

# LAB 1 — VPC: Create a Basic Network and Public Subnet

**Time:** 30–35 minutes

**Learning Objective:** Create a custom VPC, subnet, Internet Gateway, and route table for internet access.

## Prerequisites

- AWS account with MFA
- No existing VPC resources required
- Students logged into AWS Console

## Steps

1. Open **VPC** from AWS search bar.
2. Go to **Your VPCs → Create VPC**
  - Resources to create: **VPC only**
  - Name: student-vpc
  - IPv4 CIDR: 10.0.0.0/16
  - Create
3. Left menu → **Subnets → Create subnet**
  - VPC: student-vpc
  - Subnet name: public-a
  - AZ: ap-south-1a
  - CIDR: 10.0.1.0/24
  - Create
4. Left menu → **Internet Gateways → Create**
  - Name: student-igw
  - Create → Attach to VPC → student-vpc
5. Left menu → **Route tables → Create**
  - Name: public-rt
  - VPC: student-vpc
  - Create
6. Open route table → **Routes → Edit routes**
  - Add: 0.0.0.0/0 → student-igw
7. Same page → **Subnet associations → Edit**
  - Check public-a subnet
  - Save

## ✅ Student Checklist

- ✓ VPC exists with CIDR 10.0.0.0/16
- ✓ Subnet exists with CIDR 10.0.1.0/24
- ✓ IGW is attached
- ✓ Route table has default route 0.0.0.0/0
- ✓ Subnet associated with public route table

## ! Troubleshooting

- If subnet shows "No route to internet": Check route table association.

- If IGW not selectable: It was not attached to the VPC.
- 

## ✓ LAB 2 — IAM: Create a User + Access Keys + CLI Configuration

**Time:** 25–30 minutes

**Learning Objective:** Create IAM user, generate keys, configure AWS CLI, test identity.

### Prerequisites

- AWS Console access
- Ability to install AWS CLI (Windows/Mac/Linux)

### Steps

1. Open IAM
2. **Users → Create user**
  - Name: student01
  - Next
3. **Permissions → Attach policies directly**
  - Select: AmazonS3ReadOnlyAccess
  - Next → Create user
4. Open the user → **Security credentials**
  - Access keys → Create access key
  - Intended use: **CLI**
  - Download .csv

### Configure CLI

```
aws configure
# Enter Access Key
# Enter Secret
# Region: ap-south-1
# Output: json
```

### Test Commands

```
aws sts get-caller-identity
aws s3 ls
```

### ✓ Student Checklist

- ✓ User created
- ✓ Policy attached
- ✓ Keys downloaded safely

- ✓ `aws sts get-caller-identity` shows IAM username
- ✓ No permission denied errors on S3 list

### ! Troubleshooting

- "Unknown endpoint": Region not set correctly in `aws configure`
  - "Access Denied": Policy not attached or user not chosen
- 

## ✓ LAB 3 — IAM Policies: Create Group & Attach Policy

**Time:** 15–20 minutes

**Learning Objective:** Learn how IAM groups control user permissions.

### Prerequisites

- Completed Lab 2

### Steps

1. Open **IAM** → **User groups** → **Create group**
  - Name: `readonly-group`
2. Attach policy → search `ReadOnlyAccess`
3. Create group
4. Add user:
  - **IAM** → **Users** → `student01` → **Permissions** → **Add user to group** → `readonly-group`

### ✓ Student Checklist

- ✓ Group created
- ✓ Policy attached
- ✓ `student01` added to group

### ! Troubleshooting

- If student cannot access a service: Make sure group permissions include it
- 

## ✓ LAB 4 — Billing and Cost Control: Create Budget Alert

**Time:** 15 minutes

**Learning Objective:** Learn to control AWS cost with automated email alerts.

### Prerequisites

- Billing access enabled for account

### Steps

1. AWS search → **Billing**
2. Left menu → **Budgets** → **Create budget**
3. Budget type: **Cost budget**
4. Amount: 5 USD monthly

5. Alert threshold: 80% actual cost
6. Enter student email
7. Create budget

### ✅ Student Checklist

- ✓ Budget created
- ✓ Email alert configured
- ✓ Shows ACTIVE status

### ! Troubleshooting

- Students don't get the email? Check spam folder or enter correct email
- 

## ✅ LAB 5 — Lambda: Create & Test a Simple Function

Time: 25 minutes

Learning Objective: Run code without servers.

### Prerequisites

- None

### Steps

1. AWS search → **Lambda**
2. Create function → Author from scratch
  - Name: hello-student
  - Runtime: Python 3.x
  - Permissions: Create a new role with basic execution
3. Replace code:

```
def lambda_handler(event, context): return "Hello from Lambda"
```

4. Deploy → Test
5. Create test event → run again

### ✅ Student Checklist

- ✓ Function deploys without error
- ✓ Test returns "Hello from Lambda"

### ! Troubleshooting

- If function fails: Check "Monitor" → CloudWatch logs
- 

## ✅ LAB 6 — CloudWatch Logs: View Lambda Logs

Time: 10–15 minutes

Learning Objective: Understand where Lambda writes logs.

## Prerequisites

- Lab 5 complete

## Steps

1. Lambda → hello-student
2. Top menu: **Monitor** → **View logs in CloudWatch**
3. Open latest log stream
4. Find:
  - START
  - Report
  - "Hello from Lambda"

## ✅ Student Checklist

- ✓ Located log group
- ✓ Opened a log stream
- ✓ Found execution entries

## ! Troubleshooting

- If no logs: Run the Lambda at least once
- 

# ✅ LAB 7 — API Gateway: Create REST API Triggering Lambda

**Time:** 30–35 minutes

**Learning Objective:** Build a public HTTP URL calling Lambda.

## Prerequisites

- Lab 5 completed

## Steps

1. AWS search → **API Gateway**
2. Create API → **REST API**
3. Resources → **Actions** → **Create Resource**
  - Name: hello
  - Path: /hello
4. Create Method → GET
  - Integration type: Lambda
  - Function: hello-student
5. **Deploy API**
  - Stage: prod
6. Copy the invoke URL and test in browser or CLI:

```
curl https://<api-id>.execute-api.<region>.amazonaws.com/prod/hello
```

## ✅ Student Checklist

- ✓ REST API created
- ✓ GET /hello deployed
- ✓ URL returns Lambda output

## ! Troubleshooting

- If URL says "Forbidden": Missing deployment
  - If "Internal server error": Check Lambda logs
- 

## ✅ LAB 8 — S3: Create Bucket & Upload Objects

**Time:** 20 minutes

**Learning Objective:** Store and view files in a bucket.

### Prerequisites

- AWS CLI configured (from Lab 2)

### Steps

1. AWS search → **S3**
2. Create bucket
  - Name: student-bucket-<random>
  - Region: ap-south-1
3. Upload → choose a file → Upload

### Optional CLI:

```
aws s3 ls s3://student-bucket-123
```

## ✅ Student Checklist

- ✓ Bucket created
- ✓ File uploaded
- ✓ File visible

## ! Troubleshooting

- If upload fails: Bucket name not unique → add random numbers
- 

## ✅ LAB 9 — EC2: Launch and Connect to Instance

**Time:** 30–40 minutes

**Learning Objective:** Start a Linux EC2 and connect via SSH.

### Prerequisites

- VPC and subnet from Lab 1

## Steps

1. Open **EC2**
2. Launch instance:
  - Amazon Linux 2023
  - t2.micro
  - VPC: `student-vpc`
  - Subnet: `public-a`
  - Auto-assign Public IP: **Enable**
3. Create key pair → download `.pem`
4. Add security group rule:
  - SSH port 22 from **My IP**
5. Launch
6. Connect:

```
ssh -i mykey.pem ec2-user@<Public-IP>
```

## ✅ Student Checklist

- ✓ Instance running
- ✓ Public IP assigned
- ✓ SSH connection successful

## ! Troubleshooting

- Connection timeout: Security group missing SSH rule
- Permission denied: Wrong key permissions → run `chmod 400 mykey.pem`

---

## ✅ LAB 10 — CloudFormation: Deploy a Simple Stack

**Time:** 25 minutes

**Learning Objective:** Deploy infrastructure as code.

### Prerequisites

- None

## Steps

1. Open **CloudFormation**
2. Create stack → With new resources
3. Upload this template:

```
AWSTemplateFormatVersion: 2010-09-09 Description: Create an S3 Bucket
Resources: StudentCFBucket: Type: AWS::S3::Bucket Properties: BucketName:
student-cf-bucket-12345
```

4. Next → Next → Create stack

5. Wait for **CREATE\_COMPLETE**
6. Check S3 → bucket should exist

### **Student Checklist**

- ✓ Stack created
- ✓ Bucket appears in S3
- ✓ CloudFormation shows CREATE\_COMPLETE

### **Troubleshooting**

- If CREATE\_FAILED: Bucket name not unique → edit name and redeploy