KEYSIGHT

TECHNOLOGIES

CLOUD DEPLOYMENT RUNBOOK

CloudLens Virtual License Manager as an AWS AMI

Commercial and AWS GovCloud

Build the CloudLens vLM appliance as an Amazon Machine Image and deploy it to support production vPB licensing, in commercial AWS and in GovCloud

Purpose	**Production vLM licensing in AWS**
Source image	**CloudLens vLM 1.7**
Validated	**AMI ami-0a22bc9af29ceab12 (us-east-1)**
Prepared by	**Keysight Sales Engineering**

Table of Contents

1. Executive Summary

This runbook builds the Keysight CloudLens Virtual License Manager (vLM) as an Amazon Machine Image (AMI) and deploys it in AWS, in both the commercial partition and AWS GovCloud. The vLM is the network license server that standalone Virtual Packet Brokers (vPBs) pull their capacity licenses from, so this AMI lets the vLM run natively in AWS to support production vPB workloads.

The build takes the vLM appliance image, converts it to a VM Import friendly format, uploads it to S3, and imports it into an AMI. The process is proven end to end: it was executed live and produced a working, bootable AMI in the commercial account. The same steps, and the same automation script, run unchanged in GovCloud against a GovCloud account.

- **Delivered.** A validated AMI (ami-0a22bc9af29ceab12) built from CloudLens vLM 1.7 in us-east-1.
- **Automated.** A single, partition-aware script (Appendix A) performs the whole build, commercial or GovCloud.
- **GovCloud ready.** GovCloud is a separate partition and account. Section 6 lists exactly what changes.
- **Production intent.** The AMI is EBS-backed, HVM, ENA-enabled, so it launches on current EC2 instance types.

GovCloud in one line

AWS GovCloud is an isolated partition (arn:aws-us-gov) with its own account and credentials. An AMI built in commercial does not appear in GovCloud. You run the same import inside a GovCloud account to produce the GovCloud AMI. Everything else is identical.

2. How It Works

AWS VM Import and Export converts a disk image into an EBS snapshot and registers it as an AMI. The pipeline is the same in every AWS region and partition.

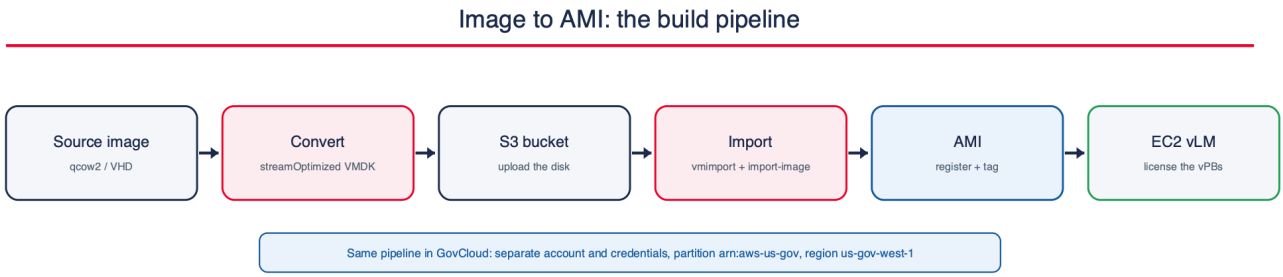

Figure 1. The build pipeline. The same flow runs in commercial AWS and in GovCloud; only the account, credentials, partition ARNs, and region differ.

- VM Import accepts VMDK, VHD, VHDX, OVA, and raw. It does not accept qcow2, so a qcow2 source is converted first.

- A one-time IAM service role named vmimport lets the VM Import service read the disk from S3 and register the AMI.
- import-image runs asynchronously: it converts, boots the image to validate it, prepares the AMI, and completes.
- The result is an EBS-backed AMI you can launch, snapshot, copy across regions, or share to other accounts.

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 for AWS. 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 / .vmdk).

☐ AWS CLI v2 installed and configured for the target account.

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

☐ Credentials for the target account with IAM, EC2, and S3 permissions (create role, create bucket, import image).

☐ VM Import and Export available in the account and region (enabled by default in most accounts).

☐ For GovCloud: a GovCloud account and its own credentials or SSO profile.

One-time vs per-build

The vmimport IAM role is created once per account. The S3 bucket can be reused. The image conversion and the import are per build, or per new vLM version.

4. Automated Path (One Command)

Two scripts ship with this runbook. Both are partition-aware, so they run unchanged in commercial AWS and in GovCloud.

- **deploy-cloudlens-vlm.sh** The premium interactive wizard. It asks a few friendly questions, builds the AMI if you need one, launches the vLM instances, and prints a clean access summary with the login. This is the recommended path.
- **create-cloudlens-vlm-ami.sh** The non-interactive engine for CI or scripted builds. It builds only the AMI. The wizard calls it when you choose to build a new AMI.

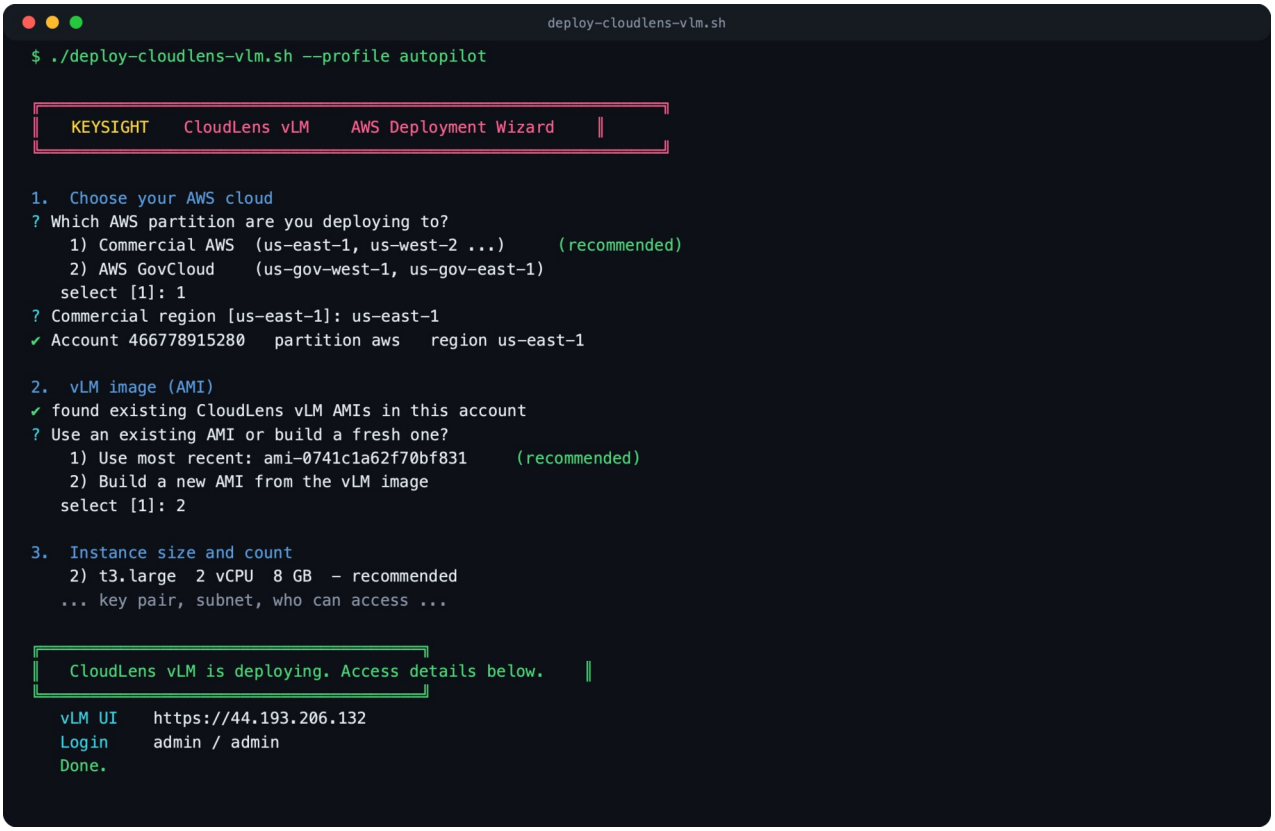

Figure 2. The interactive wizard. It confirms the account and partition, offers to reuse or build the AMI, asks the instance size, count, key pair, subnet, and access, then launches and prints the vLM URL and login.

Run the wizard:

```
./deploy-cloudlens-vlm.sh                                          # full interactive
./deploy-cloudlens-vlm.sh --profile autopilot                      # commercial
./deploy-cloudlens-vlm.sh --profile govcloud --region us-gov-west-1   # GovCloud
```

The wizard walks these questions:

- Commercial AWS or GovCloud, and the region.
- Reuse an existing vLM AMI, or build a fresh one from the image.
- Instance size (t3.medium, t3.large recommended, m5.large, c5.xlarge, or a custom type) and how many.
- SSH key pair (pick an existing one or create a new one).
- Subnet to launch into, and who may reach the vLM (your IP, a CIDR, or anywhere).

It shows a review, launches on your confirmation, then waits until each vLM web UI is live on port 443 and verifies the administrator login, 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.

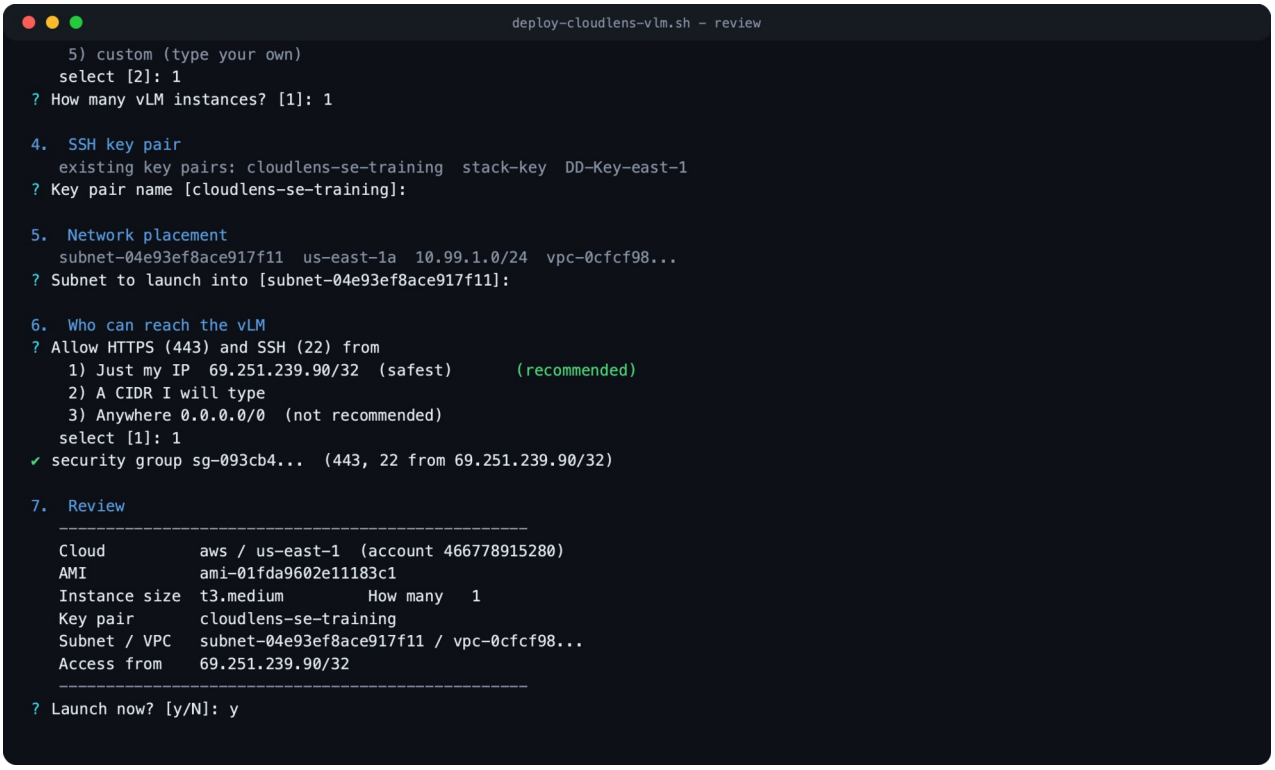

Figure 3. The wizard collects the key pair, subnet, and access scope, then shows a clear review of every choice before it launches. Recommended options are the default in brackets, so pressing Enter accepts them.

Non-interactive AMI build (CI or scripted):

```
./create-cloudlens-vlm-ami.sh --image <image> --region us-east-1 --profile autopilot
./create-cloudlens-vlm-ami.sh --image <image> --region us-gov-west-1 --profile govcloud
```

The engine prints the account, partition, and region it detected, runs to completion, and reports the new AMI 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 AWS CLI is pointed at the target account (add --profile and --region as needed).

5.1 Prepare the image

First download the official CloudLens vLM image from the Keysight software download portal (software.keysight.com). VM Import does not accept qcow2, so convert it to a streamOptimized VMDK (small, fast to upload). If you already have a VHD or VMDK from Keysight, skip the conversion.

```
qemu-img convert -p -f qcow2 -O vmdk -o subformat=streamOptimized \
  CloudLens-Virtual-License-Manager-1.7.qcow2  CloudLens-vLM-1.7.vmdk
```

5.2 Create the S3 bucket

Create a bucket in the target region to stage the disk.

```
aws s3api create-bucket --bucket cloudlens-vlm-import-<ACCOUNT_ID> --region us-east-1
# outside us-east-1 add: --create-bucket-configuration LocationConstraint=<region>
```

5.3 Create the vmimport role

VM Import assumes a service role named vmimport. The trust policy allows the VM Import service; the permissions policy grants read on the bucket and the EC2 import actions. In GovCloud the ARNs use the arn:aws-us-gov partition (see Section 6).

```
# trust-policy.json
{ "Version":"2012-10-17","Statement":[{"Effect":"Allow",
  "Principal":{"Service":"vmie.amazonaws.com"},"Action":"sts:AssumeRole",
  "Condition":{"StringEquals":{"sts:Externalid":"vmimport"}}}]}

aws iam create-role --role-name vmimport \
  --assume-role-policy-document file://trust-policy.json
aws iam put-role-policy --role-name vmimport --policy-name vmimport-vlm \
  --policy-document file://role-policy.json   # S3 get on the bucket + ec2 import
actions
```

5.4 Upload the disk

```
aws s3 cp CloudLens-vLM-1.7.vmdk s3://cloudlens-vlm-import-<ACCOUNT_ID>/
```

5.5 Import and monitor

Start the import and poll it. It moves through converting, updating, booting, preparing, then completed with an ImageId.

```
aws ec2 import-image --description "CloudLens vLM 1.7" \
  --disk-containers '[{"Description":"CloudLens vLM 1.7","Format":"vmdk",
    "UserBucket":{"S3Bucket":"cloudlens-vlm-import-<ACCOUNT_ID>","S3Key":"CloudLens-
vLM-1.7.vmdk"}}]'

aws ec2 describe-import-image-tasks --import-task-ids <import-ami-id> \
  --query 'ImportImageTasks[0].{status:Status,msg:StatusMessage,image:ImageId}'
```

5.6 Tag, launch, verify

```
aws ec2 create-tags --resources <ami-id> \
  --tags Key=Name,Value=CloudLens-vLM-1.7 Key=Product,Value=CloudLens-vLM
aws ec2 run-instances --image-id <ami-id> --instance-type t3.small \
  --key-name <key> --security-group-ids <sg> --subnet-id <subnet>
```

After boot, open the vLM web UI over HTTPS, confirm the License Manager page loads, then point each standalone vPB at the vLM private IP so it checks out its capacity license.

6. GovCloud Specifics

GovCloud is a physically and logically separate AWS partition for US government workloads. The build is identical; only these differ.

Item	Commercial	GovCloud
Partition	arn:aws	arn:aws-us-gov
Regions	us-east-1, us-west-2, ...	us-gov-west-1, us-gov-east-1
Account / credentials	commercial account	separate GovCloud account and credentials
Console	console.aws.amazon.com	console.amazonaws-us-gov.com
vmimport role ARNs	arn:aws:s3:::bucket	arn:aws-us-gov:s3:::bucket
VM Import and Export	available	available

What to do for GovCloud:

- Authenticate the AWS CLI to the GovCloud account (separate access keys or SSO profile).
- Run the automation script with --region us-gov-west-1 and the GovCloud profile. It detects the aws-us-gov partition automatically and writes the role ARNs accordingly.
- Everything else, bucket, upload, import, AMI, is the same. The AMI id is created in the GovCloud partition and is only visible there.

Why we validate in commercial first

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

7. Validated Result

The pipeline was executed live in the commercial demo account and produced a working AMI. VM Import booted the image during conversion, which confirms it is launchable.

Attribute	Value
AMI id	ami-0a22bc9af29ceab12
Name / tags	CloudLens-vLM-1.7 (Product=CloudLens-vLM, Version=1.7)
State	available
Architecture	x86_64, HVM, ENA enabled
Root device	EBS, 16 GiB (snapshot snap-03a4f64679c146a1a)
Region / account	us-east-1 / 466778915280 (commercial)
Source	CloudLens-vLM-1.7.vmdk (converted from CloudLens vLM 1.7 qcow2)

Import stages observed end to end: converting, updating, booting, preparing, completed. Total time was on the order of twenty minutes for a 16 GiB image.

7.1 Live boot verification

An instance was launched from the AMI to confirm it boots into a working vLM, not just a disk that imported.

- **Instance.** A t3.medium launched from ami-0a22bc9af29ceab12, reached the running state, and passed its status checks.
- **vLM UI confirmed.** The License Manager web application 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 against the running instance: admin / admin returns a successful authenticated session, and an authenticated call returned the license host id. Change the default password after first login. Reach the UI at https://<instance-public-ip> from an allowed source.
- **Access control.** Restrict the security group 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 against the commercial account, both parts:

- **Engine.** create-cloudlens-vlm-ami.sh ran unattended and independently produced a second working AMI, ami-0741c1a62f70bf831.
- **Wizard.** deploy-cloudlens-vlm.sh ran the full interactive flow and launched a vLM instance whose web UI and admin login were then verified live (successful authenticated session).

This confirms the same scripts will build and deploy the vLM in a GovCloud account, changing only the account, credentials, and region.

8. Troubleshooting

Symptom	Cause	Fix
import-image fails on disk validation	VM Import cannot read the image or wrong format	Confirm the S3 key, the Format field, and that the disk is VMDK/VHD/OVA/raw, not qcow2
AccessDenied during import	vmimport role missing or wrong bucket ARNs	Recreate the role; ensure the policy lists the exact bucket and, in GovCloud, arn:aws-us-gov
Unsupported OS on import	VM Import rejects a locked-down appliance	Fallback: import-snapshot on the disk, then register-image against the returned snapshot
Bucket create fails outside us-east-1	Missing location constraint	Add --create-bucket-configuration LocationConstraint=<region>
AMI built but will not boot on a type	Older virtualization	The produced AMI is HVM + ENA; use current families (t3, m5, c5, GovCloud equivalents)

Fallback build (register from a snapshot) if VM Import ever refuses the appliance:

```
aws ec2 import-snapshot --disk-container '{"Format":"vmdk",
  "UserBucket":{"S3Bucket":"<bucket>","S3Key":"CloudLens-vLM-1.7.vmdk"}}'
# wait for the snapshot, then:
aws ec2 register-image --name CloudLens-vLM-1.7 --architecture x86_64 \
  --root-device-name /dev/sda1 --virtualization-type hvm --ena-support \
  --block-device-mappings '[{"DeviceName":"/dev/sda1",
    "Ebs":{"SnapshotId":"<snap-id>","VolumeSize":16,"VolumeType":"gp3"}}]'
```

Appendix A. Automation Script

deploy-cloudlens-vlm.sh, the interactive wizard listed below. Keep it beside create-cloudlens-vlm-ami.sh (the engine it calls to build an AMI), make both executable, then run ./deploy-cloudlens-vlm.sh. Partition-aware, so it runs in commercial AWS and in GovCloud.

```
set -uo pipefail

# ---- palette ------------------------------------------------------------
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' "$RED"
  printf '  ╔══════════════════════════════════════════════════════════════════╗\n'
  printf '  ╠                                                                  ╣\n'
  printf '  ╠      %sKEYSIGHT%s   CloudLens vLM   %sAWS Deployment Wizard%s      %s╣\n' "$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(){ # ask VAR "Prompt" "default"
  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(){ # menu VAR "Prompt" default_index  "label1|value1" "label2|value2" ...
  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; }

# ---- args ----------------------------------------------------------------
PROFILE=""; REGION=""; ASSUME_YES=false; IMAGE=""
while [[ $# -gt 0 ]]; do case "$1" in
  --profile) PROFILE="$2"; shift 2;;
  --region)  REGION="$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: $1"; exit 1;;
esac; done
AWSP=(); [[ -n "$PROFILE" ]] && AWSP=(--profile "$PROFILE")
aws_(){ aws "${AWSP[@]}" --region "$REGION" "$@"; }

banner

# ---- 1. cloud + auth -----------------------------------------------------
step "1." "Choose your AWS cloud"
if [[ -z "$REGION" ]]; then
  menu CLOUD "Which AWS partition are you deploying to?" 1 \
    "Commercial AWS  (us-east-1, us-west-2 ...)|commercial" \
    "AWS GovCloud    (us-gov-west-1, us-gov-east-1)|govcloud"
  if [[ "$CLOUD" == govcloud ]]; then
    menu REGION "GovCloud region" 1 "us-gov-west-1|us-gov-west-1" "us-gov-east-1|us-gov-east-1"
  else
    ask REGION "Commercial region" "us-east-1"
  fi
fi
if [[ -z "$PROFILE" ]]; then
  ask PROFILE "AWS CLI profile to use (blank = default/env)" ""
  [[ -n "$PROFILE" ]] && AWSP=(--profile "$PROFILE")
fi
note "checking credentials ..."
CALLER="$(aws_ sts get-caller-identity --output json 2>/dev/null)" || {
  warn "not authenticated for region $REGION."
  note "Commercial SSO:  aws sso login --profile ${PROFILE:-<profile>}"
  note "GovCloud:        configure a GovCloud profile, then re-run with --profile <that>"
  fail "authenticate and re-run"
}
ACCOUNT=$(python3 -c 'import json,sys;print(json.load(sys.stdin)["Account"])' <<<"$CALLER")
ARN=$(python3 -c 'import json,sys;print(json.load(sys.stdin)["Arn"])' <<<"$CALLER")
PARTITION=$(cut -d: -f2 <<<"$ARN")
ok "Account ${B}${ACCOUNT}${R}   partition ${B}${PARTITION}${R}   region ${B}${REGION}${R}"

# ---- 2. AMI: reuse or build ----------------------------------------------
step "2." "vLM image (AMI)"
readlines EXIST < <(aws_ ec2 describe-images --owners self \
   --filters "Name=tag:Product,Values=CloudLens-vLM" \
   --query 'reverse(sort_by(Images,&CreationDate))[].[ImageId,Name]' --output text 2>/dev/null)
AMI=""
if [[ ${#EXIST[@]} -gt 0 && -n "${EXIST[0]}" ]]; then
  ok "found existing CloudLens vLM AMIs in this account:"
  for e in "${EXIST[@]}"; do note "$e"; done
  DEFAMI=$(awk '{print $1}' <<<"${EXIST[0]}")
  menu AMICHOICE "Use an existing AMI or build a fresh one?" 1 \
    "Use most recent: $DEFAMI|$DEFAMI" "Build a new AMI from the vLM image|BUILD"
  AMI="$AMICHOICE"
else
  note "no CloudLens vLM AMI found in this account/region"
  AMI="BUILD"
fi

if [[ "$AMI" == "BUILD" ]]; then
  [[ -z "$IMAGE" ]] && ask IMAGE "Path to the vLM image (.qcow2 / .vhd / .vmdk)" "$HOME/Downloads/CloudLens-
```

```
Virtual-License-Manager-1.7.qcow2"
  [[ -f "$IMAGE" ]] || fail "image not found: $IMAGE"
  step " " "Building the AMI (this runs the full import, about 20 minutes)"
  ENGINE="$(dirname "$0")/create-cloudlens-vlm-ami.sh"
  [[ -x "$ENGINE" ]] || fail "engine not found: $ENGINE (keep create-cloudlens-vlm-ami.sh beside this script)"
  BUILD_ARGS=(--image "$IMAGE" --region "$REGION"); [[ -n "$PROFILE" ]] && BUILD_ARGS+=(--profile "$PROFILE")
  AMI=$("$ENGINE" "${BUILD_ARGS[@]}" | tee /dev/stderr | awk '/AMI ready:/{print $3; exit}')
  [[ "$AMI" == ami-* ]] || fail "AMI build did not return an id"
fi
ok "using AMI ${B}${AMI}${R}"

# ---- 3. instance shape ------------------------------------------------------
step "3." "Instance size and count"
menu ITYPE "Instance size for each vLM" 2 \
  "t3.medium   2 vCPU   4 GB   - minimal|t3.medium" \
  "t3.large    2 vCPU   8 GB   - recommended (matches the appliance)|t3.large" \
  "m5.large    2 vCPU   8 GB   - general purpose|m5.large" \
  "c5.xlarge   4 vCPU   8 GB   - higher throughput|c5.xlarge" \
  "custom (type your own)|custom"
[[ "$ITYPE" == custom ]] && ask ITYPE "Enter the instance type" "t3.large"
ask COUNT "How many vLM instances?" "1"
[[ "$COUNT" =~ ^[0-9]+$ ]] || fail "count must be a number"

# ---- 4. key pair ------------------------------------------------------------
step "4." "SSH key pair"
readlines KEYS < <(aws_ ec2 describe-key-pairs --query 'KeyPairs[].KeyName' --output text 2>/dev/null | tr '\t'
'\n')
if [[ ${#KEYS[@]} -gt 0 && -n "${KEYS[0]}" ]]; then
  note "existing key pairs: ${KEYS[*]}"
  ask KEY "Key pair name" "${KEYS[0]}"
else
  ask KEY "No key pairs found. Enter a name to create one" "cloudlens-vlm-key"
  aws_ ec2 create-key-pair --key-name "$KEY" --query 'KeyMaterial' --output text > "$HOME/.ssh/$KEY.pem"
2>/dev/null \
    && { chmod 600 "$HOME/.ssh/$KEY.pem"; ok "created key, saved to ~/.ssh/$KEY.pem"; }
fi

# ---- 5. network -------------------------------------------------------------
step "5." "Network placement"
readlines SUBS < <(aws_ ec2 describe-subnets \
  --query 'Subnets[?MapPublicIpOnLaunch==`true`].[SubnetId,AvailabilityZone,CidrBlock,VpcId]' --output text
2>/dev/null)
if [[ ${#SUBS[@]} -eq 0 || -z "${SUBS[0]}" ]]; then
  readlines SUBS < <(aws_ ec2 describe-subnets --query 'Subnets[].[SubnetId,AvailabilityZone,CidrBlock,VpcId]' --
output text 2>/dev/null)
fi
[[ ${#SUBS[@]} -gt 0 && -n "${SUBS[0]}" ]] || fail "no subnets found in $REGION"
note "available subnets:"; for s in "${SUBS[@]}"; do note "$s"; done
DEFSUB=$(awk '{print $1}' <<<"${SUBS[0]}"); DEFVPC=$(awk '{print $4}' <<<"${SUBS[0]}")
ask SUBNET "Subnet to launch into" "$DEFSUB"
VPC=$(aws_ ec2 describe-subnets --subnet-ids "$SUBNET" --query 'Subnets[0].VpcId' --output text 2>/dev/null)

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)|0.0.0.0/0"
[[ "$ACCESS" == custom ]] && ask ACCESS "Enter allowed CIDR" "${MYIP}/32"

SG=$(aws_ ec2 create-security-group --group-name "cloudlens-vlm-$RANDOM" \
      --description "CloudLens vLM access" --vpc-id "$VPC" --query GroupId --output text 2>/dev/null)
for p in 443 22; do aws_ ec2 authorize-security-group-ingress --group-id "$SG" --protocol tcp --port $p --cidr
"$ACCESS" >/dev/null 2>&1; done
ok "security group $SG  (443, 22 from $ACCESS)"

# ---- 7. review --------------------------------------------------------------
step "7." "Review"
```

```
hr
printf '     %-16s %s\n' "Cloud"         "$PARTITION / $REGION (account $ACCOUNT)"
printf '     %-16s %s\n' "AMI"           "$AMI"
printf '     %-16s %s\n' "Instance size" "$ITYPE"
printf '     %-16s %s\n' "How many"      "$COUNT"
printf '     %-16s %s\n' "Key pair"      "$KEY"
printf '     %-16s %s\n' "Subnet / VPC"  "$SUBNET / $VPC"
printf '     %-16s %s\n' "Access from"   "$ACCESS"
hr
confirm "Launch now?" || { warn "cancelled (nothing launched); security group $SG left in place"; exit 0; }

# ---- 8. launch -------------------------------------------------------------
step "8." "Launching ${COUNT} vLM instance(s)"
IIDS=$(aws_ ec2 run-instances --image-id "$AMI" --instance-type "$ITYPE" --count "$COUNT" \
  --key-name "$KEY" --security-group-ids "$SG" --subnet-id "$SUBNET" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=cloudlens-vlm},{Key=Product,Value=CloudLens-
vLM}]' \
  --query 'Instances[].InstanceId' --output text 2>&1) || fail "run-instances failed: $IIDS"
ok "launched: $IIDS"
note "waiting for the OS to boot ..."
aws_ ec2 wait instance-running --instance-ids $IIDS 2>/dev/null

# ---- 9. bring the vLM online and verify ------------------------------------
step "9." "Bringing the vLM online (this waits until the web UI answers)"
printf '\n  %s================ Deployment summary ================%s\n\n' "$GRN$B" "$R"
ALL_LIVE=true
for id in $IIDS; do
  pub=$(aws_ ec2 describe-instances --instance-ids "$id" --query 'Reservations[0].Instances[0].PublicIpAddress' --
output text 2>/dev/null)
  prv=$(aws_ ec2 describe-instances --instance-ids "$id" --query 'Reservations[0].Instances[0].PrivateIpAddress'
--output text 2>/dev/null)
  printf '  %s%s%s  %s%s%s ' "$B" "$id" "$R" "$DIM" "waiting for vLM UI on https://$pub" "$R"
  status="still initializing"; scol="$YEL"
  for i in $(seq 1 48); do        # up to ~8 minutes
    code=$(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 instances are LIVE and reachable."
else warn "Some instances are still initializing; the vLM app can take a few minutes. Re-open the URL shortly.";
fi
note "Teardown:  aws ec2 terminate-instances --instance-ids $IIDS ; aws ec2 delete-security-group --group-id $SG"
printf '\n  %sDone.%s\n\n' "$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.vmdk (streamOptimized)
S3 bucket pattern	cloudlens-vlm-import-<ACCOUNT_ID>
Interactive wizard	deploy-cloudlens-vlm.sh (build AMI + launch VMs, commercial or GovCloud)
AMI build engine	create-cloudlens-vlm-ami.sh (non-interactive, called by the wizard)
IAM role	vmimport (trust vmie.amazonaws.com, ExternalId vmimport)
Commercial AMI (validated)	ami-0a22bc9af29ceab12 (us-east-1, 466778915280)
Commercial region	us-east-1
GovCloud regions	us-gov-west-1, us-gov-east-1
GovCloud partition	arn:aws-us-gov
Azure origin (context)	vlm-vm 10.1.2.5, built from the same vLM 1.7 image

Prepared by Keysight Sales Engineering. The commercial AMI is validated; run the same script in a GovCloud account to produce the GovCloud AMI for production vPB licensing.