Azure CLI Practice Lab

What is Azure CLI?

Azure CLI (`az`) is a **cross-platform command-line tool** for managing Azure resources. It lets you create, inspect, update and delete Azure services from your terminal or scripts (Bash, PowerShell, Windows CMD).
Why use it?

- Scriptable and automatable (CI/CD, automation runs).
- Works interactively or in non-interactive scripts (service principals).
- Outputs JSON/table/TSV and supports `--query` (JMESPath) for extracting data.
- Available locally or via **Azure Cloud Shell** in the portal (no install).

Quick prerequisites

- An **Azure subscription** (free/trial or pay-as-you-go).
- Terminal (macOS/Linux/WSL/PowerShell/CMD).
- Install Azure CLI (or use Cloud Shell in the portal).

Quick install & verify (one-liners)

macOS (Homebrew):

```
brew update && brew install azure-cli
```

Debian/Ubuntu:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Windows (winget):

```
winget install --id Microsoft.AzureCLI
```

Verify:

```
az version
```

If you don’t want to install: open https://shell.azure.com or use **Cloud Shell** from portal.azure.com (Bash or PowerShell).

Lab: step-by-step practical exercises (progressive, with use-cases)

Each lab contains exact `az` commands and short explanation. Replace placeholders (<. . .>) with your values. Use the **same resource group** (e.g., `LabRG`) for easier cleanup.

Lab 1 — Basics & Resource Management (use-case: organize resources)

Goal: learn login, subscription, create/list resource groups, query outputs, tagging, and export an ARM template.

1. **Login**

```
az login
```

(If headless: `az login --use-device-code`)

2. **Check subscriptions and set the one you want**

```
az account list -o table
az account set --subscription "Your Subscription Name or ID"
```

3. **Create a resource group**

```
az group create -n LabRG -l eastus
```

4. **List resource groups & show details**

```
az group list -o table
az group show -n LabRG -o json
```

5. **Tag the resource group**

```
az group update -n LabRG --set tags.Environment=Practice Owner=Venkat
```

6. **List resources in the RG**

```
az resource list -g LabRG -o table
```

7. **Export an ARM template for the RG (useful for infra-as-code)**

```
az group export -n LabRG > LabRG-template.json
```

8. **Use `--query` to format results**

```
az resource list -g LabRG --query "[].{Name:name,Type:type}" -o table
```

Practice tasks (Lab 1):

- Create another RG in a different region and compare.
- Use `--query` to list only resources of type `Microsoft.Compute/virtualMachines`.

Lab 2 — Storage + Virtual Machine (use-case: host a small app or store files)

Goal: create storage, upload files, create a Linux VM, SSH into it.

1. **Create a storage account**

Storage account names must be globally unique, lowercase, 3–24 chars.

```
az storage account create -n labstorage$(date +%s%N | cut -c1-6) -g LabRG -
l eastus --sku Standard_LRS
```

2. **Get a storage account key (to authenticate storage CLI operations)**

```
STORAGE_NAME=$(az storage account list -g LabRG --query "[0].name" -o tsv)
STORAGE_KEY=$(az storage account keys list -g LabRG -n $STORAGE_NAME --
query "[0].value" -o tsv)
```

3. **Create a container and upload a file**

```
az storage container create --name mycontainer --account-name $STORAGE_NAME
--account-key $STORAGE_KEY
echo "Hello from Azure CLI lab" > hello.txt
az storage blob upload --account-name $STORAGE_NAME --account-key
$STORAGE_KEY --container-name mycontainer --file hello.txt --name hello.txt
```

4. **Create an Ubuntu VM (auto-generates SSH keys if not present)**

```
az vm create -g LabRG -n LabVM --image UbuntuLTS --admin-username azureuser
--generate-ssh-keys --size Standard_B1s
```

5. **Open SSH port**

```
az vm open-port --port 22 --resource-group LabRG --name LabVM
```

6. **Get public IP and SSH**

```
PUBLIC_IP=$(az vm show -d -g LabRG -n LabVM --query publicIps -o tsv)
ssh azureuser@$PUBLIC_IP
```

7. **(Optional) Upload files from your machine to blob and download inside VM via curl/wget if you make them public or use SAS**
   To find the static website endpoint or blob URL:

```
az storage blob url --container-name mycontainer --name hello.txt --
account-name $STORAGE_NAME
```

Practice tasks (Lab 2):

- Configure a **static website** on the storage account:

```
az storage blob service-properties update -n $STORAGE_NAME --static-website
--index-document index.html --404-document 404.html --account-key
$STORAGE_KEY
# Upload website files to $web container
az storage blob upload-batch -s ./site -d '$web' --account-name
$STORAGE_NAME --account-key $STORAGE_KEY
az storage account show -n $STORAGE_NAME -g LabRG --query
"primaryEndpoints.web" -o tsv
```

- Create a second data disk, attach to VM, format & mount it (practice disk operations).

Lab 3 — App Service + Deployment + Database (use-case: deploy a simple web app connected to a DB)

Goal: Create App Service plan + Web App, deploy a zip app, and create Azure SQL DB and firewall rule.

1. **Create App Service plan (Linux)**

```
az appservice plan create -g LabRG -n LabPlan --is-linux --sku B1
```

2. **Create a Web App (node example)**

App name must be globally unique.

```
az webapp create -g LabRG -p LabPlan -n labwebapp$(date +%s%N | cut -c1-6)
--runtime "NODE|18-lts"
```

3. **Deploy your app (zip deploy)**
   Prepare your app folder locally, then:

```
zip -r app.zip .
az webapp deployment source config-zip --resource-group LabRG --name
<YourAppName> --src app.zip
```

4. **Tail logs**

```
az webapp log tail -g LabRG -n <YourAppName>
```

5. **Create Azure SQL server & DB**

```
az sql server create -g LabRG -n labsqlserver$(date +%s%N | cut -c1-6) -l
eastus -u sqladmin -p 'StrongPassw0rd!'
```

```
az sql db create -g LabRG -s <server-name> -n LabDB --service-objective S0
```

6. **Allow your client IP to connect to the SQL server**

```
MYIP=$(curl -s ifconfig.me)
az sql server firewall-rule create -g LabRG -s <server-name> -n AllowMyIP -
-start-ip-address $MYIP --end-ip-address $MYIP
```

7. **Connection string (example)**

```
Server=tcp:<server-name>.database.windows.net,1433;Initial
Catalog=LabDB;User ID=sqladmin;Password=StrongPassw0rd!;Encrypt=True;
```

Practice tasks (Lab 3):

- Configure connection string in App Service settings and test DB connection from your app.
- Scale App Service plan up/down and observe billing implications.

Lab 4 — Automation, RBAC & Monitoring (use-case: CI/CD + secure automation)

Goal: Create a service principal for automation, assign role, view activity logs.

1. **Create a service principal with Contributor rights to the resource group (use for CI/CD)**

```
az ad sp create-for-rbac --name "http://LabAutomation$(date +%s%N | cut -
c1-6)" --role Contributor --scopes /subscriptions/$(az account show --query
id -o tsv)/resourceGroups/LabRG
```

Note returned values: `appId`, `password`, `tenant`. Use them in your pipeline to authenticate:

```
az login --service-principal -u <appId> -p <password> --tenant <tenant>
```

2. **Create a Reader role assignment for a user on the RG**

```
az role assignment create --assignee user@domain.com --role "Reader" --
scope /subscriptions/$(az account show --query id -o
tsv)/resourceGroups/LabRG
```

3. **View recent activity logs**

```
az monitor activity-log list --resource-group LabRG --max-events 20 -o
table
```

4. **Create Log Analytics workspace (for logs/metrics collection)**

```
az monitor log-analytics workspace create -g LabRG -n LabWorkspace
```

Practice tasks (Lab 4):

- Use service principal credentials in a sample GitHub Action to deploy an ARM template or run `az` commands.
- Assign a Managed Identity to your Web App and use it to access Key Vault secrets.

Clean up (very important to avoid charges)

When you're finished, delete the resource group that contains everything:

```
az group delete -n LabRG --yes --no-wait
# or without --no-wait if you want the CLI to wait for completion
```

(If you used multiple RGs, delete them similarly.)

Handy `az` shortcuts & tips (cheat-sheet)

- `az login` — sign in.
- `az account set --subscription <id|name>` — choose subscription.
- `az group create -n <name> -l <location>` — create resource group.
- `az resource list -g <rg>` — list resources in RG.
- `az vm create ...` — create vm.
- `az storage blob upload ...` — upload file to storage.
- `az webapp deployment source config-zip ...` — deploy zip to webapp.
- `az role assignment create ...` — RBAC assign.
- `az ad sp create-for-rbac ...` — create service principal (for automation).
- `--output table` or `-o tsv` — human-readable outputs; `--query` for filtering.

Formatting & parsing

- Use `--query` (JMESPath) to extract fields; combine with `-o tsv` to get easy scriptable values.

Suggested progressive exercises (with increasing difficulty)

1. Create `LabRG`, deploy a VM, SSH in, serve a static page from the VM.
2. Host a static website on Storage static website feature and point a custom domain (optional).
3. Deploy a Node/.NET sample app to App Service and connect to Azure SQL.

4. Create a service principal and deploy the same app from a local script using the service principal.
5. Create an ARM template from your deployed resources, then re-deploy from the template into a new RG.
6. Automate the whole flow in a Bash script: create RG → storage → VM → web app → SQL → output connection strings.