MAKING IT ALL PRACTICAL

- Commit code updates as frequently as possible, suggested workflow:

Suggested habits (rem it takes 3 weeks to form a new habit):

Before starting work for the day	1. Check build pipeline status 2. Fix testing failures 3. Fix scan findings
Every 1 hour	Commit every function/module updated.
Before Lunch	Commit at least one time.
After Lunch	1. Check build pipeline status 2. Fix testing failures 3. Fix scan findings
Every 1 hour	Commit every function/module updated.
Before logging off from work	Commit at least one time.

*The above steps allow you to "shift left" practically.

- Always target to make the build process as fast as possible.
- Always target to make the build process as transparent as possible.
- Build and repository to artifact as often as possible. Ie whenever new code checking
- Build all projects asset with a single repository.
- Do not store dependencies, documentation in the source repository.
- All dependencies to be downloaded on build.
- Always avoid building failing artifact.
- Always push the artifact to the designated artifact repository.
- If branching, keep the branches always updated, which is been used to build.
    - Build only for specific branches ie for example (Feature or RC or Hotfix)
- Build only one CI pipeline and then a CD pipeline for each environment ie UAT or DEV or PROD or SIT.
- Artifact generated should be environment agnostic.
- Build step should be performed as single step within pipeline per task.
- Abide by the "hard gates" set in place.

OUR SCAN FINDINGS "CHASER"

1st week of every month	To be decided on which teams to follow-up
3rd week of every month	To be decided on which teams to follow-up

DETAILS

• SCAN BEFORE UAT/PROD

• CONTAINER SCANS

i. PRISMA

- Validate if any configuration or secrets been used without proper protection. Ideally should not contain any secrets or misconfiguration.
- Choose right size base image.
- Scan for OS vulnerability regularly.
- Use multi-stage build
- Update image regularly if latest base image available.
- Always get base image with trusted source, digitally signed and verified.
- Scan for vulnerabilities both in development and production environment.
- Make sure user permissions is handled for container. Avoid using root user.
- Always tag vulnerabilities
- Use COPY to Duplicate Files in Docker

ii. CLAIR

- Scan for CVE listed vulnerability regularly.

• LIBRARIES SCANS (NEXUS-IQ)

- Always scan the code to verify if any vulnerability issue is there with the code.
- Always aim to fix the reported vulnerability as much as possible.
- Fix the vulnerability with best option rather than always suppress or ignore.
- Identify all open-source code, dependencies and keep it always on check.
- Open-source library scanning:
- Scans open-source libraries for all popular formats, including NPM, Nuget, Maven, Bowser and
- more.
- Protect your deployments from the latest security risks exposed in your open-source library
- usage.

• FORTIFY SAST SCAN

- Fix the vulnerability with best option rather than always suppress or ignore.
    - All suppressions must come with justifications and mitigations.
- Always scan the code to verify if any vulnerability issue is there with the code.
- Always aim to fix the reported vulnerability as much as possible.

• COMMON FOR ALL SCAN TOOLS

Static Application Security Testing (SAST) - Fortify

- Leveraging on Micro Focus Fortify SCA and SonarQube tool to scan source codes for code quality, security flaws or vulnerabilities.
- Helps to drive quality secured coding practices.

- Lower the development effort, as the findings could be fixed as soon as they are detected.
- Detecting code “smells” quality issues.
- Dashboard provided to help you manage the findings.
- Static Application Security Testing (SAST) & Code Quality
- Using Bamboo Pipeline, trigger the Build and Scan in CI Environment, either by scheduling job or upon any code commit.
- At end of scanning, results are uploaded into a common dashboard.
- User can manage the findings/issues in the dashboard.
- Possibility to enable the automated tickets creation in SHIP (Jira) for the findings.
- Generate PDF and CSV reports for analysis.

Dynamic Application Security Testing (DAST) - WebInspect

- Identifies vulnerabilities in web applications and APIs while they are running in production.
- Support for the latest web technologies and pre-configured policies for major compliance regulations.
- Monitor trends and use dynamic analysis to act on vulnerabilities.

References

It will expedite the individual's SHIP/HATs journey to read up and understand the below practices and practice them daily in the course of work:

1. Continuous Delivery https://martinfowler.com/delivery.html
2. Principles of Continuous Integration
   https://martinfowler.com/articles/continuousIntegration.html
   1. Maintain a Single Source Repository
   2. Automate the Build
   3. Make Your Build Self-Testing
   4. Everyone Commits to the Mainline Every Day
   5. Every Commit Should Build the Mainline on an Integration Machine
   6. Fix Broken Builds Immediately
   7. Keep the Build Fast
   8. Test in a Clone of the Production Environment
   9. Make it Easy for Anyone to Get the Latest Executable
   10. Everyone can see what's happening
   11. Automate Deployment
3. DevOps Culture https://www.devopsinstitute.com/what-is-devops/

   Bridging the daily "grind" of Ops and Devs through:

   1. Frequent comms
   2. Reduction of inter-dept walls
   3. Tools that can "talk" to various platforms
   4. Automate as much as possible
   5. Ease of feedback and automate the handling of these feedback to a high degree

Continuous Integration Pipelines

- Always try keep configuration as dynamic as possible.
- Configuration should be of YAML and more dynamic
- Make it, if possible, configuration as code, eliminating manual configuration and enforce consistence
- Make it, if possible, infrastructure as code, eliminating manual configuration and enforce consistence
- Make sure have service account to be used as common account across all pipelines.
- Have all the common task as scripts shared across all pipelines.

Continuous Deployment Pipelines

- Always try keep configuration as dynamic as possible.
- Automated building and testing of software
- Reporting tools for statistical analysis
- Visibility info, and control over, release artifacts and environments

Functional Test Automation

- Test Scripts Development
- Integrated development environment (IDE) with syntax and keyword highlighting feature
- Supports emulators (Xcode iOS and Android) during development
- Automate the build using SHIP (Bamboo), and then deploy your application to a test environment.
- Triggered the automated test from SHIP (Bamboo) and execute your test cases in the device farm.
- Test reports are pushed to SHIP (Bamboo) as artifacts.