DevSecOps → EndSem Notes

DevSecOps →

DevOps is a collaborative culture b/w dev & Ops teams that emphasizes automation, continuous integration & rapid delivery of software.

DevSecOps is integrating security early & throughout the DevOps lifecycle.

Need of Security

→ High Deployment Speed
→ Security integration throughout the lifecycle
→ Continuous Vulnerability scanning.
→ Fast & secure feedback loops.

In CI →

Security Considerations

→ Source Code Management (Github with 2FA)

→ Run automated static code analysis (Static Application Security Testing [SAST]).

→ Verify dependicies for known vulnerabilities (OWASP Dependency Check)
Tools → Jenkins, Gitlab CI, Circle CI.

Checklist →

→ Secure Credentials in CI Tools

→ Use least privilege access

→ Use signed commits & secure tokens.

In CD →

Security Considerations

- → Secure Artifacts storage & deployment Pipelines.
- → Deploy only signed, verified artifacts
- → Environment segregation (Dev, QA, Prod.)

  Tools → Spinnaker, ArgoCD, Harness.

Risk	Mitigation
→ Tampered Deployment	Signed images/ Artifacts
→ Unauthorized Access	Role-based access control (RBAC)
→ Configuration Drift	Immutable infra + audits.

IaaS (Infrastructure as a Service)

It allows provisioning compute resources through API's or dashboards (eg. AWS, Azure)

Security Implications

→ Shared responsibility model
→ Misconfigured services (e.g. Open S3 buckets).
→ Excessive permissions in IAM roles.

Security Controls →

→ Use IaC (eg. Terraform, CloudFormation)
→ Conduct regular IaC Security Scans.
→ Apply network segmentation & firewalls (eg. AWS Security Groups).

Security Challenges →

1. **Fast Release Cycles**
   Time pressure may bypass security checks.

2. **Toolchain Complexity**
   Each tool adds potential vulnerabilities.

3. **Infrastructure Drift**
   Manual changes diverge from codebases.

4. **Insecure Dependencies**
   Open source packages may be exploitable.

5. **Inadequate Testing**
   Limited time for security test coverage.

→ Securing CI/CD Pipeline

Code Repository

→ Enforced Signed commits
→ 2FA
→ branch protection

CI server

→ Harden build agents
→ avoid secrets in code

CD pipeline

→ Use RBAC
→ audit logs
→ encrypted credentials

Artifacts

→ Scan for malware
→ sign & verify

Tools → Git secrets, SonarQube
Aqua Security (image scanning)

Secure Infrastructure as Code (IaC)

It lets us automate & version control infrastructure
eg.- Terraform, Ansible

Security Principles

→ store IaC in secure Git repos.
→ Avoid hardcoding secrets.
Use → Vault or SSM.
→ Validate IaC against policies
eg.→ Open Policy Agent, Checkov.

Monitoring Examples →

→ Use Prometheus + Grafana for infra metrics.
→ Use ELK stack for log analytics
→ Use tools like Falco for runtime security.

Best Practices & tools →

→ Shift Left Security
→ Principle of least privilege
→ Immutable infrastructure
→ Regular audits & Pen-Testing.

Popular DevSecOps Tools →

Phase	Tool	Purpose
Dev	SonarQube	Code quality & SAST
Build	Trivy, Snyk	Image and dependency scanning
Deploy	Terraform + OPA	Policy-as-code
Monitor	Falco, ELK, Prometheus	Runtime security & logging

Test Driven Security →

→ Inspired by Test Driven Development
→ Security Tests are written before implementation.
→ It validates that security policies are enforced & potential vulnerabilities are caught early.

Tools →

→ Gauntlt
→ OWASP ZAP (automated security tests)
→ NUnit or JUnit with security assertions

Input validation Test →

Prevent XSS/SQLi

Unauthorized Access →

Verify access controls

Rate Limit Test →

Prevent Brute force attacks.

→ Monitoring & Responding to Attacks

Effective Security requires const. visibility into System behaviour & quick response mechanisms to contain & remediate threats.

Tools →

→ Falco → Detect Container anomalies

→ Prometheus + Grafana : Infra. metrics

→ ELK Stack : Log analytics

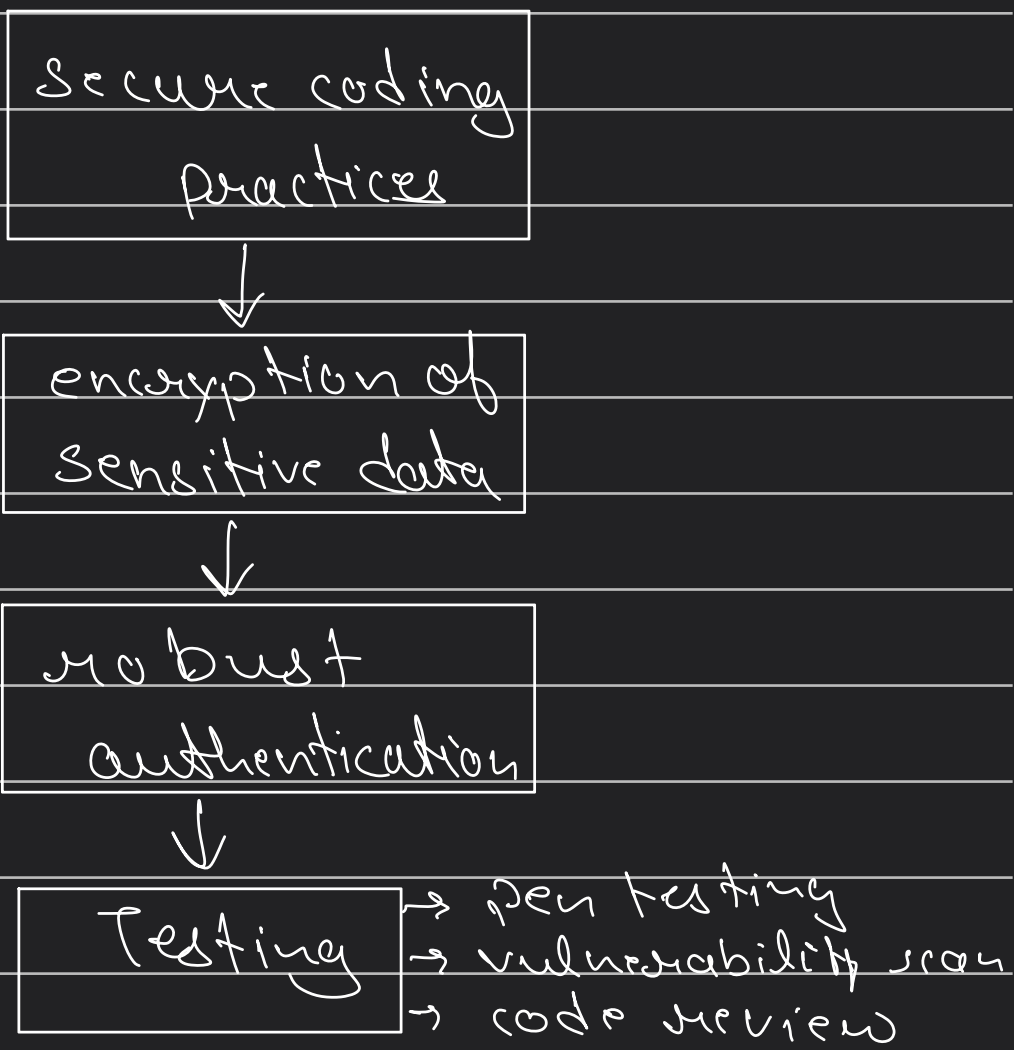
→ Wazuh or OSSEC : Host-based intrusion detection.

Unit-2

1. Securing & Testing Web Apps. →

→ It involves identifying vulnerabilities, mitigating risks and ensuring that the application behaves securely under different scenarios.

→ Security begins with →



SAST →

Analyzes source code for vulnerabilities.

Tools → SonarQube, Fortify

DAST →

Test running apps for vulnerabilities

Tools → OWASP ZAP, Burp Suite

Penetration Testing

Simulates real world attacks

Tools → kali Linux.

2. Website Attacks & Content Security

Website phases threats such as injection attacks, session hijacking, phishing and Denial of Service (DOS).

3. Cross Site Scripting → ⭐⭐

XSS is a vulnerability where attackers inject malicious scripts into web pages viewed by users

→ Leading to data theft

→ session hijacking & unauthorized actions.

→ Most common cause - input validation.

Mitigations →

→ Encode output with HTML entities.

→ Use frameworks with auto escaping.

4. Content Security Policy →

→ A browser level Defence mechanism that restricts which resources (scripts, images, styles) can be loaded. helps mitigate,

→ Clickjacking

→ XSS

→ data injection,

by enforcing rules declared in HTTP headers or meta tags.

→ CSP allows developers to whitelist trusted domains.

Mitigation →

→ Define restrictive CSP rules

→ Avoid unsafe inline inscript rules.

5. CSRF (Cross Site Request Forgery)

→ These attacks tricks authenticated users into performing actions they did not intend by exploiting session cookies.

eg →

A malicious link could lead a logged in user to change their password without consent.

prevention involves →

→ using anti CSRF tokens

→ verifying request origins.

Mitigation

→ Use anti-CSRF tokens in forms

→ Set cookies with SameSite = Strict.

6. Clickjacking

→ It overlays transparent frames on UI elements over a legitimate webpage, tricking users into clicking hidden buttons.

Triggers →

unintended actions such as → webcam enabling transferring funds.

Primary defence →

X-Frame options HTTP header or frame-ancestors directive in CSP, which prevents embedding of the site in iframe.

Protection Technique	Implementation
X-Frame Options	DENY or SAMEORIGIN
CSP frame-ancestors	Restrict allowed domains
Frame sandboxing	Limit iframe permissions

Mitigation →

→ Add headers

→ Use frame-ancestors in CSP.

7. Iframe Protection ★

→ Iframes allow embedding external content but can be exploited for malicious purposes like phishing, clickjacking or loading malicious scripts.

→ For protection developers can set sandbox attributes (<iframe sandbox="allowscripts">)

→ use CSP rules

→ avoid untrusted resources.

Mitigation →

→ Use sandbox attributes in iframes.

Rest Unit-2 ChatGPT →

Unit-3

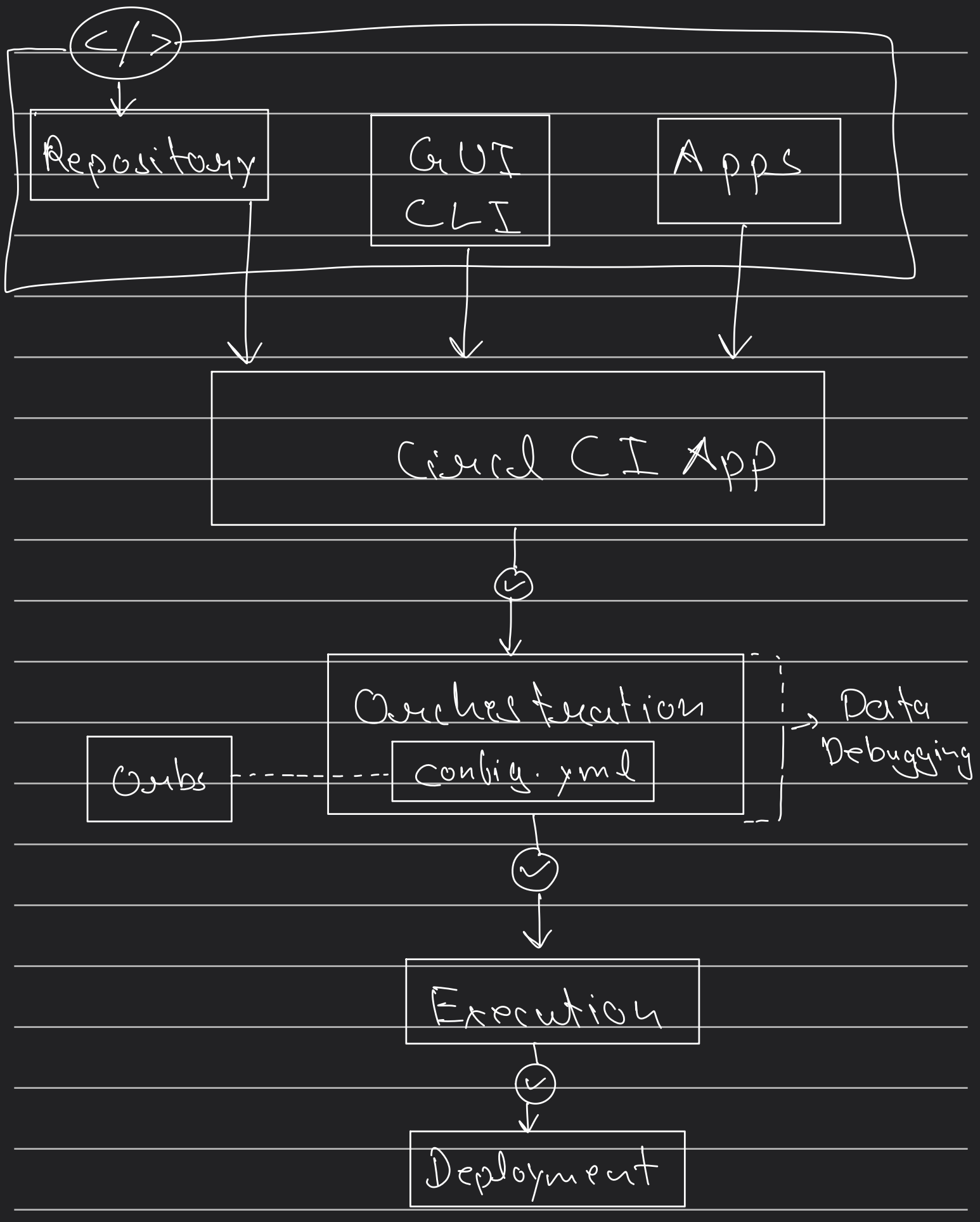
Circle CI ☆

→ CircleCI is a cloud based CI & CD platform that can be used to implement DevOps practices.
→ It allows developers to set up automated workflows that trigger on code changes, enabling them to integrate security across test & deploy for their applications.
→ helps streamline their CI/CD process.

Features →

→ Automated Builds
→ Testing Automation
→ CD
→ Customizable workflows
→ Cloud based Service.

Circle CI Architecture →



Circle CI provides 4 execution Environments →

→ Docker
→ Linux VM
→ MacOs
→ Windows

Orbs →

→ A reusable package of YAML configuration that condenses repeated pieces of config into a single line of code.

Benefits →

1) Save time in project configuration.
2) Simplify third-party integrations.

→ GitHub - Circle CI integration Security Best Practices.

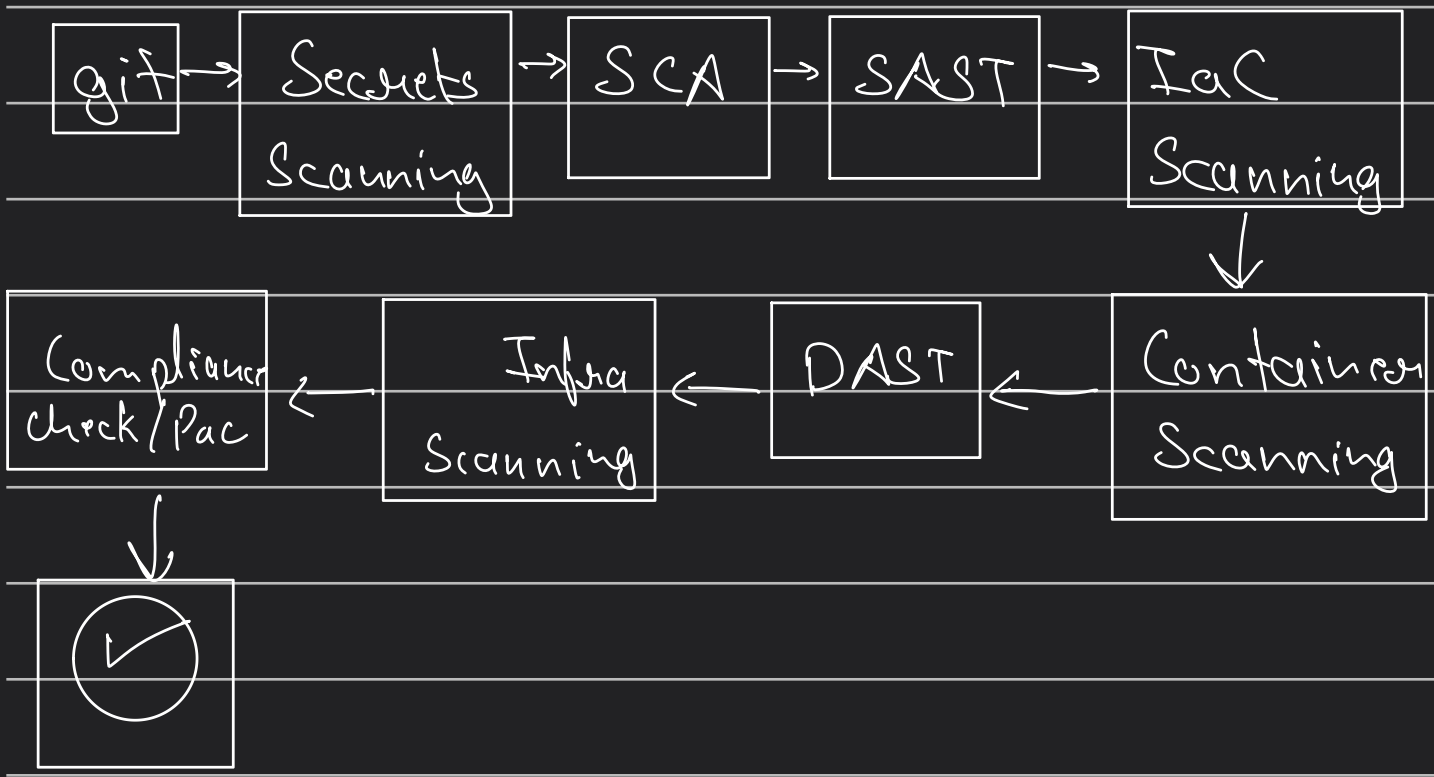
→ Use Secrets manager like vault.
→ Never hardcode secrets in config.yml.
→ Restrict GitHub Access.
→ Use GitHub Teams to manage access control.
→ Enable Required Status Checks.
→ Enable branch protection rules to require Circle CI checks before merging.
→ Use context for sensitive data.

→ DockerHub - CircleCI Integration Security Best Practices

→ Use PATs.
1. Limit scope of access.
2. Rotate Credentials regularly.
3. Avoid Hardcoding Secrets.
→ Use Secrets manager like vault.
→ Use private repositories wisely.
→ Audit and monitor Usage.

→ Use image signing & verification.

Integrating Tools to CI/CD pipeline.



Signing Git Commits & Tags → ★

Why? →

→ Ensure authenticity and Integrity.
→ Prevent unauthorized changes.
→ Build trust in collaborative environments.

Mechanisms →

→ GPG
→ SSH
→ S/MIME

→ Git Signing using SSH

1. Generate SSH key.
2. Add public key to Server or Git Service.
3. Sign in using private key.

→ Git Signing using S/MIME

1. Obtain S/MIME Certificate
2. Configure Git to use S/MIME
3. Sign a commit.

→ Signing Commits

→ git commit -S -m "Your message".

→ Verify

→ git log --show-signature.

→ Signing Tags

→ git tag -s v1.0 -m "Signed release"

→ Verify

git tag -v <tag-name>

→ Considerations during CI/CD integration in context to Git.

→ Enforce signed commits and tags in pipeline
eg:- CircleCI, GitHub Actions.
→ Security policies for signed commits & verification.
→ Limited access to code & build artifacts.
→ Secure key management
1. No secrets / keys in git.
2. Use dedicated keys for CI/CD systems.
3. Protect keys via using Secrets manager.
4. Limited access
5. Rotate keys periodically.
→ Audit & logging.

→ Security Requirements within CI/CD pipeline

1. Source Code Security
2. Secure Build Environment
3. Static & Dynamic Analysis
4. Secure Deployment Practices.
5. Logging & Monitoring.
6. Continuous patching & updates.

What is a Docker Registry? ☆

→ A storage and distribution system for Docker images.
→ Eg: Docker Hub, GitHub Container Registry, ECR, JFrog Artifactory.

Registry Architecture Components

→ Registry server
→ Storage backend
→ Authentication & authorization.
→ Notary

→ DevSecOps & Container Security

→ Containers are part of the software supply chain.
→ Risks: tampered images, outdated dependencies, vulnerabilities.
→ DevSecOps integrate security into pipelines.

→ What is image Scanning?

→ Cryptographic process to verify image authenticity & integrity.

→ Ensures image has not been tampered.

Tools for image scanning →

→ Docker Container Trust (DCT)

→ Uses Notary for signing & verification.

→ Cosign (by Sigstore): A tool for signing OCI images.

→ Notation: for signing and verifying artifacts.

AWS ACCESS CONTROL → Chatgpt.

DCT → ☆

→ Enable with DOCKER_CONTENT_TRUST=1

→ Automatically signs & verifies images.

→ Code:

1. export DOCKER_CONTENT_TRUST=1
2. docker push myimage Example: latest

Integrating Signing in CI/CD →

→ Add signing steps post-build in CI pipelines.

→ Verify before deployment

→ Use policies to enforce signed images only.

Content Trust Security Best Practices →

1. Always sign images before pushing.
2. Use private registries with Access Control.
3. Automate vulnerability Scanning.
4. Rotate signing keys regularly.

Unit-4

Risk Management → ☆

Risk management is the entire process of identifying, controlling and minimizing information system-related risks to a level that is acceptable and alligned with the value of the assets being protected.

Purpose→

→ Protect Organizational assets.
→ Ensure business continuity
→ support secure decision making.
→ Reduce financial & operational loss.
→ Enable organizations to achieve their mission securely.

Risk Formula

RISK = Threat x vulnerability x impact

CIA Triad → Already done Crypto.

Threat →

A threat is anything that has the potential to cause damage, loss or harm to information or system assets.

Common Organizational Threats →

→ Phishing attacks
→ Credential promise
→ Data leaks
→ insider threats
→ cloud misconfigurations
→ Denial of Service (DoS)

Thread Modelling →

1. Structured Approach to identify threats
2. Helps prioritize security controls
3. Used STRIDE

STRIDE	Threat Type	Violates
S	Spoofing	Authenticity
T	Tampering	Integrity
R	Repudiation	Non-Repudiation
I	Information Disclosure	Confidentiality
D	Denial of Service	Availibility
E	Elevation of Privilege	Authorization

STRIDE is used to systematically identify threats.

Impact of Risk →

Impact is the potential loss caused by a risk, usually measured in monetary terms.

Examples →

→ Financial Loss (€ 10 million data breach)
→ legal penalties

→ loss of customer trust
→ operational downtime
→ Reputational damage

Vulnerability →

A vulnerability is a weakness in a system that allows an attacker to reduce system's information assurance.

Eg →

→ weak password
→ misconfigured cloud storage
→ unpatched software
→ Exposed credentials in repositories.

CVSS (Common Vulnerability Scoring System)

used to measure vulnerability severity.

CVSS Metric Groups

1. Basic Metrics →

- → Attacker Vector
- → Attack Complexity
- → Privileges required
- → User interaction
- → Scope
- → impact (CIA)

2. Temporal Metrics →

- → exploit code maturity
- → Remediation level
- → Report confidence

3. Environmental Metrics →

- → Business impact requirements
- → Modified Base Metrics

Rapid Risk Assessment → ☆

Rapid Risk Assessment is a quick, structured approach to identify, evaluate and prioritize risk during early stages of a project deployment.

Steps →

1. Scope
   → what is being assessed.

2. Context Gathering
   → Data handled
   → Technology stack
   → Deployment environment

3. Threat Identification
   → what could go wrong?
   → Use STRIDE framework.

4. Risk Scoring
   → Likelihood x impact

5. Recommendations
   → Actions and timelines.

→ RRA Template

1. project information
2. Application & data context
3. Risk identification & Evaluation.
4. Risk Register
5. Monitoring & improvment
6. Final remarks.

Risk Register →

A risk register is a structured document used to record, track & monitor risks over time.

Field	Description
Risk	Identified risk
Cause	Root cause
Impact	Financial / operational loss

Likelihood	Probability of Occurance
Outcome	Potential Result
Risk Level	Low / Med. / high
Type	Technical / Organizational
Mitigation Plan	Security controls
Stats	Open / in-progress / closed.

Eg:

Risk → Credential compromise

Cause → phishing mail

Impact → ₹10 million loss

Risk level → Medium

Status → Open

DREAD → Risk Assessment Model ★

D → Damage Potential
How severe would the impact be if the vulnerability is exploited?

R → Reproducibility
How easily can the attack be repeated?

E → Exploitability
How easy is it for an attacker to exploit the vulnerability?

A → Affected Users
How many users would be impacted by the attack?

D → Discoverability
How easy it is for an attacker to find the vulnerability?

Risk Calculation using DREAD Model →

STEP-1

Assign scores (0-10 scale)

D → if exploited, attacker gains full access → Score → 9

R → Easy to repeat, once discovered Score → 8

E → Requires Minimal efforts → Score → 9

A → All customers impacted → Score → 10

D → Config files are in Repo → Score → 8

STEP-2 → Calculate Risk

→ Add all scores

$9+8+9+10+8 = 44$

→ Avg. Score = $44/5 = 8.8$ (High Risk)

STEP-3 → Assign Priority & Recommend Mitigation

Continuous Risk Monitoring → ☆

Risk management is not one time, it requires continuous monitoring.

Monitoring Areas →

→ infrastructure as cloud posture
→ Network traffic
→ Application behavior
→ identity and access
→ CI/CD pipelines

Security Tools →

→ SIEM (Splunk, ELK)
→ CSPM (Aws Security Hub)
→ Runtime protection (Falco)
→ WAF (Web Application Firewall)
→ Vulnerability Scanners (Trivy, Snyk)

Unit - 5

1. Testing Security & Maintaining Security

Testing security refers to the continuous process of identifying vulnerabilities, validating defences, and ensuring systems remain secure over time, not just at release time.

Why Security must be maintained Continuously?

→ New vulnerabilities appear regularly.
→ Software changes includes new risks.
→ threat landscape evolves continuously.

Continuous Security Maintenance Activities →

→ **Patch Management**
Regularly updating software to fix vulnerabilities.

→ **Security Monitoring**
Continuous monitoring for threats and anomalies.

→ **Configuration Management**
Ensuring secure system configuration.

→ **Change Management**
controlling and reviewing system changes.

Ensures CIA triad is preserved throughout system lifecycle.

Internal Security Auditing →

A Systematic review of applications, infrastructure and services to ensure. security controls are correctly implemented and followed.

Objectives of Internal auditing →

→ identify security gaps
→ Ensure compliance with standards
→ Maintain a security baseline
→ improve incident readiness.

key activities →

→ Regularly scheduled security audits.
→ Establishing security baselines and compliance standards.
→ using automated tools for monitoring and logging.
→ implementing secrets management and IAM best practices.
→ Can be compliance driven or Security driven.

Advanced Security Testing →

Red Teams →

Simulate real world attackers to test:

1. Organization's defences.
2. Detection capabilities
3. incident response readiness.

Characteristics →

→ long term, stealthy attacks.
→ Mimic real threat attackers
→ Test people process, technology.

Purpose →

→ Measure how well an organization can detect and respond to sophisticated attacks.
→ improve security maturity.

External Penetration Testing →

It involves hiring third party security experts to legally and safely attack systems to find vulnerabilities.

Characteristics →

→ Time boxed assessments.
→ Scope-defined attacks
→ focused on vulnerabilities discovery.

Difference Between Red Team & Pen Testing

Aspect	Red Team	Pen Testing
Duration	Long-term	Short-term
Goal	Test defenses & response	Find vulnerabilities
Approach	Stealthy	Direct
Focus	Detection & reaction	Security gaps

Bug Bounty Programs →

Bug Bounty Programs incentivize external security researchers to find and responsibly disclose vulnerabilities.

Types →

→ Public Programs → Open to Everyone
→ Private Programs → invite-only researchers

Key Components →

→ Clear rules of engagement
→ Defined scope
→ Reward structure
→ Responsible disclosure process.

Benefits →

→ Access to global security talent
→ Cost-effective vulnerability discovery
→ Complements internal security efforts
→ Improves real-world security coverage.

Continuous Security → Practice & Repetition

The 10,000 hours rule states that mastery in any field requires extensive practice.
In cybersecurity continuous learning and hand-on-practice are essential.

Why?

→ Attack techniques evolve
→ Tools and technologies change rapidly.
→ Static knowledge become outdated.

Practical method →

→ Hands on projects
→ Security research
→ Capture the flag (CTF) competitions.
→ Certifications and training
→ Real-world-incident handling.

DevSecOps Philosophy →

Security must be →

→ Built-in, not bolted-on

→ Automated

→ Integrated into CI/CD pipelines.

Security Testing in DevSecOps →

1. SAST
2. DAST
3. Software Composition Analysis (SCA)
4. Container Security Scanning.
5. IaC Scanning
6. Cloud & Application posture scanning.

Incident Response Planning →

1. Preparation
2. Detection & Analysis
3. Containment & Eradication.
4. Recovery.
5. Lessons Learned.

Disaster Recovery & Business Continuity →

Key Metrics →

→ RTO → Recovery Time Objective
Maximum Acceptable Downtime

→ RPO → Recovery Point Objective
Maximum acceptable data loss.

Preparation ensures →

→ faster recovery
→ Reduced business impacts
→ Regulatory compliance

Security Culture

Security is not a one time activity; it is continuous organizational practice.

Key elements →

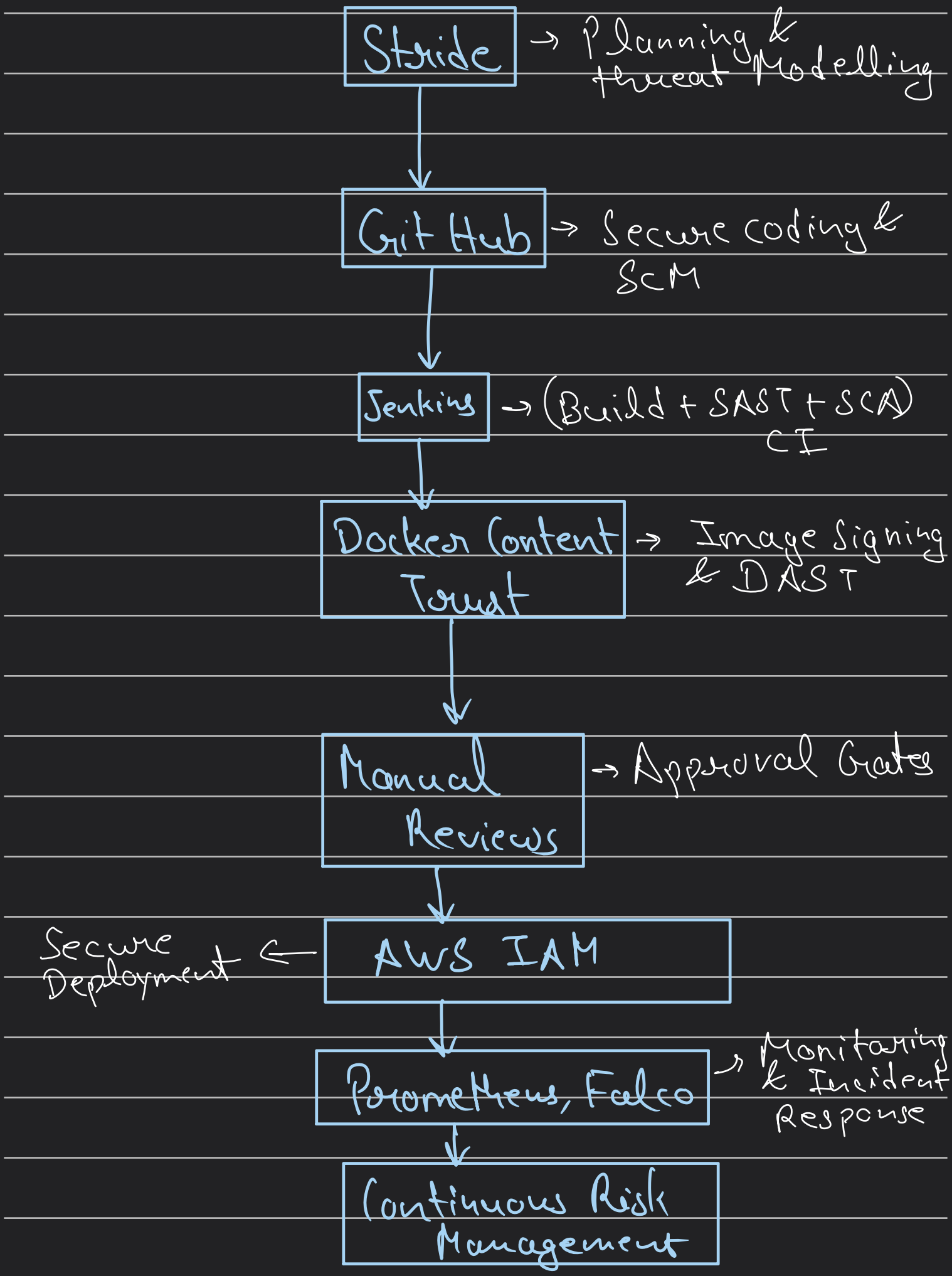
→ Security mindset accross all teams
→ Continuous training
→ Threat Modelling
→ Incident Response drills.

→ Security should enable speed, not block it.
→ Automation over manual controls
→ shared responsibility model.

Security Metrics to Track →

→ MTTD
→ MTTR
→ Vulnerability aging
→ Patching Rate
→ Security Test Coverage
→ False positive rate
→ Critical vulnerability backlog.

Pipeline →


Stride
→ Planning & threat Modelling
GitHub
→ Secure coding & SCM
Jenkins
→ (Build + SAST + SCA) CI
Docker Content Trust
→ Image Signing & DAST
Manual Reviews
→ Approval Gates
Secure Deployment ←
AWS IAM
Prometheus, Falco
→ Monitoring & Incident Response
Continuous Risk Management