Sure! Here’s a complete **LinkedIn-ready PDF content** combining your EC2 creation playbook and all project explanation. You can copy this to a Markdown or LaTeX editor and export as PDF.

---

# Project: Automating EC2 Instances with Ansible and Passwordless SSH

## Project Overview

This project demonstrates how to **create and manage AWS EC2 instances** using **Ansible**. The main goals were: - Launch multiple EC2 instances (Ubuntu). - Set up **passwordless SSH** from a control node to all managed nodes. - Configure an **Ansible inventory**. - Perform operations like **conditional shutdown** using playbooks.

This project is ideal for DevOps beginners learning **AWS, Ansible, and infrastructure automation**.

---

## Step 1: Launch EC2 Instances via Ansible

**Playbook:** `ec2_create.yml`

```
---
- name: Create EC2 instances
  hosts: localhost
  connection: local
  gather_facts: false
  vars_files:
    - aws_credentials.yml
  tasks:
    - name: Launch EC2 instances
      amazon.aws.ec2_instance:
        name: "{{ item.name }}"
        key_name: "lll"
        instance_type: t2.micro
        security_group: default
        region: us-east-1
        aws_access_key: "{{ ec2_access_key }}"
        aws_secret_key: "{{ ec2_secret_key }}"
        network:
          assign_public_ip: true
        image_id: "{{ item.image }}"
        tags:
          environment: "{{ item.name }}"
      loop:
        - { image: "ami-0360c520857e3138f", name: "manage-node-1" }
        - { image: "ami-0360c520857e3138f", name: "manage-node-2" }
        - { image: "ami-0360c520857e3138f", name: "manage-node-3" }
```

**Key learning:** Automating instance creation saves time and ensures consistency.

---

## Step 2: Passwordless SSH Setup

**Why:** Allows Ansible to run commands/playbooks without entering a password or PEM file every time.

**Commands:**

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa

cat ~/.ssh/id_rsa.pub | ssh -i ~/lll.pem ubuntu@<managed_node_ip> 'mkdir -p
~/.ssh && cat >> ~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys'

ssh ubuntu@<managed_node_ip>  # Test login
```

**Mistakes to avoid:** - Don't generate keys on managed nodes. - Use the correct username (`ubuntu`). - Set proper permissions: `chmod 600 ~/.ssh/authorized_keys`.

---

## Step 3: Create Ansible Inventory (`hosts.ini`)

```
[ubuntu_nodes]
manage-node-1 ansible_host=3.235.91.83 ansible_user=ubuntu
manage-node-2 ansible_host=44.201.22.7 ansible_user=ubuntu
manage-node-3 ansible_host=13.219.250.218 ansible_user=ubuntu

[all_nodes:children]
ubuntu_nodes
```

---

## Step 4: Test Connectivity

```
ansible -i hosts.ini all -m ping
```

✅Expected: `pong` from all nodes.

## Step 5: Shutdown Playbook (`shutdown.yml`)

```
---
- name: Shutdown all Ubuntu instances
  hosts: all_nodes
  become: true
  gather_facts: true
  tasks:
    - name: Shutdown nodes
      ansible.builtin.command: /sbin/shutdown -h now
      ignore_errors: yes
```

**Notes:** - `become: true` → root access. - `ignore_errors: yes` → prevents playbook from failing because SSH closes during shutdown.

---

## Step 6: Lessons Learned

1. SSH key management is critical — generate only on control node.
2. Inventory grouping simplifies management.
3. Ansible tasks need proper permissions.
4. OS differences matter (`ubuntu` vs `centos`).
5. Test connectivity before running playbooks.
6. Automation saves time and reduces errors.

---

## Step 7: Project Outcome

- Successfully launched 3 EC2 instances.
- Configured **passwordless SSH** for all nodes.
- Built Ansible inventory for grouped management.
- Ran conditional shutdown playbooks on Ubuntu nodes.
- Learned **Ansible automation, AWS EC2 management, and SSH practices**.

---

This project can now be showcased on **LinkedIn** as a **DevOps automation project**.

---

**Tip for New Users:** - Start with 1 node to test SSH and playbooks. - Maintain a clean directory for PEM and SSH keys. - Document IPs and hostnames. - Use `ignore_errors: yes` for shutdown tasks. - Match OS and username carefully.