Migration Exam Cheatsheet — Ubuntu commands & step-by-step

A concise, exam-ready checklist you can revise quickly. Focus: commands to install required tech stacks on Ubuntu and exact migration steps (FileZilla transfer + run). Use this for Node.js, Django (Python), PHP, Ruby on Rails, MySQL/Postgres, Docker, and basic S3 steps.

Before you start (VM prep)

1. **Increase VM resources (VirtualBox)**: 4 GB RAM, 2 CPUs. Enable Host I/O Cache for SATA.
2. **Remove installer ISO** (if present) so VM boots from the virtual disk.
3. **Enable SSH in Ubuntu (one-time)**:

```
sudo apt update
sudo apt install -y openssh-server
sudo systemctl enable ssh
sudo systemctl start ssh
sudo systemctl status ssh  # should show active (running)
```

1. **Find Ubuntu IP** (use this in FileZilla):

   ```
   ip addr
   # look for inet 192.168.x.x or 10.x.x.x under the adapter (enp0s8/enp3s0)
   ```

2. **FileZilla** on Windows: Host = sftp://<ubuntu-ip>, Port 22, Username = ubuntu, Password = <your password>.

General tips for transfers & reports

- Use FileZilla to drag & drop project folder into `/home/ubuntu`.
- Keep terminal open while running servers; take screenshots in order: transfer, terminal commands, browser output.
- When binding dev servers use `-b 0.0.0.0` so host/Windows can access: `rails server -b 0.0.0.0 -p 3000` or `npm start` with host config.

---

1) Node.js (Express / Node apps)

Install Node & npm on Ubuntu

```
sudo apt update
sudo apt install -y nodejs npm
# Optional: install NodeSource for newer node (22.x recommended if needed)
# curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
# sudo apt install -y nodejs
node -v
npm -v
```

Migrate & run (after FileZilla transfer)

```
cd ~/your_project_folder
npm install
npm start   # or node app.js / npm run dev
# If app listens only on localhost and you want Windows to access, configure to
listen on 0.0.0.0 or use port-forwarding.
```

Browser: `http://localhost:3000` (or `http://<ubuntu-ip>:<port>` from Windows)

Screenshots: npm start terminal, browser showing app, FileZilla transfer.

---

2) Python + Django

Install Python, pip, venv

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

Typical project steps (after transfer)

```
cd ~/your_django_project
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
# or if requirements missing: pip install django
```

```
python manage.py migrate
python manage.py runserver 0.0.0.0:8000
```

Browser: `http://localhost:8000` or `http://<ubuntu-ip>:8000`

Notes: If using `ALLOWED_HOSTS`, add your ubuntu-ip or `['*']` in settings for exam.

Screenshots: venv creation, pip install output, `runserver` output, browser.

---

3) PHP (Apache + PHP)

Install Apache & PHP

```
sudo apt update
sudo apt install -y apache2 php libapache2-mod-php php-mysql php-xml php-
mbstring
sudo systemctl enable apache2
sudo systemctl start apache2
sudo systemctl status apache2
```

Deploy (after transfer)

```
# move or copy project to web root
sudo mv ~/php-project-folder /var/www/html/yourphp
sudo chown -R www-data:www-data /var/www/html/yourphp
sudo chmod -R 755 /var/www/html/yourphp
sudo systemctl restart apache2
```

Browser: `http://localhost/yourphp` (or `http://<ubuntu-ip>/yourphp`)

If index.php not auto-loading: edit `/etc/apache2/mods-enabled/dir.conf` to place `index.php` first and restart apache.

Screenshots: Apache status, project folder in `/var/www/html`, browser output.

---

4) Ruby on Rails

(Simplest, use Ubuntu package Ruby) Install Ruby & Rails

```
sudo apt update
sudo apt install -y ruby-full build-essential libsqlite3-dev sqlite3
sudo gem install bundler rails
ruby -v
rails -v
```

Run project (after transfer)

```
cd ~/rails_project
bundle install
rails db:setup    # or rails db:create && rails db:migrate
rails server -b 0.0.0.0 -p 3000
```

Browser: `http://localhost:3000` or `http://<ubuntu-ip>:3000`

Note: Old Rails 3 projects may fail on modern Ruby. For exam, use a modern sample or create a fresh app in Ubuntu if GitHub blocked.

Screenshots: bundle install, rails server, browser.

---

5) Databases

MySQL / MariaDB

```
sudo apt update
sudo apt install -y mysql-server libmysqlclient-dev
sudo systemctl enable mysql
sudo systemctl start mysql
# secure install
sudo mysql_secure_installation
```

Create DB/user (example):

```
sudo mysql -u root -p
CREATE DATABASE mydb;
CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON mydb.* TO 'myuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

PostgreSQL

```
sudo apt install -y postgresql postgresql-contrib libpq-dev
sudo systemctl enable postgresql
sudo systemctl start postgresql
sudo -u postgres createuser --interactive  # create role
sudo -u postgres createdb mydb
```

Screenshots: DB service status, created DB list, connection test.

---

6) Docker (optional question)

Install Docker & Docker Compose

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -
o /etc/apt/keyrings/docker.gpg
echo
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/
docker.gpg] https://download.docker.com/linux/ubuntu
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /
dev/null
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
sudo usermod -aG docker $USER
newgrp docker
docker --version
docker compose version
```

Run sample container:

```
docker run --rm -p 8080:80 nginx:alpine
# visit http://localhost:8080
```

Screenshots: docker version, running container, browser showing nginx page.

---

7) AWS S3 basics (if asked in migration tasks)

You can demonstrate uploading a file to S3 using AWS CLI or SDK. If college account blocked, just show commands.

Install AWS CLI v2

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
aws --version
```

Configure (only if credentials allowed)

```
aws configure
# enter access key, secret, region
```

Upload file example

```
aws s3 cp ./file.txt s3://your-bucket-name/ --acl private
```

Screenshots: aws --version, aws s3 ls s3://your-bucket (if allowed). If AWS account not permitted, write the commands in the report and explain.

---

8) Common troubleshooting commands

- Check listening ports:

  ```
  ss -tulnp | grep 3000
  ```

- Check process:

  ```
  ps aux | grep rails
  ```

- Check disk usage:

```
df -h
```

• Check logs (systemd):

```
sudo journalctl -u apache2 --no-pager -n 50
sudo journalctl -u ssh --no-pager -n 50
```

---

9) Screenshot checklist (universal)

For each migration question include: 4–6 screenshots 1. Project on Windows (local) — folder listing 2. FileZilla connected & transfer in progress 3. Ubuntu terminal showing commands and success (install/build/migrate) 4. Ubuntu terminal showing server running 5. Ubuntu browser showing the app (localhost) 6. (Optional) Windows browser accessing Ubuntu IP:port

---

10) Quick copy-paste command packs (one-line blocks)

Node (full):

```
sudo apt update && sudo apt install -y nodejs npm openssh-server
```

Django (full):

```
sudo apt update && sudo apt install -y python3 python3-venv python3-pip openssh-server
```

PHP (full):

```
sudo apt update && sudo apt install -y apache2 php libapache2-mod-php php-mysql php-xml php-mbstring
```

Ruby (full):

```
sudo apt update && sudo apt install -y ruby-full build-essential libsqlite3-dev
sqlite3
sudo gem install bundler rails
```

MySQL (full):

```
sudo apt update && sudo apt install -y mysql-server libmysqlclient-dev
```

Docker (full):

```
echo "install docker steps..."  # use the multi-line steps above in actual use
```

If you want this exported as a **Word (.docx)** file I can create a downloadable document for you. Tell me the filename you prefer (default: `Migration_Exam_Cheatsheet.docx`) and I will generate it now.