Checklist for Infrastructure

The Infrastructure category ensures Orgo is deployable, maintainable, and scalable while providing tools for monitoring, backups, and system updates. This checklist ensures smooth operations and supports offline/online configurations.

---

1. Deployment

[ ] Automation:

Include deployment scripts (e.g., setup.py) for automated setup of dependencies and configurations.

[ ] Docker:

Provide Dockerfile and docker-compose.yaml for containerized deployment.

[ ] Kubernetes:

Include Kubernetes manifests (deployment.yaml, service.yaml, ingress.yaml) for scalable deployment.

[ ] Environment-Specific Settings:

Use separate configurations for development, staging, and production.

[ ] CI/CD Integration:

Automate deployments with CI/CD tools (e.g., GitHub Actions, GitLab CI).

---

2. Backup and Restore

[ ] Database Backups:

Automate daily backups for PostgreSQL and SQLite databases.

Store backups in secure locations (e.g., encrypted storage, S3).

[ ] File Backups:

Backup configuration files, logs, and other critical data.

[ ] Restore Mechanism:

Provide scripts to restore data from backups.

Validate restoration by testing on a staging environment.

Example Script: backup.py

import os

import shutil

from datetime import datetime

def backup\_database(db\_path, backup\_dir):

os.makedirs(backup\_dir, exist\_ok=True)

timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")

backup\_file = os.path.join(backup\_dir, f"db\_backup\_{timestamp}.sql")

shutil.copy(db\_path, backup\_file)

print(f"Backup created: {backup\_file}")

---

3. Offline Synchronization

[ ] Synchronization Scripts:

Provide scripts to sync .pst files for offline email processing.

Store synced data in /data/sync\_files/.

[ ] Conflict Resolution:

Handle data conflicts (e.g., last-modified timestamps).

[ ] Logging:

Log all sync activities for traceability.

---

4. Monitoring and Alerts

[ ] Health Checks:

Monitor CPU, memory, disk usage, and email server connectivity.

Provide a health\_check.py script for automated checks.

[ ] Performance Metrics:

Track response times for workflows, APIs, and database queries.

[ ] Alerts:

Send alerts for critical issues (e.g., database downtime, high CPU usage).

[ ] Integration:

Use monitoring tools like Prometheus, Grafana, or Elastic Stack.

Example Health Check Script

import psutil

def check\_system\_health():

cpu\_usage = psutil.cpu\_percent()

memory = psutil.virtual\_memory()

print(f"CPU Usage: {cpu\_usage}%")

print(f"Memory Usage: {memory.percent}%")

if cpu\_usage > 80 or memory.percent > 80:

print("Warning: High resource usage detected!")

---

5. Logging

[ ] Infrastructure Logs:

Log deployment actions, backup operations, and monitoring results.

[ ] Error Logs:

Capture errors from scripts (e.g., failed backups, health check issues).

[ ] Retention Policies:

Enforce retention periods for infrastructure logs (e.g., 6 months).

---

6. Security

[ ] Access Control:

Restrict access to infrastructure scripts and sensitive directories.

[ ] Encryption:

Encrypt backups and configuration files containing sensitive data.

[ ] Audit Trails:

Log all infrastructure changes (e.g., deployments, script executions).

[ ] Environment Variables:

Store sensitive values (e.g., credentials, API keys) as environment variables.

---

7. Retention Policies

[ ] Database Backups:

Retain backups for a defined period (e.g., 30 days).

[ ] Logs:

Rotate and delete old logs based on policy (e.g., keep 6 months of logs).

---

8. Scalability

[ ] Horizontal Scaling:

Ensure containers or services can be scaled horizontally (e.g., multiple instances of workflow handlers).

[ ] Task Queues:

Use tools like Celery with Redis/RabbitMQ for asynchronous task management.

[ ] Database Optimization:

Implement indexing and query optimization for PostgreSQL.

---

9. Testing

[ ] Infrastructure Tests:

Test deployment scripts and manifests in a staging environment.

Verify backups and restores are functional.

[ ] Load Testing:

Simulate high traffic to ensure scalability (e.g., 1,000 tasks in parallel).

[ ] Error Handling:

Test how scripts handle failures (e.g., missing dependencies, insufficient permissions).

---

10. Automation

[ ] Deployment Pipelines:

Automate deployments with tools like Ansible, Terraform, or Kubernetes Helm charts.

[ ] Scheduled Backups:

Automate backups using cron jobs or scheduled tasks.

[ ] Alerting:

Use integrations (e.g., Slack, email) to send alerts for critical issues.

---

Example Application

Infrastructure Scripts Directory

/infrastructure/

├── deployment/

│ ├── docker-compose.yaml # Docker Compose file for containerized deployment

│ ├── setup.py # Initial setup script

│ ├── kubernetes/

│ │ ├── deployment.yaml # Kubernetes deployment manifest

│ │ ├── service.yaml # Kubernetes service definition

│ │ └── ingress.yaml # Kubernetes ingress definition

├── scripts/

│ ├── backup.py # Database and file backup script

│ ├── restore.py # Restore data from backups

│ ├── sync.py # Offline synchronization script

│ ├── log\_cleaner.py # Log rotation and cleanup script

│ └── health\_check.py # System health check script

├── monitoring/

│ ├── performance\_metrics.py # Tracks system performance metrics

│ ├── email\_monitor.py # Checks email server connectivity

│ └── alerts.py # Sends alerts for system issues

Checklist Applied

[ ] Deployment:

Scripts (setup.py, docker-compose.yaml) automate deployment and configuration.

[ ] Monitoring:

health\_check.py monitors CPU, memory, and disk usage.

[ ] Backups:

backup.py automates daily database backups.

Backups are stored securely and encrypted.

---

Optimized Workflow

1. Start with Deployment Automation:

Create scripts and manifests for consistent deployments.

2. Add Backup and Restore:

Automate backups and test restores in staging.

3. Implement Monitoring:

Write health check scripts and configure alerts.

4. Test and Optimize:

Validate infrastructure scripts in a staging environment.

---

This checklist ensures Orgo’s infrastructure is robust, scalable, and secure. Would you like to focus on a specific infrastructure component (e.g., deployment, monitoring)?