Digital Operations Framework

A reusable operating model for managing digital identity, accounts, domains, web services, and security controls.

Purpose

The Digital Operations Framework provides a structured approach for managing digital identity, email addresses, accounts, domains, web services, security controls, and online operations.

The goal is to create a maintainable system that reduces risk, improves visibility, limits exposure, and keeps digital services organized over time.

This framework can be applied to individuals, small teams, independent projects, personal brands, or small organizations.

Core Principles

1. Protect the primary identity

The primary email address or identity should be reserved for trusted, long-term, high-value use.

It should not be used for random services, trials, newsletters, temporary accounts, low-trust platforms, or one-off registrations.

Use the primary identity only when the relationship is important, durable, and expected to remain relevant over time.

Examples of appropriate use:

Personal communication
Important professional accounts
Core recovery accounts
High-trust services
Long-term identity platforms

2. Use purpose-based addresses

Permanent email addresses should be created around clear operational purposes.

Example structure:

personal@example.com	Primary personal or professional identity
contact@example.com	Public general contact
security@example.com	Security reports and responsible disclosure
admin@example.com	Administrative services and platform management
domains@example.com	Domain, DNS, registrar, and hosting administration
finance@example.com	Banking, tax, invoices, subscriptions, and payments
media@example.com	Media, streaming, content, and publishing platforms
games@example.com	Gaming and entertainment platforms

These addresses should be limited, intentional, and easy to understand.

Avoid creating a permanent email address for every individual service unless there is a strong operational reason.

3. Prefer service-specific aliases

For most services, use unique service-specific aliases.

Preferred pattern:

```
service-specific-alias@example-alias-domain.com
```

Each service should ideally have its own unique alias.

Benefits:

- Leaks are contained
- Spam can be stopped per service
- Compromised services are easier to identify
- The primary identity remains protected
- Account ownership remains traceable
- Password manager entries remain clear

4. Use fallback categories when aliases are rejected

Some services reject known alias or forwarding domains.

When this happens, use a category-level fallback address instead of creating a new permanent address for a single low-value service.

Example fallback model:

media@example.com	media services that reject aliases
games@example.com	games or entertainment services that reject aliases
shop@example.com	stores or commerce platforms that reject aliases

Decision rule:

Preferred:	unique service-specific alias
Fallback:	category-level address
Avoid:	permanent address per low-value service
Never:	primary identity for disposable or low-trust services

5. Treat the inbox as an action queue

The inbox should represent items that still require attention.

Recommended workflow:

```
New mail arrives  Inbox + label
Handled mail  Archive
Archived mail remains searchable through labels
```

This creates a clear operational meaning:

```
Inbox + Finance label = finance item requiring attention
Finance label only = finance item already handled
```

6. Use labels for context

Labels identify what an email is about.

Recommended labels:

```
Finance
Domains
Security
Administration
Media
Games
Shopping
Projects
Receipts
Notifications
```

Labels are useful because one message can belong to more than one context without being moved out of the inbox.

7. Use folders selectively

Folders should be used for mail that can safely bypass the inbox or belongs in archive-style storage.

Good folder candidates:

```
Newsletters
Receipts
Promotions
Automated notifications
Low-priority updates
Archived project mail
```

Important mail should generally remain visible in the inbox until reviewed or handled.

8. Store all credentials in a password manager

A password manager should be the source of truth for account access.

Each entry should include:

- Service name
- Login URL
- Email address or alias used
- Unique password
- 2FA status
- Recovery details
- Relevant notes
- Account category

Passwords must be unique per service and should not be reused.

9. Require 2FA for important accounts

Two-factor authentication should be enabled for important accounts wherever possible.

Priority accounts:

- Email
- Password manager
- Banking
- Domain registrar
- DNS provider
- Hosting provider
- Cloud platforms
- Code repositories
- Social media
- Financial platforms
- Business-critical tools

Prefer authenticator apps, passkeys, or hardware security keys over SMS where possible.

10. Treat domains as infrastructure

Domains should be managed as operational assets.

For each important domain, maintain:

- DNSSEC enabled
- SPF configured
- DKIM configured
- DMARC configured
- CAA records where appropriate
- HTTPS enabled
- Security headers configured
- Clear DNS ownership
- Documented purpose
- Documented hosting location

Domains should have clear ownership, purpose, and maintenance expectations.

11. Apply a baseline to public websites

Each public website should follow a consistent baseline.

Recommended static site structure:

```
index.html
style.css
site.js
_headers
robots.txt
sitemap.xml
404.html
README.md
.well-known/security.txt
```

Optional:

```
humans.txt
favicon.svg
og-image.png
```

Recommended security headers:

```
Strict-Transport-Security
Content-Security-Policy
X-Frame-Options
X-Content-Type-Options
Referrer-Policy
Permissions-Policy
```

Security headers should be configured at the hosting or edge layer where possible.

12. Define one canonical domain

Each website should have one canonical domain.

Example:

```
https://example.com/
```

Alternative hostnames should redirect to the canonical version.

Example:

https://www.example.com/	https://example.com/

This improves consistency, search indexing, analytics, and security testing.

13. Keep public information intentional

Public-facing pages should disclose only what is useful and appropriate.

Appropriate public information:

- General purpose
- Public contact address
- Security contact address
- High-level project or organization description
- Public documentation

Avoid publishing:

- Internal account structures
- Private infrastructure details
- Administrative workflows
- Recovery information
- Sensitive hostnames
- Unnecessary personal data

14. Use layered security

Security should not depend on a single control.

A strong setup uses multiple layers:

- Unique aliases
- Unique passwords
- Two-factor authentication
- SPF, DKIM, and DMARC
- DNSSEC
- HTTPS
- Security headers
- Access separation
- Minimal exposure
- Regular review
- Clear documentation

The objective is to reduce blast radius. If one service is compromised, the rest of the environment should remain protected.

15. Accept reasonable provider limitations

Some limitations are controlled by service providers and cannot be fully resolved by the user or administrator.

Examples:

- Hosting providers may limit TLS cipher configuration
- Email providers may not support IPv6 on all mail servers
- Some services may reject email aliases
- Some platforms may not support strong 2FA methods

These limitations should be documented and accepted where the risk is reasonable.

The objective is strong practical security, not chasing perfect test scores at the cost of maintainability.

16. Review important services periodically

Important accounts and services should be reviewed on a recurring basis.

Review checklist:

- Is the account still needed?
- Is the email address or alias appropriate?
- Is the password unique?
- Is 2FA enabled?
- Are recovery options current?
- Is ownership documented?
- Is the account stored correctly in the password manager?
- Does it belong in the correct category?

Suggested review frequency:

Critical services:	every 3–6 months
Financial and domain services:	every 3–6 months
General services:	annually
Unused accounts:	remove when discovered

17. Use a clear new-account decision flow

When creating a new account, follow this process:

1. Is the account important, long-term, or identity-related?	Yes use an appropriate purpose-based address
1. Is the account important, long-term, or identity-related?	No use a service-specific alias
2. Does the service accept the alias?	Yes use the alias
2. Does the service accept the alias?	No use a category-level fallback address
3. Does the service involve money, identity, administration, or recovery?	Yes enable 2FA and document carefully
3. Does the service involve money, identity, administration, or recovery?	No still use a unique password
4. Is this a temporary or low-trust service?	Yes do not use the primary identity

18. Use professional and consistent naming

Names should be clear, predictable, and easy to understand later.

Recommended naming qualities:

Purpose-based

Consistent

Professional

Short

Descriptive

Maintainable

Examples:

security@example.com

finance@example.com

admin@example.com

domains@example.com

contact@example.com

projects@example.com

Avoid names that are unclear, overly specific, difficult to maintain, or tied to temporary services.

19. Keep the system maintainable

A framework is only effective if it can be understood and maintained over time.

Prefer:

- Clear categories
- Limited permanent addresses
- Service-specific aliases
- Documented decisions
- Simple rules
- Repeatable patterns
- Regular review

Avoid:

- Uncontrolled address creation
- Unclear naming
- Overlapping systems
- Undocumented exceptions
- Manual workflows that will not be maintained
- Security settings added without understanding their impact

20. Operating principle

- Protect core identity.
- Separate services by purpose.
- Contain exposure.
- Document decisions.
- Review regularly.

Summary

The Digital Operations Framework is designed to create a clean, secure, and maintainable digital environment.

It balances privacy, usability, security, and long-term management by separating identities, using aliases where possible, applying consistent technical baselines, and documenting important operational decisions.

The framework should remain practical. It should support daily use, reduce risk, and make future decisions easier.