

---

# SOK Development Guidelines

*A handbook created by developers for developers to help engineering teams align their software practices and share know-how.*

Development Community



2024-11-05

This is a high-level document containing a collection of best practices, commonalities between projects and values proven to be practical. Team's should follow these guidelines when implementing their software.

## Contents

<b>Codebase</b>	<b>3</b>
Version Control . . . . .	3
Peer Review . . . . .	3
Coding Standards . . . . .	3
Architecture . . . . .	3
<b>Release Management</b>	<b>3</b>
<b>Environments</b>	<b>4</b>
Design . . . . .	4
<b>Architecture</b>	<b>4</b>
Infrastructure . . . . .	4
Security . . . . .	4
Compliance . . . . .	5
<b>Operations</b>	<b>5</b>
Monitoring . . . . .	5
<b>Guidance</b>	<b>5</b>
Documentation . . . . .	5
<b>Quality Assurance</b>	<b>6</b>
Automation . . . . .	6

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC 2119](#).

## Codebase

### Version Control

- MUST use version control
- SHOULD use Git
- MUST use `main` branch as a base for development
- MUST have documented version control flow
- MUST fork feature (and release) branches from `main` branch
- SHOULD protect default branch from pushes

### Peer Review

- MUST have a process for peer review
- SHOULD have another developer to approve code changes before executed in production

### Coding Standards

- MUST agree on a coding standard inside a team
- RECOMMENDED use automatic code formatting
- SHOULD use automatic code style checking (linting)

### Architecture

- MUST have only needed components in production (resources, interfaces, dependencies)
- MUST follow common API Guidelines
- MUST make your technology choices visible in SOK’s [Tech Radar](#)

## Release Management

- SHOULD release to production from `main` (trunk)
- MUST have identifiable releases

- RECOMMENDED to release smaller changes often over larger merges
- MUST have (at least) following stages in pipeline (in recommended order): install, test, scan, build, deploy, verify, release

## Environments

### Design

- SHOULD name AWS profiles after account-aliases
- MUST have production separated from testing environments
- SHOULD follow the Principle of Least Privilege

## Architecture

- MUST document all intentionally integrated 3rd party provided services used by the application
- MUST document selected development management tools and purpose of tools

### Infrastructure

- SHOULD use semantic versioned Docker images for building releases
- MUST have centralised logging
- SHOULD collect logs from all deployed environments
- MUST use tags on cloud resources
- MUST have all virtual machines managed by CSP's instance management service (AWS Systems Manager, Azure Automanage, ...)
- SHOULD use cloud managed services whenever possible
- MUST have periodical OS updates for all services not managed by cloud
- MUST document manually managed virtual machines' maintenance and security processes
- MUST encrypt data at rest in cloud
- MUST rotate encryption keys every 365 days (that are used for data at rest) in cloud
- MUST have billing alerts in cloud
- SHOULD have infrastructure as code
- MUST have repeatable infrastructure

### Security

- MUST run automated vulnerability checks for code and fix or mitigate the findings

- SHOULD run automated static code analysis for code quality
- MUST restrict unnecessary access to environments and endpoints from the open internet
- SHOULD conduct threat modelling when the security model of the software changes

## **Compliance**

- MUST ensure that company policy regarding cookie classification and consents are followed (if applicable)
- MUST document the application architecture, personal data flows and needed security measures to ensure compliance with business, privacy and information security requirements
- MUST document all personal data used in testing and development, backups and log data
- MUST have required monitoring functionalities concerning usage of personal data based on data classification
- MUST ensure that only necessary personal data will be collected and/or processed
- MUST implement functionality for defining retention times for all personal data
- MUST be able to erase (or anonymize) all (or partial) personal data on expiration or when requested by data subject

## **Operations**

### **Monitoring**

- MUST have monitoring
- MUST have alarms
- SHOULD provide uptime metric(s) for a service

## **Guidance**

- MUST have process how to handle security notifications

### **Documentation**

- SHOULD use README.md as a central information document inside code repository
- SHOULD document all exceptions with reasoning from Development Guidelines in project's README.md
- MUST have a documented disaster recovery plan

## Quality Assurance

- SHOULD have documented test strategy
- MUST have strategy for test automation and automatic tests designed based on it

## Automation

- SHOULD develop, use and share common solutions (at least) internally for test automation
- RECOMMENDED to consider existing solutions before looking for a new one
- RECOMMENDED to run tests for all code changes as part of CI/CD pipeline