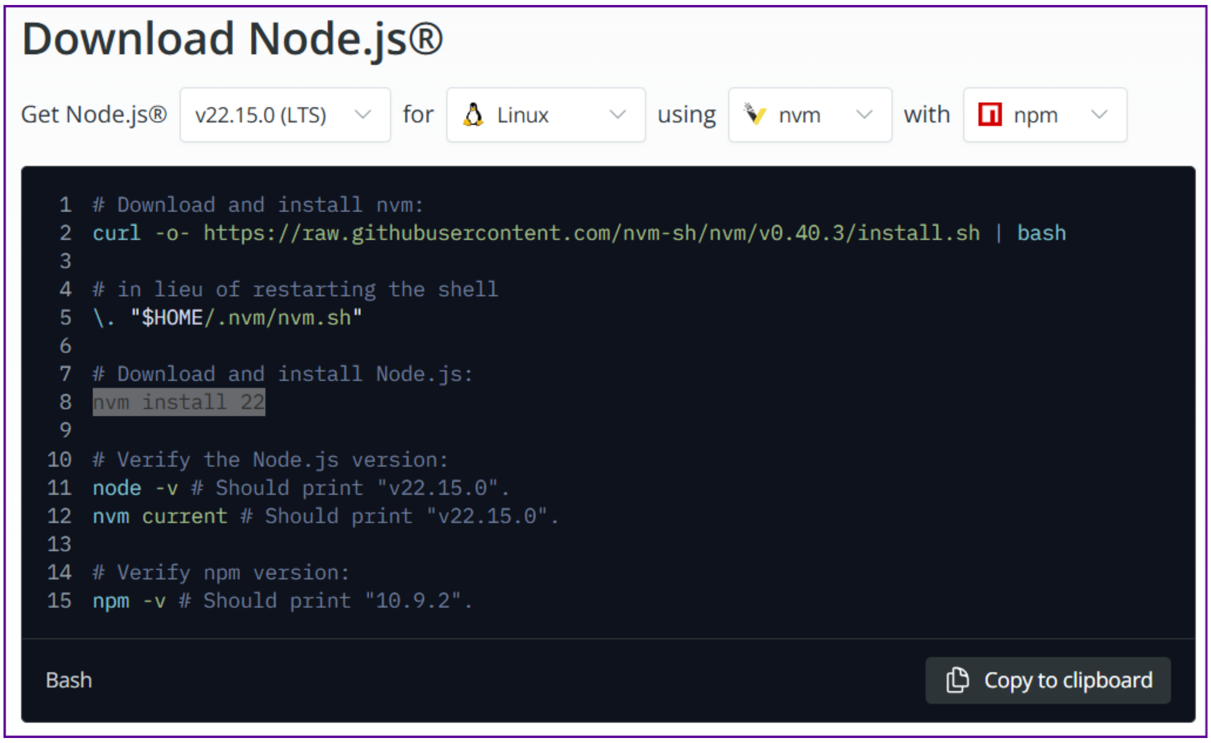
AWS COST OPTIMISATION USING AWS-COST-CLI

Install AWS COST CLI

https://nodejs.org/en/download



```
# Download and install nvm:
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash

# in lieu of restarting the shell
\. "$HOME/.nvm/nvm.sh"

# Download and install Node.js:
nvm install 22

# Verify the Node.js version:
node -v # Should print "v22.15.0".
nvm current # Should print "v22.15.0".

# Verify npm version:
npm -v # Should print "10.9.2".
```

npm install -g aws-cost-cli

Default Usage -Cost Breakdown by Service

aws-cost

- Retrieves total cost breakdown by service.

---

2. Cost Summary (No Service Breakdown)

aws-cost --summary

- Displays total cost only, without breakdown.

---

3. Plain Text Output (No colors/tables)

aws-cost --text

- Outputs the cost report in plain text format.

---

4. JSON Output (For automation)

```
aws-cost --json
```

- Outputs the cost report in JSON format.

---

5. Use Specific AWS Credentials

```
aws-cost -k <AWS_ACCESS_KEY> -s <AWS_SECRET_KEY> -r <AWS_REGION>
```

- Pass AWS credentials directly in the CLI.

---

6. Use AWS Profile (Configured via aws-cli)

```
aws-cost -p <profile_name>
```

- Use configured AWS CLI profiles.

---

7. Send Report to Slack (Breakdown Message)

```
aws-cost --slack-token <SLACK_TOKEN> --slack-channel <CHANNEL_ID>
```

- Sends the cost report directly to a Slack channel.

---

8. Docker Usage

```
docker build -t aws-cost-cli .
docker run aws-cost-cli
```

- Build and run the CLI inside Docker.

---

9. Help and Version Info

```
aws-cost --help       # Shows help info
aws-cost --version    # Displays CLI version
```

SLACK CONFIG

Step 1: Create a Slack App

1. **Go to:** Slack API: Create App https://api.slack.com/apps
2. **Click**: Create New App
3. **Choose**: From scratch
4. **App Name**: AWS Cost Notifier
5. **Select Workspace**: Choose your **Slack workspace**.

Step 2: Set Slack App Permissions

1. In your app dashboard:
2. Go to **OAuth & Permissions**.
3. Scroll to **Scopes**:
   - Under **Bot Token Scopes**, add:
     - chat:write → Allows the bot to post messages.
     - chat:write.public → (Optional) Allows posting in public channels without being invited.
     - Files:write → To be able to write to the slack channel

---

Step 3: Install the App in Your Workspace

1. Still in **OAuth & Permissions**, click **Install App to Workspace**.
2. Approve the permissions.
3. **Copy the Bot User OAuth Token**:
   - Starts with xoxb-... → **You'll need this for aws-cost-cli.**

Step 4: Find Your Slack Channel ID

1. Open Slack.
2. Go to the **channel** you want to post in.
3. Click on the **channel name** at the top → **View channel details**.
4. Copy the **Channel ID** (starts with C...).

Step 5: Test Slack Integration with aws-cost-cli

Run the following command:

```
aws-cost --slack-token xoxb-dcbsdvdsbvvsb  --slack-channel C08UUEHEU
```

Invite the Bot in Slack Channel

```
/invite @AWS-COST-Report
```

1. Install Python & pip (if not already)

```
sudo apt update
sudo apt install -y python3 python3-pip python3-venv
```

---

2. Create a Virtual Environment (Optional but Recommended)

```
python3 -m venv venv
source venv/bin/activate
```

1. **Install Slack SDK:**

```
pip install slack-sdk
```

---

2. **Create Python Script (upload_cost_report.py):**

```
from slack_sdk import WebClient
from slack_sdk.errors import SlackApiError

slack_token = "xoxb-6800278956247-8805347127382-SQIckF4laKrziI8ODyqARQRA"
client = WebClient(token=slack_token)

try:
    # Use files_upload_v2 (latest method)
    response = client.files_upload_v2(
        channel="C08PVFZV9PF",
        initial_comment="AWS Cost Report",
        file="cost-report.txt"
    )
    print("File uploaded successfully:", response)

except SlackApiError as e:
    print(f"Error uploading file: {e.response['error']}")
```

3. **Run the AWS Cost CLI + Python Script:**

```
# Generate cost report

aws-cost --text > cost-report.txt


# Upload to Slack

python3 upload_cost_report.py
```