```python
# filename: defcon_simple_tm.py
# Minimal pytm example: Business -> WebApp -> DB, and Business -> SFTP -> S3
# Run:
#   pip install pytm
#   python defcon_simple_tm.py --dfd          # make DFD diagram (dfd.png)
#   python defcon_simple_tm.py --report md    # generate markdown threats
#
# Tip: toggle booleans (e.g., usesHTTPS=False) live to show how risks change.

from pytm import TM, Actor, Server, Process, Datastore, ExternalEntity, Dataflow,
Boundary

tm = TM("DEFCON Simple Enterprise TM")
tm.description = "Two simple flows to demo STRIDE with pytm."
tm.isOrdered = True  # draw flows in order for a clean DFD

# ---- Boundaries (trust zones) ----
corp_net    = Boundary("Corp Network")
dmz         = Boundary("DMZ")
cloud       = Boundary("AWS Cloud")
internet    = Boundary("Internet")

# ---- Key entities ----
# Human/business side
business_user = Actor("Business User")              # human actor
business_user.inBoundary = internet                 # simulate remote user

# Web application stack
web_app = Server("Web App")
web_app.inBoundary = dmz
web_app.OS = "Linux"
web_app.isHardened = True
web_app.usesHTTPS = True

app_db = Datastore("Orders DB")
app_db.inBoundary = dmz
app_db.isSQL = True
app_db.isEncrypted = True           # at-rest encryption
app_db.storesPII = True             # flip to False to show fewer issues

# File transfer path
batch_job = Process("Nightly Batch")
batch_job.inBoundary = corp_net
batch_job.isPrivileged = True       # runs with service creds

sftp_gw = Server("SFTP Gateway")    # e.g., AWS Transfer Family
sftp_gw.inBoundary = cloud
sftp_gw.OS = "Managed"
sftp_gw.isHardened = True
sftp_gw.usesSSH = True

s3_bucket = Datastore("S3 Bucket")
s3_bucket.inBoundary = cloud
s3_bucket.isObjectStore = True
```

```
s3_bucket.isEncrypted = True
s3_bucket.hasAccessLogging = True    # helpful control to discuss

# Optional external dependency (to show supply-chain)
email_service = ExternalEntity("Email/SMS Provider")
email_service.inBoundary = internet

# ---- Dataflows (2 scenarios) ----

# (1) Business User -> Web App -> DB
login_req = Dataflow(business_user, web_app, "Login / App Requests")
login_req.protocol = "HTTPS"
login_req.dstPort = 443
login_req.data = "Credentials, Session Token, Order Data"
login_req.isEncrypted = True
login_req.authenticated = True

db_query = Dataflow(web_app, app_db, "Read/Write Orders")
db_query.protocol = "TLS-to-DB"
db_query.dstPort = 5432
db_query.isEncrypted = True
db_query.data = "Order Records (PII)"
db_query.authenticated = True

notify_user = Dataflow(web_app, email_service, "Order Status Notification")
notify_user.protocol = "HTTPS API"
notify_user.isEncrypted = True
notify_user.data = "Order ID, masked PII"

# (2) Batch Process -> SFTP Gateway -> S3
stage_extract = Dataflow(batch_job, sftp_gw, "Push CSV via SFTP")
stage_extract.protocol = "SFTP"
stage_extract.dstPort = 22
stage_extract.isEncrypted = True
stage_extract.data = "Daily Orders Export (PII)"
stage_extract.authenticated = True  # service key or SSH key

land_to_s3 = Dataflow(sftp_gw, s3_bucket, "Land file to S3")
land_to_s3.protocol = "AWS SDK"
land_to_s3.isEncrypted = True
land_to_s3.data = "CSV objects"
land_to_s3.authenticated = True

# ---- Quick toggles to demo risk changes live ----
# web_app.usesHTTPS = False         # show MITM/auth threats increase
# app_db.isEncrypted = False        # at-rest threats
# s3_bucket.hasAccessLogging = False# detection/forensics discussion
# stage_extract.authenticated = False# unauth uploads

if __name__ == "__main__":
    tm.process()
```