Service Provisioning Checlist

Purpose

This checklist is to ensure that all aspects of a new service are provisioned properly, completely, and in the correct order to prevent potential failures elsewhere in the system.

Steps

- [ ] Determine any potential impact to any other services; see things to look out for below
  - [ ] Is this service going to be running on app-01 or a different host?
  - [ ] Is it going to utilize SSO auth?
  - [ ] Is it going to need a database? Service files folder in /mnt/data/services on app-01?
  - [ ] Is it going to need any other secrets?
  - [ ] Does this service need to be monitored?
  - [ ] Does this service's data need to be backed up?
  - [ ] Exposed to the public internet?
  - [ ] Send pings to HealthChecks?
  - [ ] Utilizing a mailserver or ntfy to send notifications?
- [ ] Determine the most feasible deployment method
  - [ ] Docker container
  - [ ] nixOS module (preferred for reproducibility and programmatic configuration)

> Check on repology.org to verify if the nixOS module is up to date with upstream before choosing to use the nixOS module

- [ ] If the service uses a database
  - [ ] Typically, I create databases whose names are the same as the service name e.g. for forgejo, the database name is forgejo
  - [ ] Create and store database secrets under Bitwarden Secrets Manager, using the following naming convention: `webservices.<service name>.db_pass`
  - [ ] Create a new branch in `Projects/ansible` from staging using the naming convention *databases.<service name>*

- [ ] Add an entry under the appropriate database server's `host_vars` file. Run the appropriate playbook to auto-provision that database

- [ ] If this service needs a folder in the filesystem, create that respective folder under /mnt/data/services and create a symlink to it from /srv such that the symlink's location is /srv/<service name>
- [ ] If this service is being provisioned using Docker
  - [ ] Follow Docker service provisioning procedure to create a docker-compose file and appropriate config and .env files
- [ ] If this service is being provisioned using a nixOS module
  - [ ] Write wrapper module as needed
  - [ ] Add the database secrets to the SOPS config. This will be used for both database access and borgmatic backups
  - [ ] Add a SOPS secrets config block to the host config running the service and pass to the module to prevent the secret being exposed in the store
- [ ] If this service needs to send pings to HealthChecks
  - [ ] Create a ping in the HealthChecks service
  - [ ] Configure the service to contact the specific endpoint provisioned with the check
- [ ] If this service needs to send notifications using ntfy
  - [ ] Create a channel if needed and set permissions
  - [ ] Create a token with the appropriate permissions, save it to SOPS if needed
  - [ ] Configure the service with the token
- [ ] If this service needs to send email notifications
  - [ ] Configure the email server and run a test using the SMTP relay credentials
- [ ] If this service has metrics that need to be scraped
  - [ ] Set a port that the grafana-alloy collector can poll
  - [ ] Add an alloy configuration under the service's module
- [ ] If this service needs to be monitored for uptime
  - [ ] Determine if this service has a specific health/up monitoring endpoint
  - [ ] Add a check for uptime-kuma but do not engage yet
- [ ] Test service on a separate testing VM if needed
- [ ] When the service is ready to deploy, push to `staging`
- [ ] Test that the nixOS config builds and starts successfully, or the docker-compose project starts normally
- [ ] Enable uptime-kuma service monitor

Vikunja Copy-Paste Version

Revision #9
Created 2026-01-13 05:46:45 UTC by etorres
Updated 2026-01-13 06:29:30 UTC by etorres