

---

# SOK Development Guidelines

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

Development Community



2022-02-04

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

### Branching

- MUST fork feature (and release) branches from `main` branch
- SHOULD protect default branch from pushes

### Mobile development

- SHOULD use fast-forward merges only from feature branch to `main` branch
- SHOULD implement bug fixes to feature branch and cherry picked them to `main` and potential release branch
- RECOMMENDED to squash feature branches before merging to `main` branch
- MUST preserve release tags forever

### 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

## 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

## Mobile Development

- MUST use semantic versioning for releases (tags)

## Environments

### Data

### Mobile Development

- SHOULD preserve all release artefacts forever

## 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
- SHOULD run automated static code analysis for code quality
- MUST restrict access to development environments from the open internet

## Compliance

- MUST ensure that company policy regarding cookie classification and consents are followed (if applicable)
- 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

## Operations

### Monitoring

- MUST have monitoring

- MUST have alarms

## **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**

### **Automation**

- MUST have some automatic tests for the service that are repeatable
- RECOMMENDED to run tests for all code changes

### **Documentation**

- SHOULD have documented test strategy and which parts of codebase are tested