CS494 Edge Computing Systems Reference Sheet

Edge systems operate under constraints. Expect limited power, limited bandwidth, intermittent connectivity and imperfect hardware. Design for recovery, not perfection. Common edge pipeline: Sense → Process → Decide → Act. Move computation toward constraints, not convenience.

Linux Fundamentals (Edge)

Navigation and Files

- `ls`, `ls -l`, `ls -lh`
- `cd`, `pwd`
- `cp`, `mv`, `rm -r`
- `find . -name "*.py"`
- `du -sh *`

Filesystem Locations

- `/etc`: configuration
- `/var/log`: logs
- `/opt`: deployed software
- `/tmp`: temporary files
- `/dev`: devices

If you do not know where a file lives, you cannot debug it. Configuration, logs, and devices are almost never in the project directory.

Permissions and Ownership

- `chmod +x file`
- `chmod 755 file`
- `chown user:group file`
- Permission denied often means wrong group membership

Processes and Signals

- `ps aux`, `top`, `htop`
- `Ctrl-C`: SIGINT (polite stop)
- `Ctrl-Z`: SIGTSTP (pause)
- `fg`, `bg`
- `kill -TERM <pid>`
- `kill -9 <pid>` (last resort)
- `nohup cmd &`

System State

- `uname -a`, `arch`
- `uptime`
- `df -h`, `free -h`

Devices and Hardware

- `lsusb`, `lsblk`
- `dmesg | tail`
- `/dev/video*`, `/dev/tty*`

If hardware disappears, check power, cables, permissions, and kernel messages before changing code.

Services and Boot

- `systemctl status <svc>`
- `systemctl start|stop|restart <svc>`
- `journalctl -u <svc>`
- `journalctl -f`

Use services for long-running components that must restart automatically or run at boot.

Git

Core Workflow

- `git clone <url>`
- `git status`
- `git branch`
- `git checkout <branch>`

Making Changes

- `git add <file>`
- `git commit -m "msg"`
- `git push`
- `git pull`

Debugging Git

- `git log --oneline`
- `git diff`
- `git restore <file>`

Commit history is evidence. Frequent, descriptive commits make failures traceable.

Python Runtime

Virtual Environments

- `python3 -m venv .venv`
- `source .venv/bin/activate`
- `which python`
- `pip install -r requirements.txt`
- `pip freeze > requirements.txt`

Virtual environments isolate Python dependencies. Containers isolate the entire runtime. They solve different problems and are often used together.

Environment Sanity Checks

- `python --version`
- `pip --version`
- `pip list`
- `env | sort`

Execution Patterns

- `python main.py`
- `python -m package.module`

```
if __name__ == "__main__":
    main()
```

Prefer `-m` execution when imports matter. It preserves package context.

Project Layout

- `main.py` or `__main__.py`
- `requirements.txt`
- `src/` or a package directory
- `tests/` (if present)
- `.env` (do not commit)

Containers

When to Use What

- Script: quick test, interactive
- Container: reproducible runtime, portable
- Service: long-running, auto-restart, boots with device

Build and Run

- `docker build -t image .`
- `docker run image`
- `docker run -p 8080:8080 image`
- `docker ps -a`

Run Options

- `-it`: interactive
- `--rm`: auto cleanup
- `-v host:container`: volumes
- `--env VAR=value`
- `--name <name>`

Inspection and Debug

- `docker logs <ctr>`
- `docker exec -it <ctr> bash`
- `docker inspect <ctr>`

Cleanup

- `docker stop <ctr>`
- `docker rm <ctr>`
- `docker image prune`

```
FROM python:3.11-slim
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python", "main.py"]
```

If a container fails, check logs first, then inspect configuration, mounts, and port mappings.

Networking

Connectivity

- `ip addr`
- `ping <host>`
- `curl <url>`

Ports and Services

- `ss -lntp`
- `netstat -lntp`
- Container mapping: `-p host:container`

Listen means waiting. Bind means attaching to an interface. Connect means initiating traffic. Containers must expose and map ports explicitly.

If It Does Not Connect

- Is the service running?
- Is the port listening?
- Is the container port mapped?
- Is the firewall blocking?
- Are you on the right network?

Logging and Measurement

Python Logging

```
import logging
logging.basicConfig(level=logging.INFO)
logging.info("message")
```

Where Logs Go

- stdout and stderr
- Files on disk
- systemd journal
- Container logs

Logs are evidence. A system that fails silently is not done.

Measurement Reality

- First run is often slower
- Measurement changes behavior
- Logging affects timing
- Network variability matters

Edge Deployment Checklist

- Runs on target hardware
- Correct CPU architecture
- Dependencies pinned
- No hardcoded paths
- Network reachable
- Logs observable
- Restart behavior tested
- Power loss tolerated

A system that fails with evidence is debuggable. A system that fails silently is not.