Demo Project Commands

Terraform Commands

- terraform init

→ Initializes Terraform to prepare for deployment. Used before the first plan/apply.

- terraform plan -out=tfplan

→ Generates a plan to review changes before applying them.

- terraform apply -auto-approve

→ Applies infrastructure changes automatically without manual confirmation.

- terraform destroy -auto-approve

→ Destroys all resources; used to reset the environment between tests.

- terraform output

→ Displays all defined output variables such as instance IP or URL.

- terraform output -raw instance_public_ip

→ Retrieves the Elastic IP address assigned to the EC2 instance.

- terraform output -raw app_url

→ Retrieves the complete app URL if defined in outputs.

- terraform taint <resource>

→ Marks a resource for forced recreation on next apply; used to fix corrupted resources.

- terraform state show module.ec2.aws_instance.app | grep vpc_security_group_ids

→ Checks which security group is attached to the EC2 instance.

AWS CLI Commands

- aws ec2 describe-instances --instance-ids <id>

→ Displays detailed info (state, IPs, SGs) for a specific EC2 instance.

- aws ec2 describe-security-groups --group-ids <id>

→ Shows inbound/outbound rules; used to confirm port 8000 is open.

- aws ec2 describe-addresses

→ Lists Elastic IPs and instance associations; used to verify public IPs after apply.

- aws ec2 describe-network-interfaces --filters 'Name=group-id,Values=<sg-id>'

→ Identifies dependencies preventing SG deletion.

- aws ec2 terminate-instances --instance-ids <id>

→ Manually terminates an EC2 instance stuck in a dependency error.

- aws ec2 describe-route-tables --filters 'Name=vpc-id,Values=$(terraform output -raw vpc_id)'

→ Confirms internet route (0.0.0.0/0) exists for connectivity.

- aws ssm start-session --target <instance-id>

→ Opens an interactive shell via AWS Systems Manager; no SSH key needed.

Docker Commands (Inside EC2)

- docker ps -a

→ Lists all running and stopped containers; used to verify Django and PostgreSQL containers are running.

- docker logs <container>

→ Views real-time logs; used to confirm Django migrations and Gunicorn startup.

- docker exec -it <container> bash

→ Accesses container shell for debugging environment or ports.

- docker exec -it <container> ss -tuln | grep 8000

→ Checks if Django is listening on port 8000 inside the container.

- docker images

→ Lists all available Docker images on the instance; used to verify ECR image was pulled successfully.

Network & App Debugging Commands

- curl http://localhost:8000

→ Used inside EC2 to verify the Django app is responding.

- curl http://<public-ip>:8000

→ Used from your Mac to confirm external access through the Elastic IP.

- netstat -tuln | grep 8000

→ Checks if port 8000 is listening (after installing net-tools).

- ss -tuln | grep 8000

→ Alternative to netstat; verifies network socket status inside container.

- apt update && apt install -y net-tools

→ Installs networking tools needed for debugging inside the container.

GitHub Actions & CI/CD

- workflow_dispatch

→ Allows manual workflow triggers for testing deployments.

- aws-actions/configure-aws-credentials@v4

→ Configures AWS credentials in GitHub Actions for Terraform/ECR use.

- hashicorp/setup-terraform@v3

→ Installs Terraform in CI/CD pipeline for automated runs.

- aws-actions/amazon-ecr-login@v2

→ Authenticates Docker to push/pull images from ECR.

- docker build --platform linux/amd64 -t <repo> ./backend

→ Builds the Django app Docker image for the correct CPU architecture.

- docker push <ECR_URL>:latest

→ Uploads the new Docker image to Amazon ECR for deployment.

Troubleshooting Dependency & Destroy Issues

- aws ec2 describe-network-interfaces --filters 'Name=attachment.instance-id,Values=<instance-id>'

→ Checks for network interfaces blocking security group deletions.

- terraform apply -replace='module.ec2.aws_security_group.app_sg'

→ Recreates a security group that failed to destroy properly.

- terraform refresh

→ Updates the local state file with the actual remote resource state.

- terraform state rm <resource>

→ Removes a problematic resource from the Terraform state to fix stuck deletions.