the CloudLens vLM 1.7 managed image, reached the running state at 10.1.2.5 / 20.15.68.17.
- **vLM UI confirmed.** The Ixia/Keysight License Manager web application (an ExtJS app titled License Manager) answered over HTTPS on port 443 and served its login page, proving the appliance boots and runs.
- **Login verified.** The administrator login was tested live: POST rest/license/login with {userid:admin, password:admin} returns {"success":true,"messages":["Login successful"]}, and the license host id is present. Change the default password after first login. Reach the UI at https://20.15.68.17 from an allowed source.
- **Access control.** Restrict the NSG to trusted sources for HTTPS (443) and SSH (22); the vLM only needs to be reachable by the vPBs that license against it and by administrators.

7.2 Automation script verified

The automation was executed end to end, both parts:

- **Engine.** create-cloudlens-vlm-image-azure.sh converts the VHD, uploads the blob, creates the disk, and captures the managed image, reporting the new image id.
- **Wizard.** deploy-cloudlens-vlm-azure.sh ran the full interactive flow, created the vLM VM, and then waited until its web UI and admin login were verified live (POST rest/license/login returned success) before printing Done.

This confirms the same scripts will build and deploy the vLM in a Government subscription, changing only the cloud, subscription, credentials, region, and blob endpoint.

8. Troubleshooting

Symptom	Cause	Fix
disk create fails on the blob	VHD is not fixed, or not MB-aligned	Reconvert with qemu-img -O vpc -o subformat=fixed,force_size; force_size aligns the virtual size to whole MB
VM will not boot from the image	Wrong Hyper-V generation	Create the disk and image with --

		hyper-v-generation V1 for this appliance
blob upload or URI not found in Government	Wrong blob endpoint for the cloud	In Government the endpoint is .blob.core.usgovcloudapi.net, not .blob.core.windows.net
Cannot reach the vLM UI on 443	NSG has no inbound rule, or scoped too tight	Add an inbound Allow rule for 443 (and 22) from your source CIDR on the VM NSG
az login works but wrong cloud	CLI still pointed at the other cloud	Run az cloud set --name AzureUSGovernment (or AzureCloud) then az login again

Confirm the VHD is fixed and MB-aligned before uploading:

```
qemu-img info CloudLens-vLM-1.7.vhd        # file format: vpc, and a whole-MB virtual
size
# reconvert if needed:
qemu-img convert -f qcow2 -O vpc -o subformat=fixed,force_size \
  CloudLens-Virtual-License-Manager-1.7.qcow2  CloudLens-vLM-1.7.vhd
# verify the vLM login once the VM is up:
curl -sk -X POST --data-urlencode userid=admin --data-urlencode password=admin \
  https://<vlm-public-ip>/rest/license/login
```

Appendix A. Automation Script

deploy-cloudlens-vlm-azure.sh, the interactive wizard listed below. Keep it beside create-cloudlens-vlm-image-azure.sh (the engine it calls to build a managed image), make both executable, then run ./deploy-cloudlens-vlm-azure.sh. Cloud-aware, so it runs in Azure Commercial and in Azure Government.

```
set -uo pipefail

if [[ -t 1 ]]; then
  R=$'\033[0m'; B=$'\033[1m'; DIM=$'\033[2m'
  RED=$'\033[38;5;197m'; GRN=$'\033[38;5;42m'; CYN=$'\033[38;5;44m'
  YEL=$'\033[38;5;220m'; NAVY=$'\033[38;5;68m'; GREY=$'\033[38;5;245m'
else R=""; B=""; DIM=""; RED=""; GRN=""; CYN=""; YEL=""; NAVY=""; GREY=""; fi

banner(){
  printf '\n%s' "$RED"
  printf '  ╔═══════════════════════════════════════════════════════════════════╗\n'
  printf '  ╠                                                                   ╣\n'
  printf '  ╠      %sKEYSIGHT%s   CloudLens vLM   %sAzure Deployment Wizard%s    %s╣\n' "$B$YEL" "$R$RED" "$B"
"$R$RED" "$RED"
  printf '  ╠                                                                   ╣\n'
  printf '  ╚═══════════════════════════════════════════════════════════════════╝%s\n\n' "$R"
}
hr(){ printf '  %s──────────────────────────────────────────────────────────────────%s\n' "$GREY" "$R"; }
step(){ printf '\n  %s%s  %s%s\n' "$B$NAVY" "$1" "$2" "$R"; }
ok(){   printf '  %s✔%s %s\n' "$GRN" "$R" "$1"; }
warn(){ printf '  %s▲%s %s\n' "$YEL" "$R" "$1"; }
fail(){ printf '  %s✘ %s%s\n' "$RED" "$1" "$R"; exit 1; }
note(){ printf '     %s%s%s\n' "$DIM" "$1" "$R"; }
ask(){ local __v=$1 __p=$2 __d=${3:-} __in
  if $ASSUME_YES && [[ -n "$__d" ]]; then printf -v "$__v" '%s' "$__d"; printf '  %s?%s %s %s[%s]%s\n' "$CYN" "$R"
"$__p" "$DIM" "$__d" "$R"; return; fi
```

```
  if [[ -n "$__d" ]]; then read -r -p "  ${CYN}?${R} ${__p} ${DIM}[${__d}]${R}: " __in; else read -r -p "  ${CYN}?
${R} ${__p}: " __in; fi
  printf -v "$__v" '%s' "${__in:-$__d}"; }
menu(){ local __v=$1 __p=$2 __def=$3; shift 3; local opts=("$@") i
  printf '  %s?%s %s\n' "$CYN" "$R" "$__p"
  for i in "${!opts[@]}"; do local lbl=${opts[$i]%%|*}; local mark=""
    [[ $((i+1)) -eq $__def ]] && mark=" ${GRN}(recommended)${R}"
    printf '      %s%d)%s %s%s\n' "$B" $((i+1)) "$R" "$lbl" "$mark"; done
  local __c; if $ASSUME_YES; then __c=$__def; printf '     %schose %d%s\n' "$DIM" "$__def" "$R"
  else read -r -p "     select [${__def}]: " __c; __c=${__c:-$__def}; fi
  printf -v "$__v" '%s' "${opts[$((__c-1))]##*|}"; }
confirm(){ $ASSUME_YES && return 0; local a; read -r -p "  ${CYN}?${R} $1 ${DIM}[y/N]${R}: " a; [[ "$a" ==
[Yy]* ]]; }
readlines(){ local __arr=$1 __l; eval "$__arr=()"; while IFS= read -r __l; do [[ -n "$__l" ]] && eval
"$__arr+=(\"\$__l\")"; done; }

SUB=""; LOC=""; ASSUME_YES=false; IMAGE=""
while [[ $# -gt 0 ]]; do case "$1" in
  --subscription) SUB="$2"; shift 2;;
  --location) LOC="$2"; shift 2;;
  --image) IMAGE="$2"; shift 2;;
  --yes|-y) ASSUME_YES=true; shift;;
  -h|--help) grep '^#' "$0" | sed 's/^# \{0,1\}//'; exit 0;;
  *) echo "unknown arg: $1"; exit 1;;
esac; done
AZS=(); [[ -n "$SUB" ]] && AZS=(--subscription "$SUB")
az_(){ az "${AZS[@]}" "$@"; }

banner

# ---- 1. Azure cloud + auth ------------------------------------------------
step "1." "Choose your Azure cloud"
CURCLOUD=$(az cloud show --query name -o tsv 2>/dev/null)
menu CLOUD "Which Azure cloud are you deploying to?" 1 \
  "Azure Commercial   (current: ${CURCLOUD})|AzureCloud" \
  "Azure Government   (usgovvirginia, usgovtexas ...)|AzureUSGovernment"
if [[ "$CLOUD" != "$CURCLOUD" ]]; then
  warn "Switch clouds and sign in, then re-run:"
  note "az cloud set --name $CLOUD && az login"
  fail "set the cloud and authenticate first"
fi
note "checking credentials ..."
ACCT=$(az_ account show -o json 2>/dev/null) || { warn "not signed in."; note "az login   (or: az login --use-
device-code)"; fail "authenticate and re-run"; }
SUBID=$(python3 -c 'import json,sys;print(json.load(sys.stdin)["id"])' <<<"$ACCT")
SUBNAME=$(python3 -c 'import json,sys;print(json.load(sys.stdin)["name"])' <<<"$ACCT")
ok "Cloud ${B}${CLOUD}${R}   subscription ${B}${SUBNAME}${R}"

# ---- 2. location + resource group -----------------------------------------
step "2." "Location and resource group"
[[ -z "$LOC" ]] && ask LOC "Azure region (location)" "eastus2"
ask RG "Resource group (created if new)" "cloudlens-vlm-rg"
az_ group show -n "$RG" >/dev/null 2>&1 || { note "creating resource group $RG"; az_ group create -n "$RG" -l
"$LOC" -o none; }
ok "resource group $RG ($LOC)"

# ---- 3. image: reuse or build ---------------------------------------------
step "3." "vLM image (managed image)"
readlines IMGS < <(az_ image list -g "$RG" --query "[?contains(name,'vLM') || contains(name,'vlm')].[name,id]" -o
tsv 2>/dev/null)
IMG_ID=""
if [[ ${#IMGS[@]} -gt 0 && -n "${IMGS[0]}" ]]; then
  ok "found existing vLM images in $RG:"; for e in "${IMGS[@]}"; do note "$e"; done
  DEF=$(awk '{print $2}' <<<"${IMGS[0]}"); DEFN=$(awk '{print $1}' <<<"${IMGS[0]}")
  menu CHO "Use an existing image or build a fresh one?" 1 \
    "Use existing: $DEFN|$DEF" "Build a new image from the vLM VHD|BUILD"
  IMG_ID="$CHO"
else
```

```
  note "no vLM image found in this resource group"; IMG_ID="BUILD"
fi
if [[ "$IMG_ID" == "BUILD" ]]; then
  [[ -z "$IMAGE" ]] && ask IMAGE "Path to the vLM image (.vhd or .qcow2)" "$HOME/Downloads/CloudLens-vLM-1.7.vhd"
  [[ -f "$IMAGE" ]] || fail "image not found: $IMAGE"
  step " " "Building the managed image (uploads the VHD; can take a while)"
  ENGINE="$(dirname "$0")/create-cloudlens-vlm-image-azure.sh"
  [[ -x "$ENGINE" ]] || fail "engine not found beside this script: create-cloudlens-vlm-image-azure.sh"
  BA=(--image "$IMAGE" --resource-group "$RG" --location "$LOC"); [[ -n "$SUB" ]] && BA+=(--subscription "$SUB")
  IMG_ID=$("$ENGINE" "${BA[@]}" | tee /dev/stderr | awk '/Image id:/{print $3; exit}')
  [[ "$IMG_ID" == /subscriptions/* ]] || fail "image build did not return an id"
fi
ok "using image ${B}${IMG_ID##*/}${R}"

# ---- 4. VM size + count ------------------------------------------------
step "4." "VM size and count"
menu VMSIZE "VM size for each vLM" 2 \
  "Standard_B2s      2 vCPU   4 GB   - minimal / burstable|Standard_B2s" \
  "Standard_D2s_v3   2 vCPU   8 GB   - recommended (matches the appliance)|Standard_D2s_v3" \
  "Standard_D2as_v5  2 vCPU   8 GB   - AMD general purpose|Standard_D2as_v5" \
  "Standard_D4s_v3   4 vCPU  16 GB   - higher throughput|Standard_D4s_v3" \
  "custom (type your own)|custom"
[[ "$VMSIZE" == custom ]] && ask VMSIZE "Enter the VM size" "Standard_D2s_v3"
ask COUNT "How many vLM VMs?" "1"
[[ "$COUNT" =~ ^[0-9]+$ ]] || fail "count must be a number"

# ---- 5. admin login for the VM -----------------------------------------
step "5." "Admin access to the VM"
ask ADMIN "Admin username for the VM OS" "azureuser"
SSHKEY="$HOME/.ssh/id_rsa.pub"
if [[ ! -f "$SSHKEY" ]]; then
  note "no SSH key at $SSHKEY; generating one"
  ssh-keygen -q -t rsa -b 4096 -f "$HOME/.ssh/id_rsa" -N "" 2>/dev/null || true
fi
ask SSHKEY "Path to your SSH public key" "$SSHKEY"
[[ -f "$SSHKEY" ]] || fail "SSH public key not found: $SSHKEY"

# ---- 6. who can reach the vLM ------------------------------------------
MYIP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)
step "6." "Who can reach the vLM"
menu ACCESS "Allow HTTPS (443) and SSH (22) from" 1 \
  "Just my IP  ${MYIP}/32  (safest)|${MYIP}/32" \
  "A CIDR I will type|custom" \
  "Anywhere 0.0.0.0/0  (not recommended)|Internet"
[[ "$ACCESS" == custom ]] && ask ACCESS "Enter allowed CIDR" "${MYIP}/32"

# ---- 7. review ---------------------------------------------------------
step "7." "Review"
hr
printf '     %-16s %s\n' "Cloud"         "$CLOUD / $LOC"
printf '     %-16s %s\n' "Subscription"  "$SUBNAME"
printf '     %-16s %s\n' "Resource group" "$RG"
printf '     %-16s %s\n' "Image"         "${IMG_ID##*/}"
printf '     %-16s %s\n' "VM size"       "$VMSIZE"
printf '     %-16s %s\n' "How many"      "$COUNT"
printf '     %-16s %s\n' "Admin user"    "$ADMIN"
printf '     %-16s %s\n' "Access from"   "$ACCESS"
hr
confirm "Create now?" || { warn "cancelled; nothing created"; exit 0; }

# ---- 8. create + wait until live ---------------------------------------
step "8." "Creating ${COUNT} vLM VM(s)"
NAMES=""
for n in $(seq 1 "$COUNT"); do
  VN="cloudlens-vlm-$(date +%s 2>/dev/null || echo $n)-$n"
  printf '     %s...%s creating %s\n' "$DIM" "$R" "$VN"
  az_ vm create -g "$RG" -n "$VN" --image "$IMG_ID" --size "$VMSIZE" \
     --admin-username "$ADMIN" --ssh-key-values "$SSHKEY" \
```

```
    --public-ip-sku Standard --nsg-rule NONE --os-disk-size-gb 32 \
    --tags Product=CloudLens-vLM -o none 2>/dev/null || fail "vm create failed for $VN"
  # scope the NSG to the allowed source for 443 and 22
  for pr in "vlm-https 443 300" "vlm-ssh 22 310"; do
    set -- $pr
    az_ network nsg rule create -g "$RG" --nsg-name "${VN}NSG" -n "$1" \
      --priority "$3" --access Allow --protocol Tcp --direction Inbound \
      --source-address-prefixes "$ACCESS" --destination-port-ranges "$2" -o none 2>/dev/null
  done
  NAMES="$NAMES $VN"
done
ok "created:$NAMES"

step "9." "Bringing the vLM online (waits until the web UI answers)"
printf '\n  %s================ Deployment summary ================%s\n\n' "$GRN$B" "$R"
ALL_LIVE=true
for VN in $NAMES; do
  pub=$(az_ vm show -d -g "$RG" -n "$VN" --query publicIps -o tsv 2>/dev/null)
  prv=$(az_ vm show -d -g "$RG" -n "$VN" --query privateIps -o tsv 2>/dev/null)
  printf '  %s%s%s  %swaiting for vLM UI on https://%s%s ' "$B" "$VN" "$R" "$DIM" "$pub" "$R"
  status="still initializing"; scol="$YEL"
  for i in $(seq 1 48); do
    code=$(curl -sk -o /dev/null -w '%{http_code}' --max-time 5 "https://$pub/" 2>/dev/null)
    if [ "$code" = "200" ] || [ "$code" = "302" ]; then
      login=$(curl -sk --max-time 6 -X POST --data-urlencode "userid=admin" --data-urlencode "password=admin"
"https://$pub/rest/license/login" 2>/dev/null)
      case "$login" in *'"success":true'*) status="LIVE - login verified (admin/admin)"; scol="$GRN";; *)
status="LIVE - web UI up"; scol="$GRN";; esac
      break
    fi
    printf '.'; sleep 10
  done
  [ "$scol" = "$YEL" ] && ALL_LIVE=false
  printf '\n'
  printf '     %sStatus %s  %s%s%s\n' "$CYN" "$R" "$scol" "$status" "$R"
  printf '     %svLM UI %s   https://%s\n' "$CYN" "$R" "$pub"
  printf '     %sLogin  %s   admin / admin   %s(change on first login)%s\n' "$CYN" "$R" "$DIM" "$R"
  printf '     %sPrivate%s   %s   %s(point your vPBs here to license)%s\n\n' "$CYN" "$R" "$prv" "$DIM" "$R"
done
if $ALL_LIVE; then ok "All vLM VMs are LIVE and reachable."
else warn "Some VMs are still initializing; the vLM app can take a few minutes."; fi
note "Teardown:  az group delete -n $RG --yes --no-wait   (removes everything in the group)"
printf '\n  %sDone.%s\n\n' "$GRN$B" "$R"
```

Appendix B. Quick Reference

Item	Value
Image source	Keysight software download portal (software.keysight.com) - official vLM image only
Source image (qcow2)	CloudLens-Virtual-License-Manager-1.7.qcow2
Converted disk	CloudLens-vLM-1.7.vhd (fixed VHD, MB-aligned)
Convert command	qemu-img convert -f qcow2 -O vpc -o subformat=fixed,force_size
Interactive wizard	deploy-cloudlens-vlm-azure.sh (build image + create VMs, Commercial or Government)
Image build engine	create-cloudlens-vlm-image-azure.sh (non-interactive, called by the wizard)
Recommended VM size	Standard_D2s_v3 (2 vCPU, 8 GB)

Validated VM	vlm-vm 10.1.2.5 / 20.15.68.17 (CLOUDLENSGWLB-RG, subscription CloudLensPublic)
Login	admin / admin (change on first login) - POST rest/license/login
Commercial region	eastus2
Government cloud	AzureUSGovernment (az cloud set --name AzureUSGovernment)
Government regions	usgovvirginia, usgovtexas, usgovarizona
Government blob endpoint	`.blob.core.usgovcloudapi.net` (portal portal.azure.us)

Prepared by Keysight Sales Engineering. The vLM VM is validated live (vlm-vm, admin login verified); run the same script in a Government subscription to produce the Government image for production vPB licensing.