1. AWS EC2 and AWS CLI on Ubuntu

Prerequisites: Ubuntu system with network access, an AWS account, and basic knowledge of the terminal. AWS offers a *Free Tier* for new accounts (12 months of limited free usage) bootcamp.wiki. Sign up at aws.amazon.com with an email and payment method.

1.1 Create an AWS Account

Go to the AWS website and choose *"Create an AWS Account"*. Follow the signup steps (email, password, contact details, and credit card info). Confirm your account via the verification email. Once logged in, avoid using the root user for tasks; create an IAM user with *Programmatic access* and attach the **AdministratorAccess** policy. This IAM user will have an **Access Key ID** and **Secret Access Key** for CLI access. (Store these safely; you'll use them to configure the CLI.)

1.2 Install AWS CLI on Ubuntu

Update package lists and install Python pip, since the AWS CLI is a Python package:

bash

CopyEdit

sudo apt-get update

sudo apt-get install -y python3-pip

Install the AWS CLI using pip (this installs AWS CLI v1):

bash

CopyEdit

sudo pip3 install awscli

This makes the aws command available system-widegeeksforgeeks.org.
Verify the install: Check the version by running:

bash

CopyEdit

aws --version

You should see output like aws-cli/1.x.x Python/3.x.x .... If you see an error, ensure Python 3 and pip are installed correctly.

1.3 Configure AWS CLI with Credentials

Run aws configure and enter your IAM user's credentials:

bash

CopyEdit

aws configure

You'll be prompted for:

- **AWS Access Key ID:** (enter your IAM user's key)
- **AWS Secret Access Key:** (enter your IAM user's secret)
- **Default region name:** e.g. us-east-1 (or your preferred region)
- **Default output format:** json (or text, yaml)

This creates ~/.aws/credentials and ~/.aws/config with your settings. You can verify these files exist and contain your keys. Troubleshoot "InvalidClientTokenId" or similar by re-running aws configure and ensuring the keys are correct.

1.4 Launch and Manage EC2 Instances via CLI

Before launching, create or identify a key pair and a security group. You can do this in the AWS Console or with CLI commands (aws ec2 create-key-pair, etc.). Also decide on an AMI ID (e.g. latest Ubuntu AMI in your region – use aws ec2 describe-images or find one in the AWS Console).

Launch an instance (example with placeholder IDs):

bash

CopyEdit

```
aws ec2 run-instances \
  --image-id ami-xxxxxxxxxxxxxxx \
  --instance-type t2.micro \
  --count 1 \
  --key-name MyKeyPair \
  --security-group-ids sg-xxxxxxxx \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=MyInstance}]'
```

This starts a new EC2 instance (which will appear in **pending** state, then **running**) docs.aws.amazon.com. The command's output shows details including the new InstanceId.

1.4.1 List Instances

Use the describe-instances command to see your instances:

bash

CopyEdit

```
aws ec2 describe-instances
```

This returns JSON with all instance info. To filter by your tag or instance type, you can add options. For example, to list only the IDs of your *t2.micro* instances:

bash

CopyEdit

```
aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro" \
  --query "Reservations[].Instances[].InstanceId"
```

This outputs something like ["i-0123456789abcdef0"]docs.aws.amazon.com.

1.4.2 Stop, Start, Terminate Instances

- To stop (power off) an instance without deleting it:

bash

CopyEdit

```
aws ec2 stop-instances --instance-ids i-0123456789abcdef0
```

- To start a stopped instance:

bash

CopyEdit

```
aws ec2 start-instances --instance-ids i-0123456789abcdef0
```

- To terminate (delete) an instance:

bash

CopyEdit

```
aws ec2 terminate-instances --instance-ids i-0123456789abcdef0
```

Terminating an instance deletes it permanently; once terminated, you stop incurring charges for itdocs.aws.amazon.com. You will see the instance state transition to shutting-down and then terminateddocs.aws.amazon.com.

1.4.3 Verify Operations

After launching, use describe-instances to confirm the instance is in **running** state and note its public DNS or IP. You can SSH into it (if it's a Linux instance) using the key pair:

bash

CopyEdit

```
ssh -i ~/path/MyKeyPair.pem ubuntu@ec2-3-15-234-123.compute-1.amazonaws.com
```

Replace ubuntu@... with the appropriate user/hostname (e.g. ec2-user@... for Amazon Linux). If SSH fails, check Security Group rules (ensure inbound SSH (port 22) is allowed) and that the key file permissions are secure (chmod 400).

1.5 Troubleshooting Tips

- **AWS CLI not found:** If aws --version fails, ensure /usr/local/bin (pip's default install path) is in your PATH. Try installing AWS CLI v2 using the official installer (see AWS docs) if needed.

- **Credentials errors:** If AWS commands complain about invalid credentials, re-run aws configure and double-check the keys and region. Make sure you use IAM user credentials (root user is discouraged)docs.aws.amazon.com.
- **Instance launch errors:** Common issues are invalid AMI ID, missing KeyPair, or insufficient permissions. Check the AWS error message; ensure your IAM user can create EC2 instances, and your command's syntax is correct.

2. Google App Engine on Ubuntu

Prerequisites: Ubuntu with internet, a Google account, and enabling billing on a Google Cloud project. Create a Google Cloud project via console.cloud.google.com and note the *Project ID*. Install curl if not present (sudo apt-get install -y curl).

2.1 Install Google Cloud SDK (gcloud)

Google's SDK provides the gcloud CLI. On Ubuntu, the recommended way is via the Google Cloud apt repository.

1. **Import Google Cloud public key:**

bash

CopyEdit

curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /usr/share/keyrings/cloud.google.gpg

2. **Add the Cloud SDK apt repository:**

bash

CopyEdit

echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] https://packages.cloud.google.com/apt cloud-sdk main" \

| sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list

3. **Update and install:**

bash

CopyEdit

sudo apt-get update && sudo apt-get install -y google-cloud-cli

This installs the Google Cloud CLI (gcloud)cloud.google.com.

Verify the install: Run gcloud --version. You should see version info for gcloud, bq, and gsutil. If this fails, ensure you followed the key import and repository steps exactly.

Note: The default install **does not include App Engine extensions** (e.g. Python/Java support for App Engine)cloud.google.com. You can add them if needed (e.g. gcloud components install app-engine-python).

2.2 Initialize and Authenticate gcloud

Run:

bash

CopyEdit

gcloud init

This launches a browser prompt to log in to your Google account, choose your project, and set a default region. Follow the instructions to complete authentication. (Alternatively, use gcloud auth login if preferred.)

If you need App Engine (standard environment) for Python, ensure the App Engine API is enabled in your project (via the Cloud Console API Library).

2.3 Create and Deploy a "Hello World" App

App Engine applications require an app.yaml file and your application code. For example, a minimal **Python 3** web app using Flask:

- **main.py**:

python

CopyEdit

from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello():

  return "Hello, World!"

- **app.yaml**: (specifies runtime)

yaml

CopyEdit

runtime: python39

entrypoint: gunicorn -b :$PORT main:app

Create these files in a directory (say hello_app/).

Deploy the app:

bash

CopyEdit

gcloud app deploy

You may be prompted to confirm the project and choose a region for App Engine. This uploads the code and starts serving your app. The console will show progress and give the public URL

(usually https://<PROJECT_ID>.uc.r.appspot.com) once deployedcloud.google.com cloud.google.com.

2.4 Verify the App

After gcloud app deploy completes, you can open the app in your browser:

bash

CopyEdit

gcloud app browse

This opens the deployed URL. Alternatively, use the URL printed during deploy (or find it in the Cloud Console). You should see **"Hello, World!"**.

Troubleshooting:

- If deployment fails, check that app.yaml is correctly formatted and your files are in the right directory. Common errors include indent issues in YAML or missing files.
- Ensure you initialized gcloud with the correct project. You can verify with gcloud config list.
- If you omitted the App Engine component, you might see a message to install it. Run, for example: gcloud components install app-engine-python or the appropriate component for your language.
- Check logs for errors: in the Console, navigate to **App Engine > Versions** and view logs or errors for your service.

2.5 Troubleshooting Tips

- If gcloud commands say "command not found", make sure the SDK's bin directory is in your PATH. You may need to restart the terminal after installation.
- If gcloud init hangs or doesn't open a browser, ensure you have a browser available; you can also authorize by copying the URL into a browser manually.
- If you see "AUTHENTICATION_REQUIRED" or similar, run gcloud auth login again or check network connectivity.
- For Python apps, make sure any required libraries are listed (e.g., add Flask to a requirements.txt if needed).

3. Salesforce: Apex Calculator App

Prerequisites: A Salesforce Developer Edition account (free) and Ubuntu with a web browser.

3.1 Sign Up for a Salesforce Developer Account

Go to developer.salesforce.com/signup and register for a free Developer Edition org. You'll receive an email to verify and set a password. This gives you a personal Salesforce environment to develop and test.

3.2 Install Salesforce CLI (sf or sfdx) on Ubuntu

Salesforce provides a modern CLI (called **sf**) and the older **sfdx** commands. We'll use the Salesforce CLI (newer "sf")developer.salesforce.com. You can install via a tarball or npm:

Method A: Using the tarball (recommended for no-admin installs) resources.docs.salesforce.comresources.docs.salesforce.com:

1. **Download CLI** (for Linux x64):

bash

CopyEdit

wget https://developer.salesforce.com/media/salesforce-cli/sf/channels/stable/sf-linux-x64.tar.xz

2. **Create install dir and unpack:**

bash

CopyEdit

mkdir -p ~/cli/sf

tar xJf sf-linux-x64.tar.xz -C ~/cli/sf --strip-components 1

3. **Update PATH:**

bash

CopyEdit

export PATH=~/cli/sf/bin:$PATH

Add that line to ~/.bashrc or ~/.zshrc to make it permanent. Now running sf --version should show the CLI version.

Method B: Using npm:

If you have Node.js, you can run:

bash

CopyEdit

npm install @salesforce/cli --global

This installs the sf (and sfdx) commands globally. Verify with sf --version. (If you see sfdx still working, note that newer installs alias sfdx to sf.)

Verify installation:

bash

CopyEdit

sf --version

should output something like @salesforce/cli 2.x.x.

If you have an old sfdx-cli installed, uninstall it first (npm uninstall sfdx-cli --global) resources.docs.salesforce.com to avoid conflicts.

3.3 Create a Salesforce DX Project and Authenticate

Open a terminal and create a new SFDX project:

bash

CopyEdit

sf project generate --name CalculatorApp --template hello-world

cd CalculatorApp

(or use sfdx force:project:create -n CalculatorApp for older sfdx CLI). This sets up a directory with manifest and main folders.

Authenticate to your Developer org with the CLI:

bash

CopyEdit

sf login org --web --setalias MyDevOrg --setdefaultusername

This opens a browser to log in to your Salesforce account and authorizes the CLI. After success, the alias MyDevOrg will refer to your dev org. You can verify by running sf org list (shows the logged-in orgs) or sf org display --targetusername MyDevOrg.

3.4 Build and Deploy the Calculator App

In Salesforce, "applications" often use Apex classes and Lightning components. For simplicity, we'll create an **Apex class** that performs basic arithmetic (e.g. addition).

3.4.1 Create Apex Class

Use the CLI to generate a class file:

bash

CopyEdit

sf force:apex:class:create -n Calculator

This creates Calculator.cls in force-app/main/default/classes/. Edit it in a text editor to add methods, for example:

java

CopyEdit

```
public class Calculator {
  // Adds two integers
  public Integer add(Integer a, Integer b) {
    return a + b;
```

```
    }
    // Subtracts b from a
    public Integer subtract(Integer a, Integer b) {
        return a - b;
    }
}
```

Save the file. (You could also create test methods in CalculatorTest.cls similarly.)

3.4.2 Push Source to Org

Deploy the new Apex class to your org:

bash

CopyEdit

```
sf force:source:push --targetusername MyDevOrg
```

This sends all local metadata to the scratch or target org. (For non-scratch, use force:source:deploy -p force-app/main/default/classes/Calculator.cls.) You should see success if deployment succeeds.

Alternatively, you can use sfdx force:source:deploy -p force-app/ to push the entire project.

3.4.3 Verify in Salesforce

Log in to your Developer org via browser. Go to **Setup → Apex Classes** and check that **Calculator** appears. You can open the Developer Console to run anonymous Apex for testing, e.g.:

java

CopyEdit

```
System.debug( Calculator.add(5,7) );
```

Check the debug log to see 12. Or write a simple Lightning or Visualforce UI to call your class (beyond this guide's scope). The key is that the class is present and executable in your org.

3.5 Troubleshooting Salesforce CLI

- If sf commands return errors about login, ensure pop-up blocking is off (the CLI opens a web login). Try adding --instance-url https://login.salesforce.com explicitly if needed.
- If you get authentication errors on Linux, try sf login org --web -d with different flags, or check that your default browser supports OAuth. (Some Linux systems have issues with xdg-open.)
- If force:source:push fails, read the error message. Common issues: missing namespace or permission (ensure your user has API access) or conflicts with existing metadata.

- To list available commands, run sf commands or sfdx force --help developer.salesforce.com.

Note: Salesforce CLI simplifies many tasks: "It aggregates all the tools you need to develop with and perform commands against your Salesforce org"developer.salesforce.com.

4. Custom Salesforce Cloud App (Mini Project)

In a **Task Manager** app, you might create custom objects, UI, and logic. Below is an example workflow using Salesforce Cloud tools:

4.1 Plan Your App (Objects & Features)

- **Requirements:** e.g., Users can create, view, and mark tasks. Tasks have fields like *Subject*, *Due Date*, *Status*.
- **Objects:** Use the built-in **Task** object or create a **Custom Object** (e.g., Task__c) with custom fields (checkbox, date, etc.).
- **UI:** Use Lightning App Builder to create an app and Lightning Record Pages or a Lightning Web Component for listing tasks. No coding needed for basic page layouts.
- **Logic:** Decide if you need Apex triggers or controllers. For example, auto-populating a field or sending reminders on Due Date.

4.2 Develop with Salesforce Tools

4.2.1 Declarative Setup

Login to your org and go to **Setup**:

- **Objects:** If you need a custom object, go to *Object Manager → Create → Custom Object*. Add fields (Subject, Due Date, Completed).
- **Lightning App:** Under *App Manager*, create a new Lightning App (e.g. "Task Manager"). Add tabs or items (like the object's list view).
- **Page Layouts:** Ensure the fields appear on the object's layout so users can edit them.

No CLI needed for these steps, but you could define custom object metadata and use sf force:source:retrieve/push to script them in source.

4.2.2 Project and Metadata via CLI (optional)

To track configuration as code, use your SFDX project:

bash

CopyEdit

cd ~/projects/CalculatorApp

sf project generate manifest

Edit manifest/package.xml to include your custom object (or use sfdx force:source:retrieve -m CustomObject:Task__c). This lets you pull object definitions into your local project.

4.3 Write Apex Logic (if needed)

Example: Add an Apex trigger that sets **Status = 'Overdue'** if the Due Date passes.

Create a trigger or Apex class (in force-app/main/default/classes/) such as:

java

CopyEdit

```
trigger TaskTrigger on Task__c (before insert, before update) {
  for (Task__c t : Trigger.new) {
    if (t.Due_Date__c != null && t.Due_Date__c < Date.today()) {
      t.Status__c = 'Overdue';
    }
  }
}
```

Save this in your project and push to the org (sf force:source:push). Now any task with a past due date will be marked as "Overdue" automatically.

You could also create a Lightning Web Component (LWC) for a custom UI list or form, but even simple things can be done with standard list views.

4.4 Deploy and Test

- **Push Source:** Use the CLI (sf force:source:push) to deploy changes (Apex classes, triggers, and any metadata) to your org.
- **Test Functionality:** In Salesforce UI, create a new Task record. Check that fields appear and that any automation (like the trigger) works.
- **Debug:** If an Apex error occurs, check *Debug Logs* in Setup. Ensure your Apex syntax is correct. Salesforce will report errors like "field not found" if you spelled a field/API name incorrectly.

4.5 Troubleshooting & Tips

- If you miss a field or object, deployment may fail. Use sf force:source:deploy -p force-app/main/default/objects/ to target specific metadata.
- Remember to run tests if you have Apex tests: sf force:apex:test:run --resultformat human. All tests must pass before deploying to some orgs.
- Use **Salesforce CLI commands** for common tasks: e.g. sf force:org:open to open the org in a browser; sf force:data:record:create to programmatically create sample records for testing; sf force:apex:execute to run anonymous Apex.

By following these steps, you set up cloud services and Salesforce development on Ubuntu. Each section's commands and tools are verified as described, and errors can often be resolved by checking credentials, environment variables (like PATH), and appropriate permissions. Use official docs and CLI help commands (--help) for further details as you expand these projects developer.salesforce.comgeeksforgeeks.org.

Sources: AWS and Google Cloud official docs for CLI installation and usagegeeksforgeeks.org cloud.google.comcloud.google.comdeveloper.salesforce.com; Salesforce documentation on CLI and Apex basicsresources.docs.salesforce.comsalesforce.stackexchange.com.

Sources