DEVSECOPS COURSE

FUNDAMENTAL KNOWLEDGE

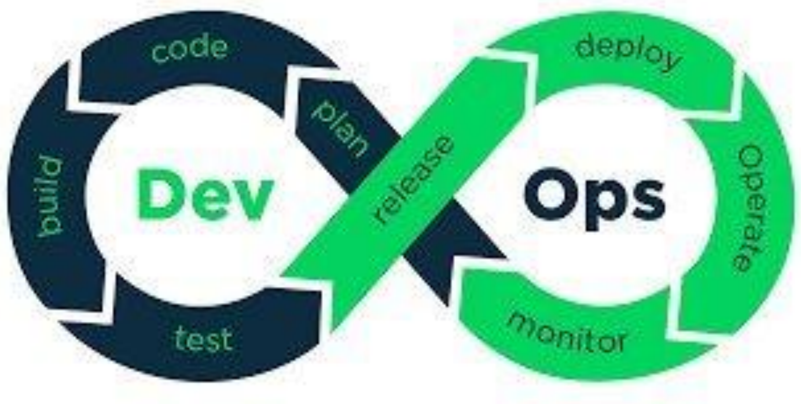
TRAINER: TRAN HUU HOA

AGENDA

- DevOps concepts
- Benefits
- Principles
- Culture
- Features
- DevSecOps concepts
- Toolchains

DEVOPS CONCEPTS

DevOps is the union of people, process, and technology to continually provide value to customers. It bridges the gap between development and operations, enabling better collaboration and more reliable higher velocity product delivery



BENEFITS

- **Velocity**
  - Accelerating time to market
  - Improving the mean time to recovery
- **Reliability**
  - Adapting to the market and competition
  - Maintaining system stability and reliability

PRINCIPLES

Collaboration

Collaboration fosters a culture of shared responsibility and collective ownership, breaking down silos between teams. It promotes transparency, knowledge sharing, and a unified approach towards common objectives.

Automation

Automation streamlines repetitive tasks, reduces human error, and ensures consistency in the software delivery pipeline. It enables rapid iterations, accelerates feedback loops, and promotes reliability.

Monitoring

Monitoring provides real-time insights into system performance, user experience, and operational metrics. It enables proactive problem detection, timely response to incidents, and continuous optimization of processes.

CULTURE

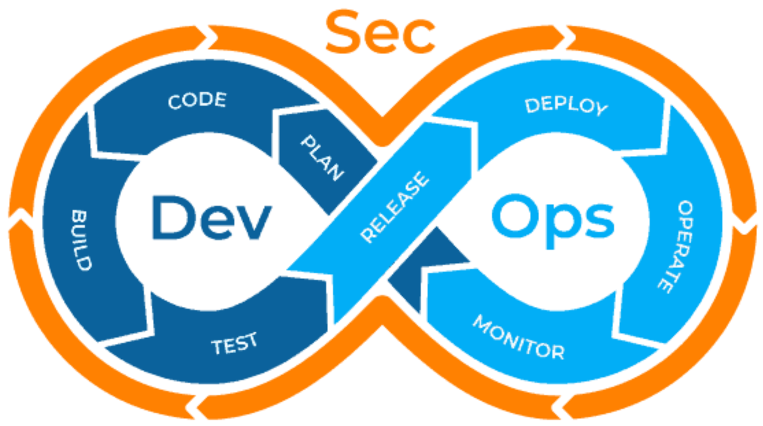
- Collaboration, visibility, and alignment
- Shifts in scope and accountability
- Shorter release cycles
- Continuous Learning and Improvement
- Autonomous Teams
- Shared Responsibility

FEATURES

- Collaboration between all stakeholder
- Agile practices
- Version control
- Infrastructure as code
- Configuration and secret management
- Continuous Integration
- Continuous Deployment
- Automation as much as possible

DEVSECOPS CONCEPTS

DevSecOps takes DevOps a step further by integrating and automating the enforcement of preventive, detective, and responsive security controls. The main purpose of DevSecOps is to provide premium security while also applying faster speed of process, accessibility, and scalability



DEVSECOPS EXTRA PRINCIPLES

Shift-Left Security: Integrate security early in development

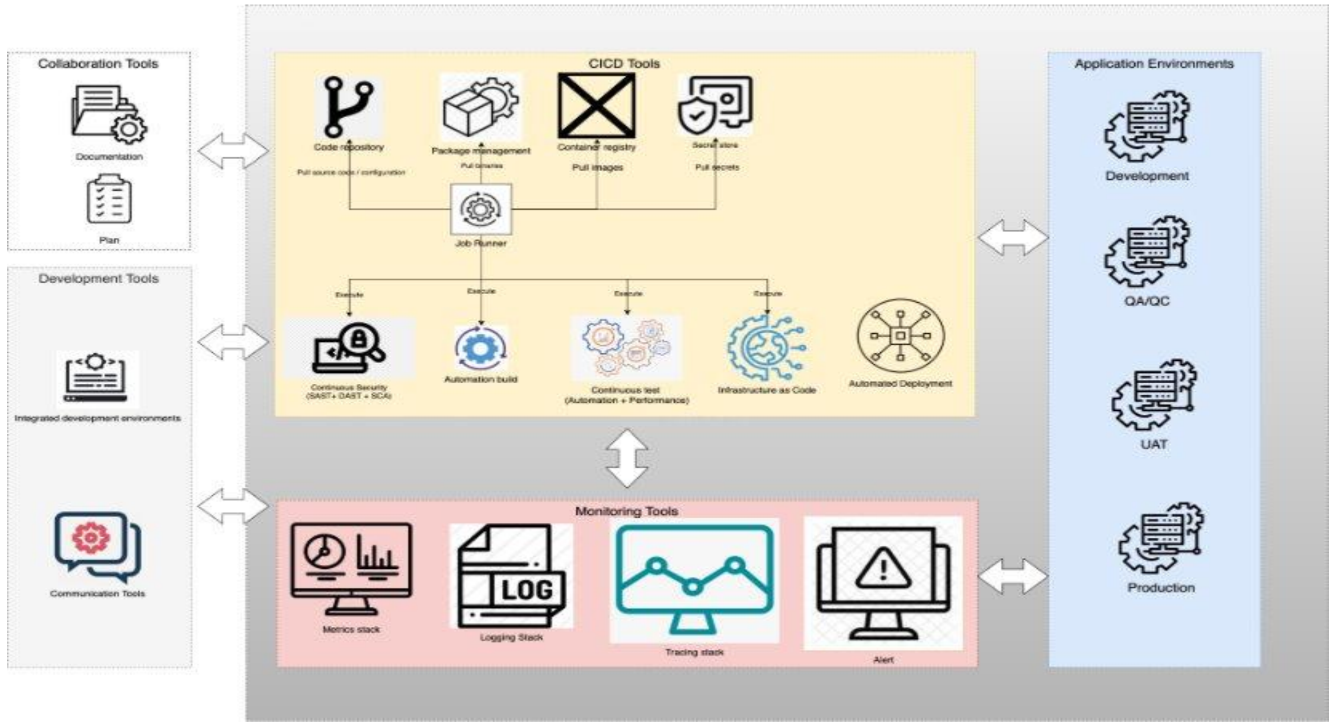
- Security as Code: Automate security into infrastructure and application code.
- Continuous Security: Embed security checks into the CI/CD pipeline.
- Collaboration & Culture: Break down silos between Dev, Sec, and Ops.
- Immutable Infrastructure: Use containers, IaC, and automated provisioning.
- Least Privilege and Zero Trust: Enforce tight access controls and trust no default access.

DEVSECOPS LIFECYCLE

Stage	Security Integration
Plan	Threat modeling(STRIDE, PASTA ...), security requirements
Develop	Secure coding, Static Application Security Testing, Secrets Detection
Build	Dependency scanning (SCA), license checks, signed SBOM (Software Bill of Materials)
Test	DAST (Dynamic Analysis),API Security Testing, fuzzing, Infrastructure as Code (IaC) scanning
Release	Container/image security scanning, Hardened Release Artifacts
Deploy	Runtime Security, RASP (Runtime Application Self-Protection), infrastructure compliance, Configuration Hardening
Operate	Continuous monitoring(metrics, logs, tracing, alert), SIEM integration
Monitor	Incident response, anomaly detection, Digital Forensics

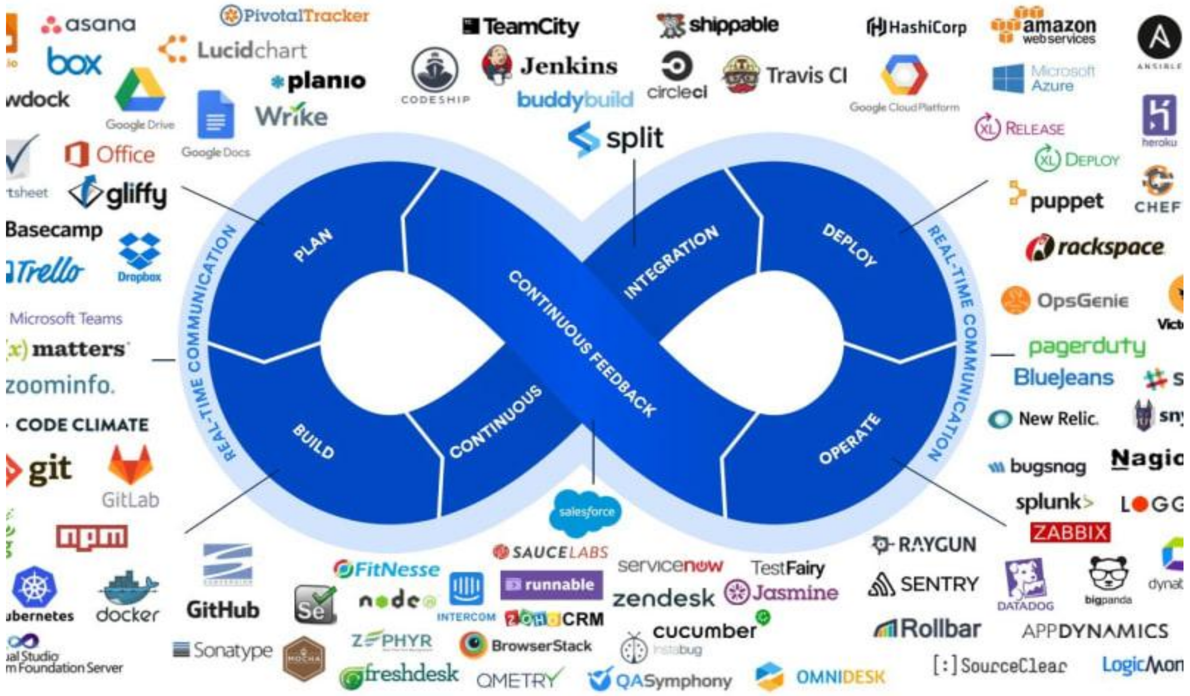
TOOLCHAINS

Overview



TOOLCHAINS

Samples



TOOLCHAINS

DevSecOps

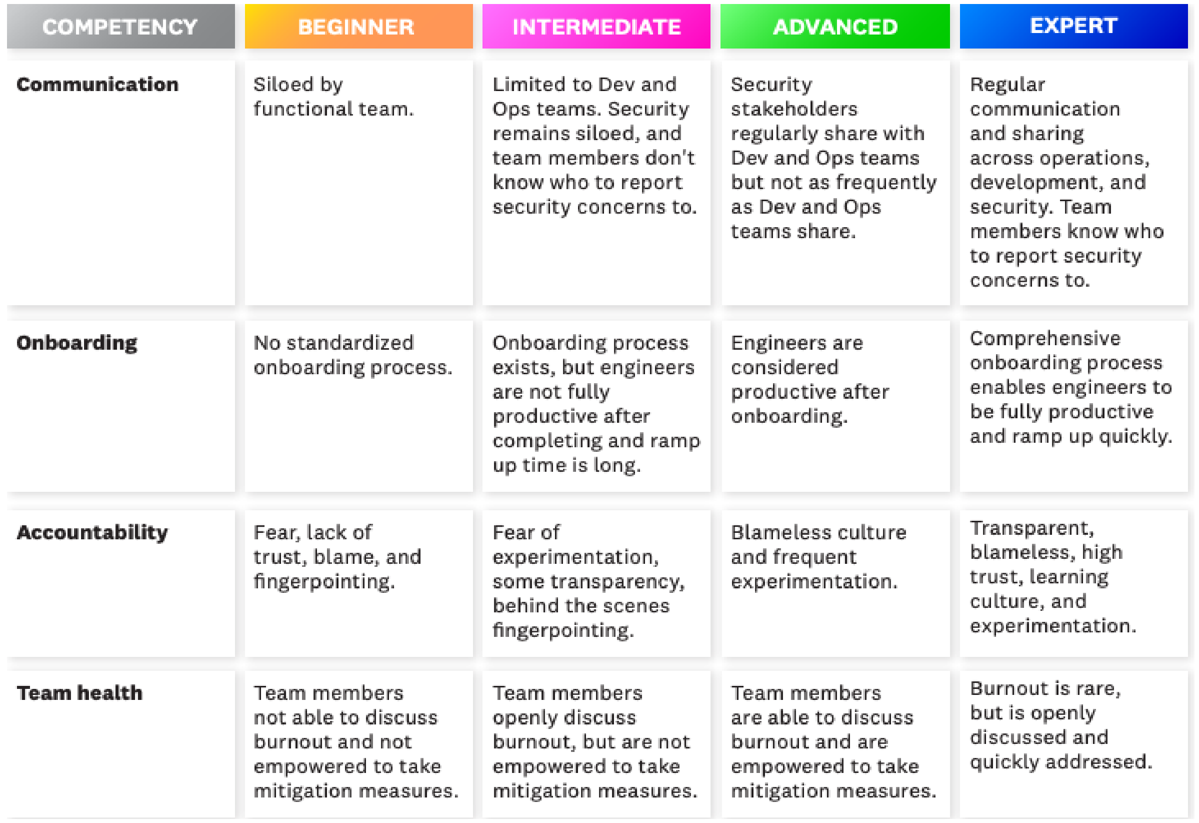
Stage	Tools / Techniques
1. Plan	OWASP Threat Dragon, Microsoft Threat Modeling, Tool Jira (with security plugins) ...
2. Develop	SonarQube, GitGuardian, ESLint (security plugins) ...
3. Build	Snyk, OWASP Dependency-Check , Syft,CycloneDX ...
4. Test	OWASP ZAP, Burp Suite, Postman Security Tests,Checkov, Jazzer ...
5. Release	Trivy, Notary (Docker Content Trust) Cosign (Sigstore) ...
6. Deploy	Falco, InSpec, Open Policy Agent (OPA), Terraform Sentinel ...
7. Operate	Splunk, Prometheus + Grafana ...
8. Monitor	CrowdStrike, Suricata,PagerDuty ...

MATURITY MODEL

- Culture
- Plan & Develop
- Build & Test
- Release & Deploy
- Operate
- Observe & Respond

MATURITY MODEL

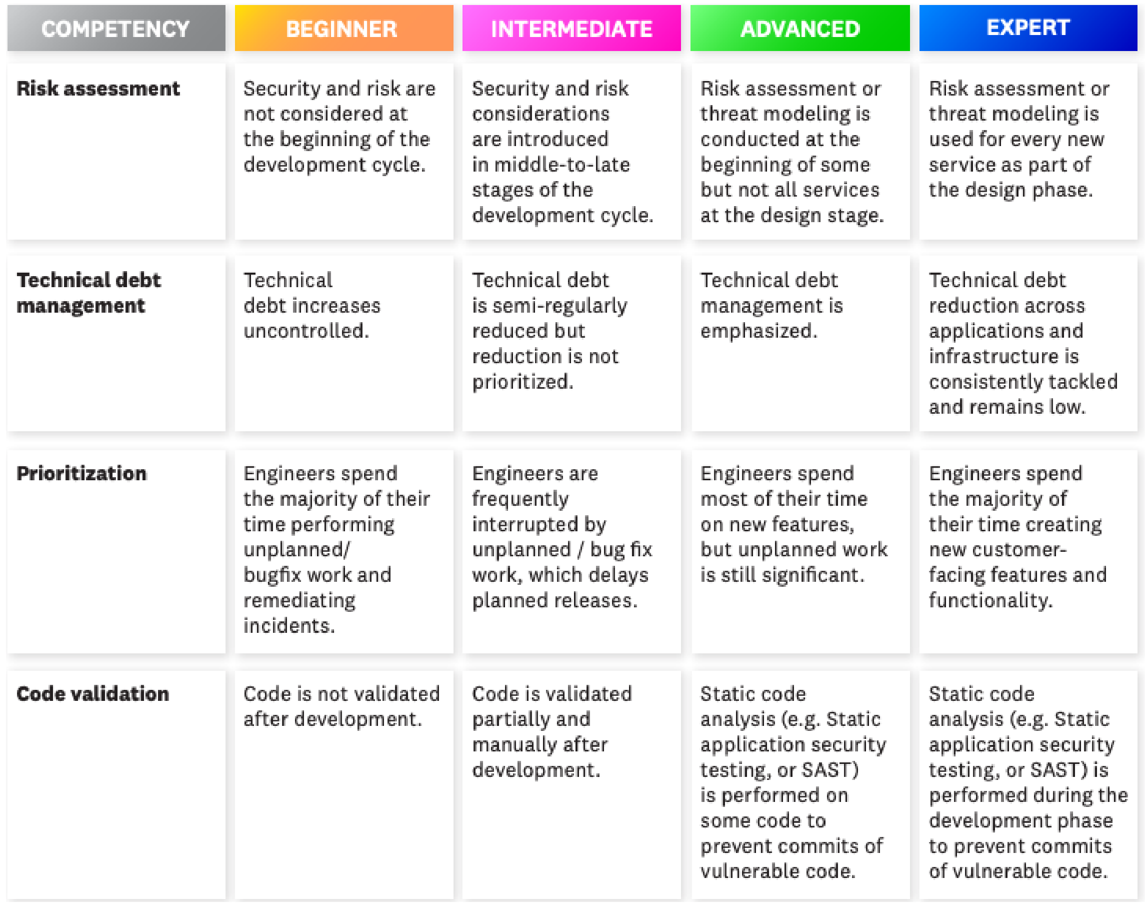
Culture



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Communication	Siloed by functional team.	Limited to Dev and Ops teams. Security remains siloed, and team members don't know who to report security concerns to.	Security stakeholders regularly share with Dev and Ops teams but not as frequently as Dev and Ops teams share.	Regular communication and sharing across operations, development, and security. Team members know who to report security concerns to.
Onboarding	No standardized onboarding process.	Onboarding process exists, but engineers are not fully productive after completing and ramp up time is long.	Engineers are considered productive after onboarding.	Comprehensive onboarding process enables engineers to be fully productive and ramp up quickly.
Accountability	Fear, lack of trust, blame, and fingerpointing.	Fear of experimentation, some transparency, behind the scenes fingerpointing.	Blameless culture and frequent experimentation.	Transparent, blameless, high trust, learning culture, and experimentation.
Team health	Team members not able to discuss burnout and not empowered to take mitigation measures.	Team members openly discuss burnout, but are not empowered to take mitigation measures.	Team members are able to discuss burnout and are empowered to take mitigation measures.	Burnout is rare, but is openly discussed and quickly addressed.

MATURITY MODEL

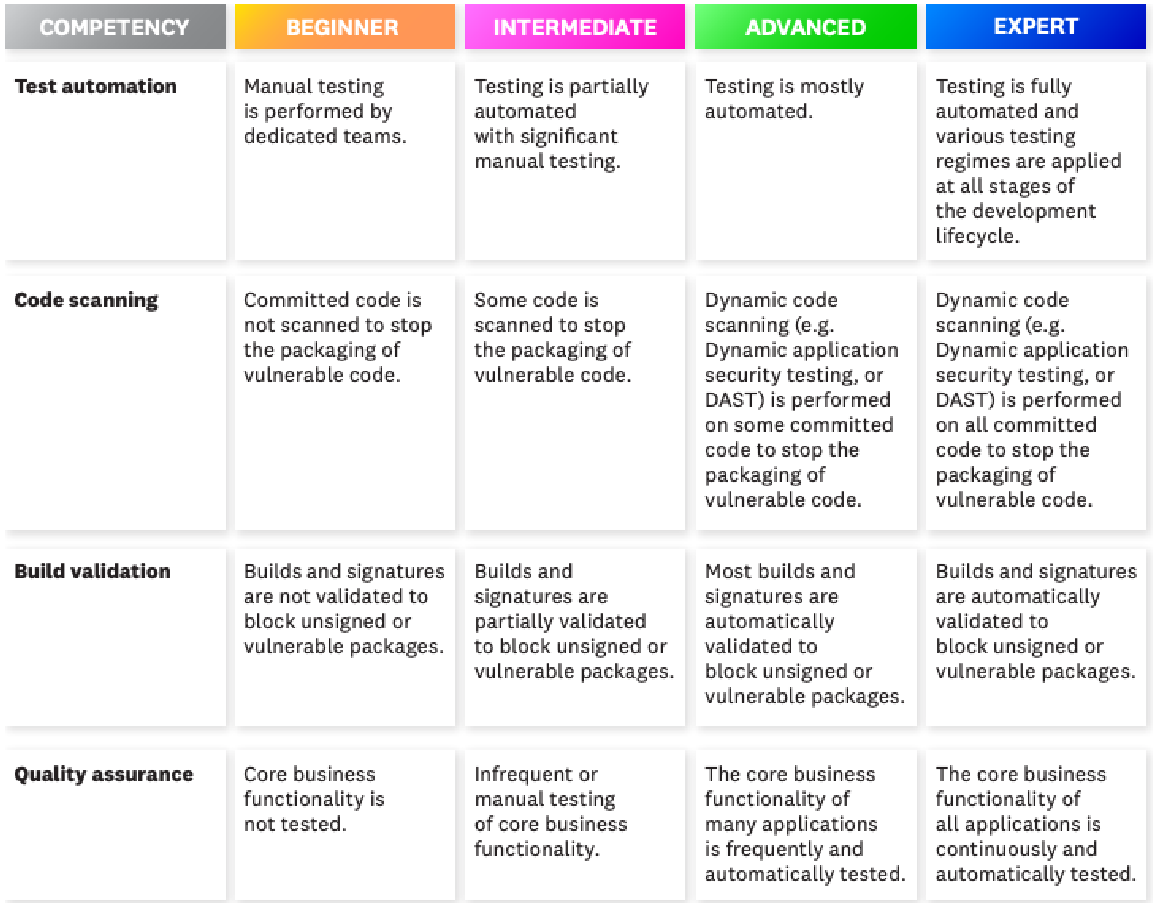
Plan & Develop



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Risk assessment	Security and risk are not considered at the beginning of the development cycle.	Security and risk considerations are introduced in middle-to-late stages of the development cycle.	Risk assessment or threat modeling is conducted at the beginning of some but not all services at the design stage.	Risk assessment or threat modeling is used for every new service as part of the design phase.
Technical debt management	Technical debt increases uncontrolled.	Technical debt is semi-regularly reduced but reduction is not prioritized.	Technical debt management is emphasized.	Technical debt reduction across applications and infrastructure is consistently tackled and remains low.
Prioritization	Engineers spend the majority of their time performing unplanned/ bugfix work and remediating incidents.	Engineers are frequently interrupted by unplanned / bug fix work, which delays planned releases.	Engineers spend most of their time on new features, but unplanned work is still significant.	Engineers spend the majority of their time creating new customer-facing features and functionality.
Code validation	Code is not validated after development.	Code is validated partially and manually after development.	Static code analysis (e.g. Static application security testing, or SAST) is performed on some code to prevent commits of vulnerable code.	Static code analysis (e.g. Static application security testing, or SAST) is performed during the development phase to prevent commits of vulnerable code.

MATURITY MODEL

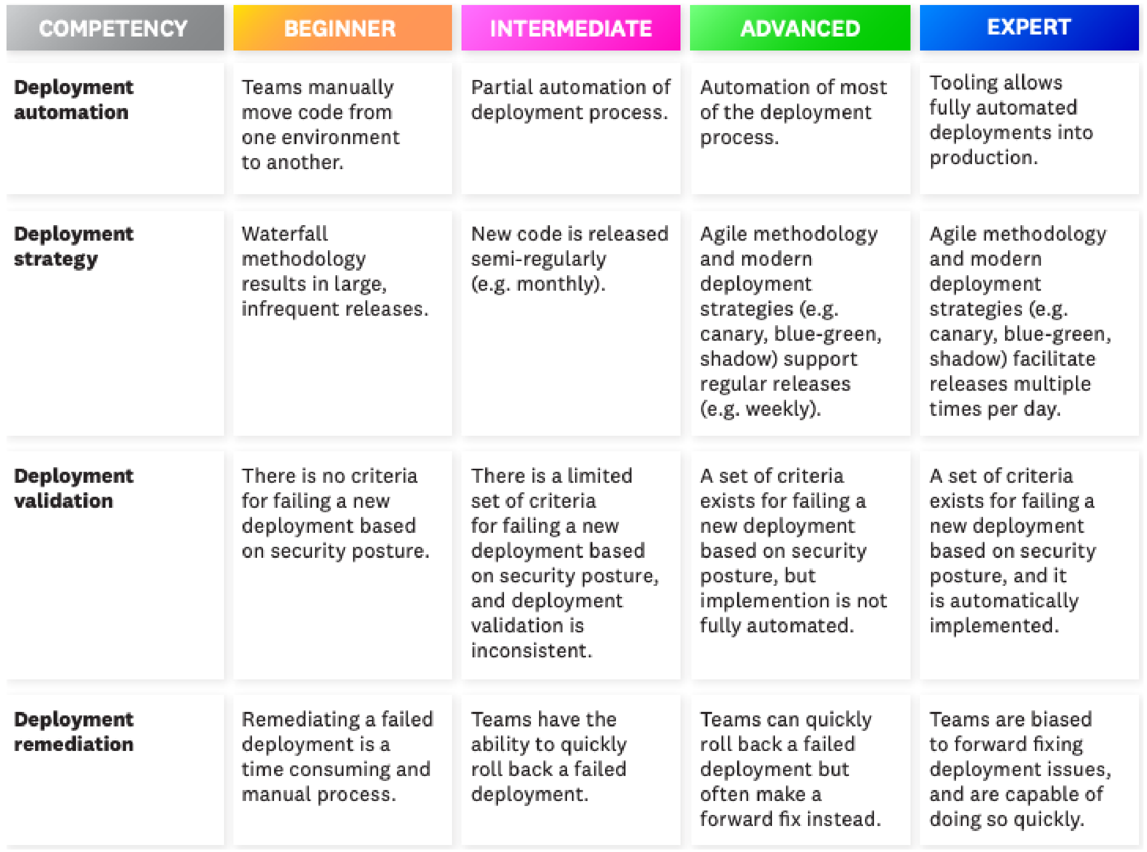
Build & Test



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Test automation	Manual testing is performed by dedicated teams.	Testing is partially automated with significant manual testing.	Testing is mostly automated.	Testing is fully automated and various testing regimes are applied at all stages of the development lifecycle.
Code scanning	Committed code is not scanned to stop the packaging of vulnerable code.	Some code is scanned to stop the packaging of vulnerable code.	Dynamic code scanning (e.g. Dynamic application security testing, or DAST) is performed on some committed code to stop the packaging of vulnerable code.	Dynamic code scanning (e.g. Dynamic application security testing, or DAST) is performed on all committed code to stop the packaging of vulnerable code.
Build validation	Builds and signatures are not validated to block unsigned or vulnerable packages.	Builds and signatures are partially validated to block unsigned or vulnerable packages.	Most builds and signatures are automatically validated to block unsigned or vulnerable packages.	Builds and signatures are automatically validated to block unsigned or vulnerable packages.
Quality assurance	Core business functionality is not tested.	Infrequent or manual testing of core business functionality.	The core business functionality of many applications is frequently and automatically tested.	The core business functionality of all applications is continuously and automatically tested.

MATURITY MODEL

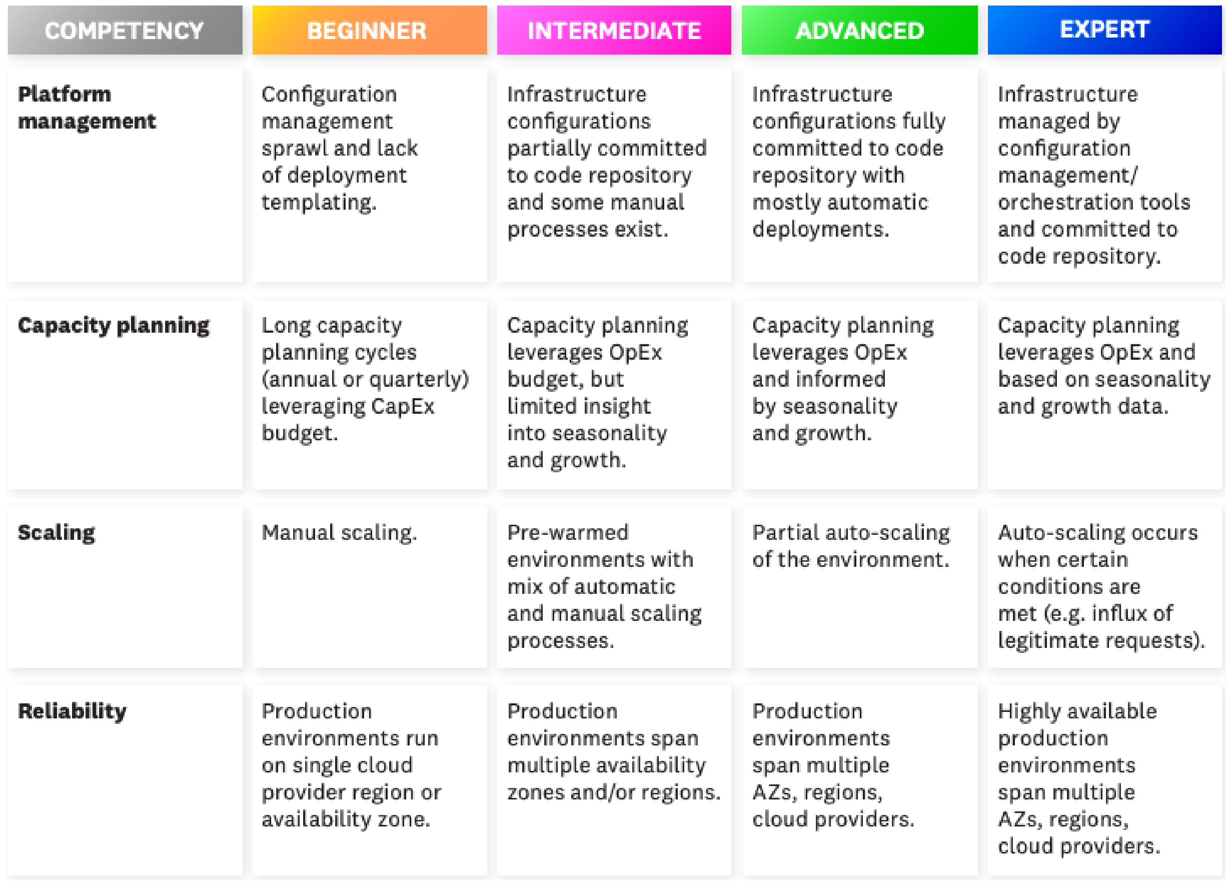
Release & Deploy



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Deployment automation	Teams manually move code from one environment to another.	Partial automation of deployment process.	Automation of most of the deployment process.	Tooling allows fully automated deployments into production.
Deployment strategy	Waterfall methodology results in large, infrequent releases.	New code is released semi-regularly (e.g. monthly).	Agile methodology and modern deployment strategies (e.g. canary, blue-green, shadow) support regular releases (e.g. weekly).	Agile methodology and modern deployment strategies (e.g. canary, blue-green, shadow) facilitate releases multiple times per day.
Deployment validation	There is no criteria for failing a new deployment based on security posture.	There is a limited set of criteria for failing a new deployment based on security posture, and deployment validation is inconsistent.	A set of criteria exists for failing a new deployment based on security posture, but implemention is not fully automated.	A set of criteria exists for failing a new deployment based on security posture, and it is automatically implemented.
Deployment remediation	Remediating a failed deployment is a time consuming and manual process.	Teams have the ability to quickly roll back a failed deployment.	Teams can quickly roll back a failed deployment but often make a forward fix instead.	Teams are biased to forward fixing deployment issues, and are capable of doing so quickly.

MATURITY MODEL

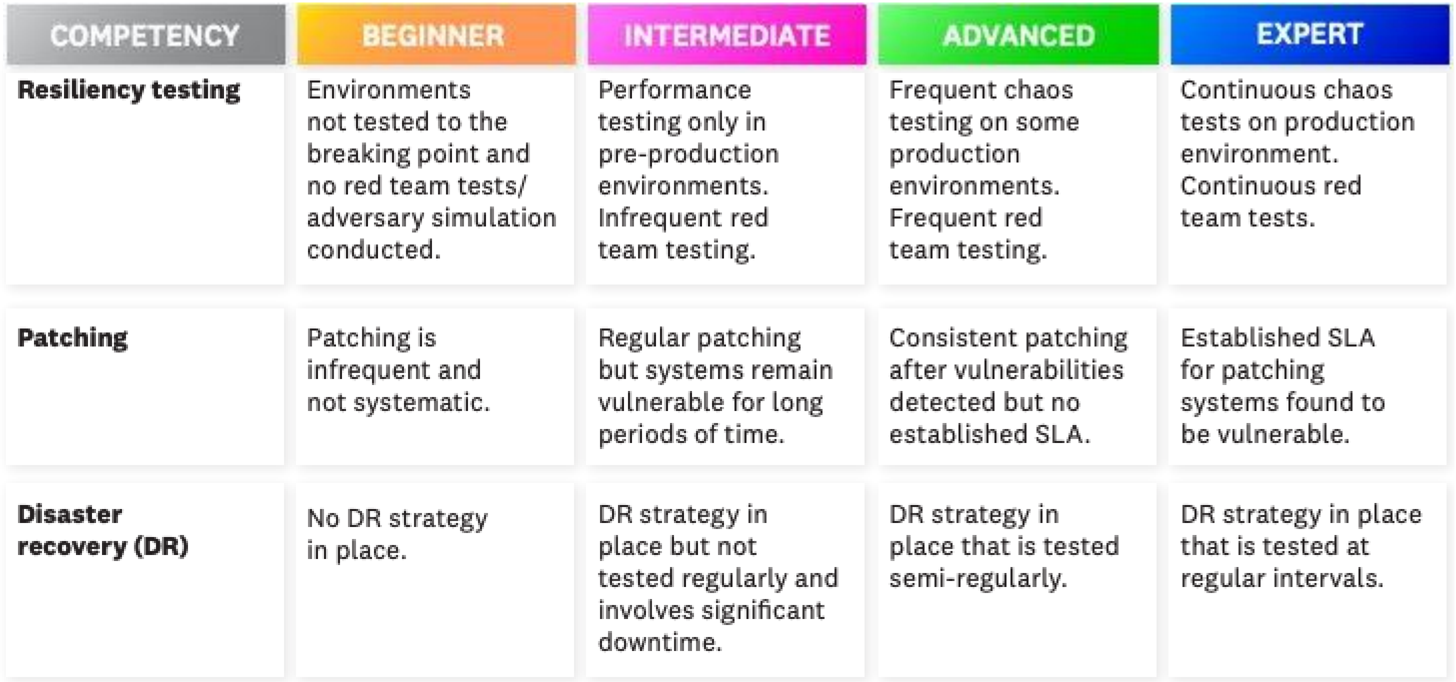
Operate



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Platform management	Configuration management sprawl and lack of deployment templating.	Infrastructure configurations partially committed to code repository and some manual processes exist.	Infrastructure configurations fully committed to code repository with mostly automatic deployments.	Infrastructure managed by configuration management/ orchestration tools and committed to code repository.
Capacity planning	Long capacity planning cycles (annual or quarterly) leveraging CapEx budget.	Capacity planning leverages OpEx budget, but limited insight into seasonality and growth.	Capacity planning leverages OpEx and informed by seasonality and growth.	Capacity planning leverages OpEx and based on seasonality and growth data.
Scaling	Manual scaling.	Pre-warmed environments with mix of automatic and manual scaling processes.	Partial auto-scaling of the environment.	Auto-scaling occurs when certain conditions are met (e.g. influx of legitimate requests).
Reliability	Production environments run on single cloud provider region or availability zone.	Production environments span multiple availability zones and/or regions.	Production environments span multiple AZs, regions, cloud providers.	Highly available production environments span multiple AZs, regions, cloud providers.

MATURITY MODEL

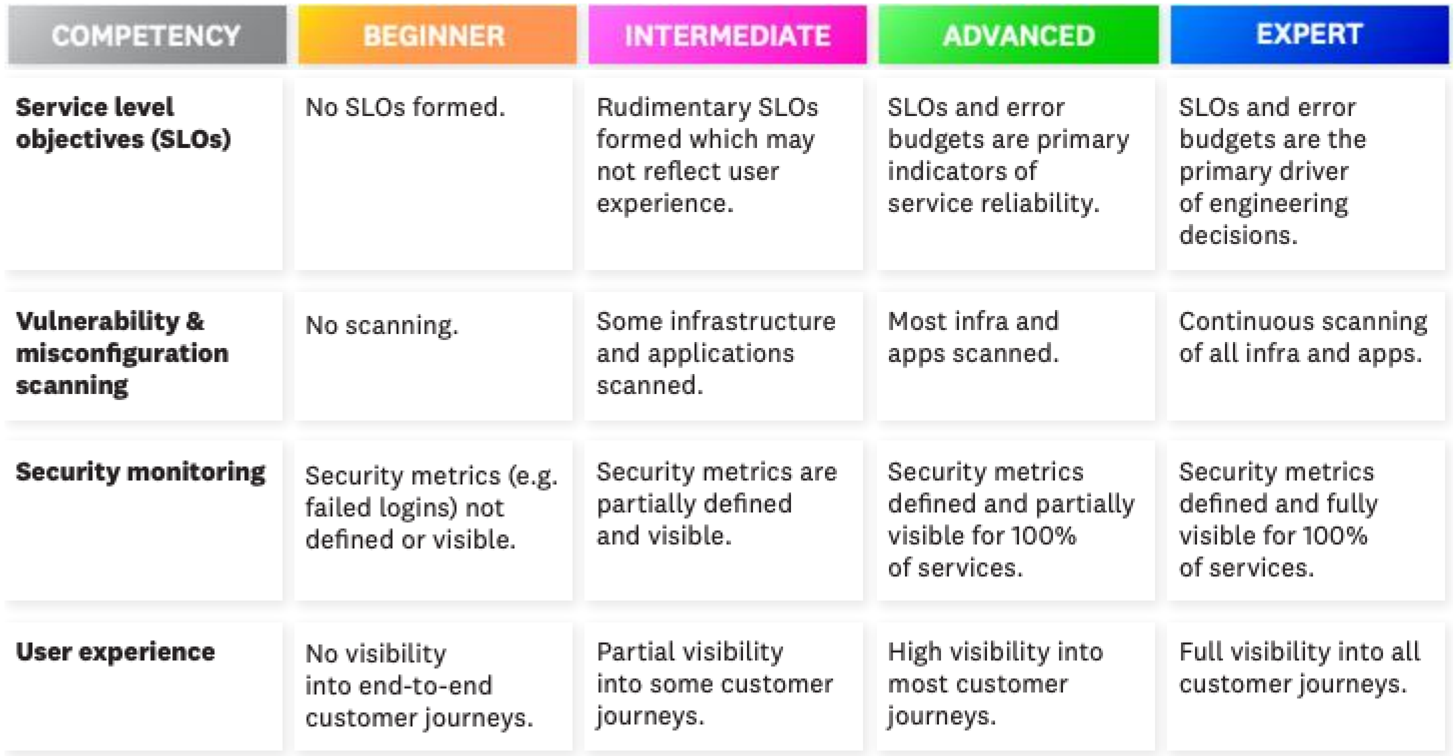
Operate



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Resiliency testing	Environments not tested to the breaking point and no red team tests/ adversary simulation conducted.	Performance testing only in pre-production environments. Infrequent red team testing.	Frequent chaos testing on some production environments. Frequent red team testing.	Continuous chaos tests on production environment. Continuous red team tests.
Patching	Patching is infrequent and not systematic.	Regular patching but systems remain vulnerable for long periods of time.	Consistent patching after vulnerabilities detected but no established SLA.	Established SLA for patching systems found to be vulnerable.
Disaster recovery (DR)	No DR strategy in place.	DR strategy in place but not tested regularly and involves significant downtime.	DR strategy in place that is tested semi-regularly.	DR strategy in place that is tested at regular intervals.

MATURITY MODEL

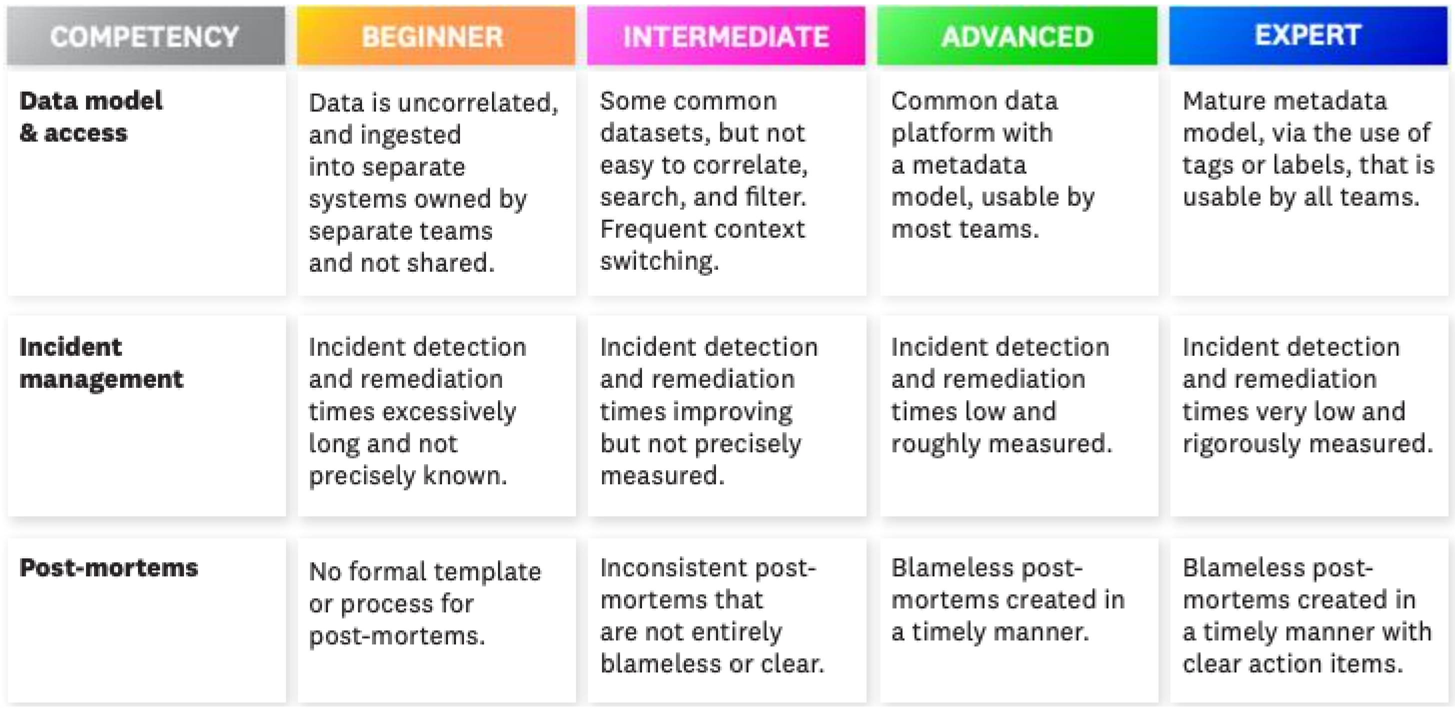
Observe & Respond



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Service level objectives (SLOs)	No SLOs formed.	Rudimentary SLOs formed which may not reflect user experience.	SLOs and error budgets are primary indicators of service reliability.	SLOs and error budgets are the primary driver of engineering decisions.
Vulnerability & misconfiguration scanning	No scanning.	Some infrastructure and applications scanned.	Most infra and apps scanned.	Continuous scanning of all infra and apps.
Security monitoring	Security metrics (e.g. failed logins) not defined or visible.	Security metrics are partially defined and visible.	Security metrics defined and partially visible for 100% of services.	Security metrics defined and fully visible for 100% of services.
User experience	No visibility into end-to-end customer journeys.	Partial visibility into some customer journeys.	High visibility into most customer journeys.	Full visibility into all customer journeys.

MATURITY MODEL

Observe & Respond



COMPETENCY	BEGINNER	INTERMEDIATE	ADVANCED	EXPERT
Data model & access	Data is uncorrelated, and ingested into separate systems owned by separate teams and not shared.	Some common datasets, but not easy to correlate, search, and filter. Frequent context switching.	Common data platform with a metadata model, usable by most teams.	Mature metadata model, via the use of tags or labels, that is usable by all teams.
Incident management	Incident detection and remediation times excessively long and not precisely known.	Incident detection and remediation times improving but not precisely measured.	Incident detection and remediation times low and roughly measured.	Incident detection and remediation times very low and rigorously measured.
Post-mortems	No formal template or process for post-mortems.	Inconsistent post-mortems that are not entirely blameless or clear.	Blameless post-mortems created in a timely manner.	Blameless post-mortems created in a timely manner with clear action items.