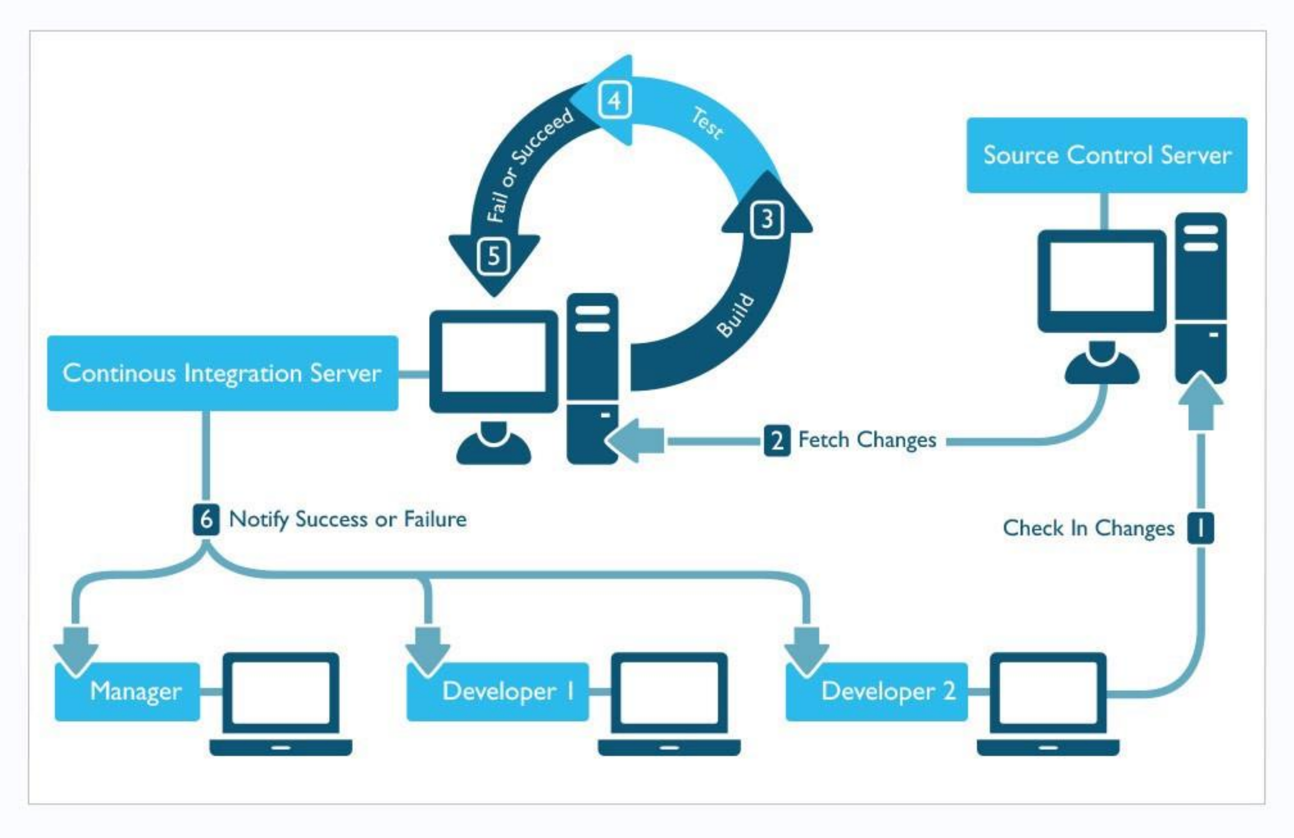
DEVSECOPS

TEXAS SKILLS DEVELOPMENT FUND
TRAINING PARTNERSHIP

AUSTIN
COMMUNITY
COLLEGE
DISTRICT

Continuous Integration (CI)

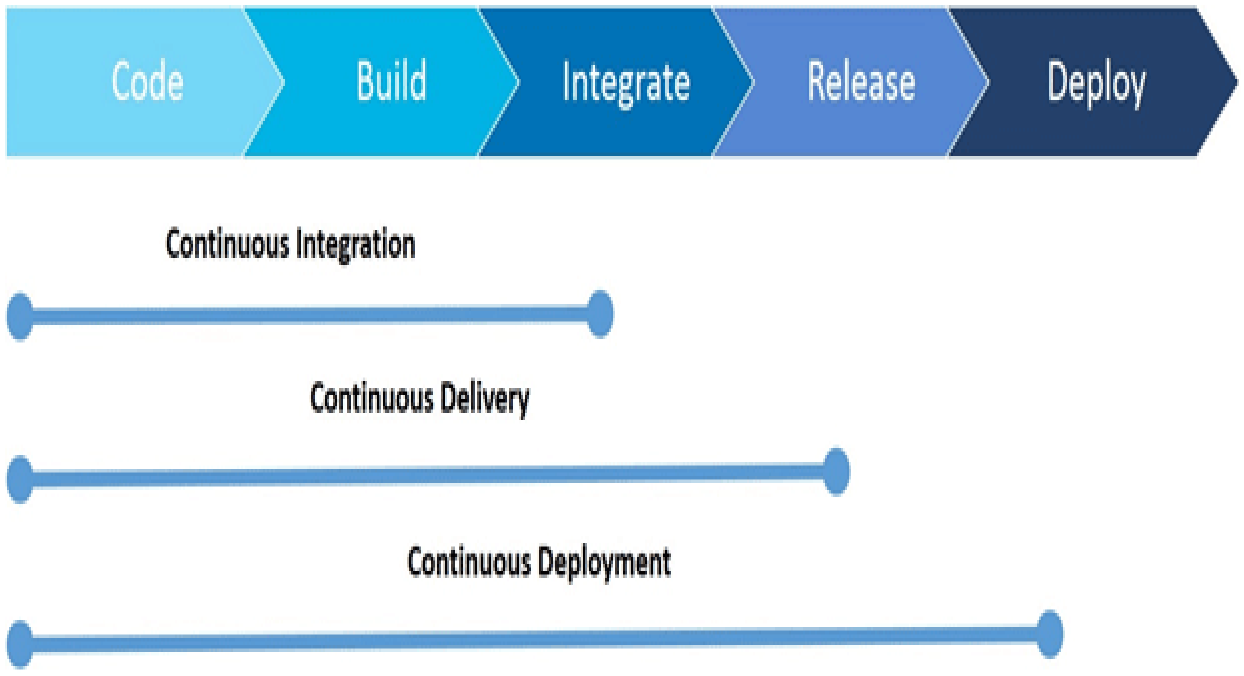
Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control.



Source: developers.redhat.com

Continuous Delivery (CD)

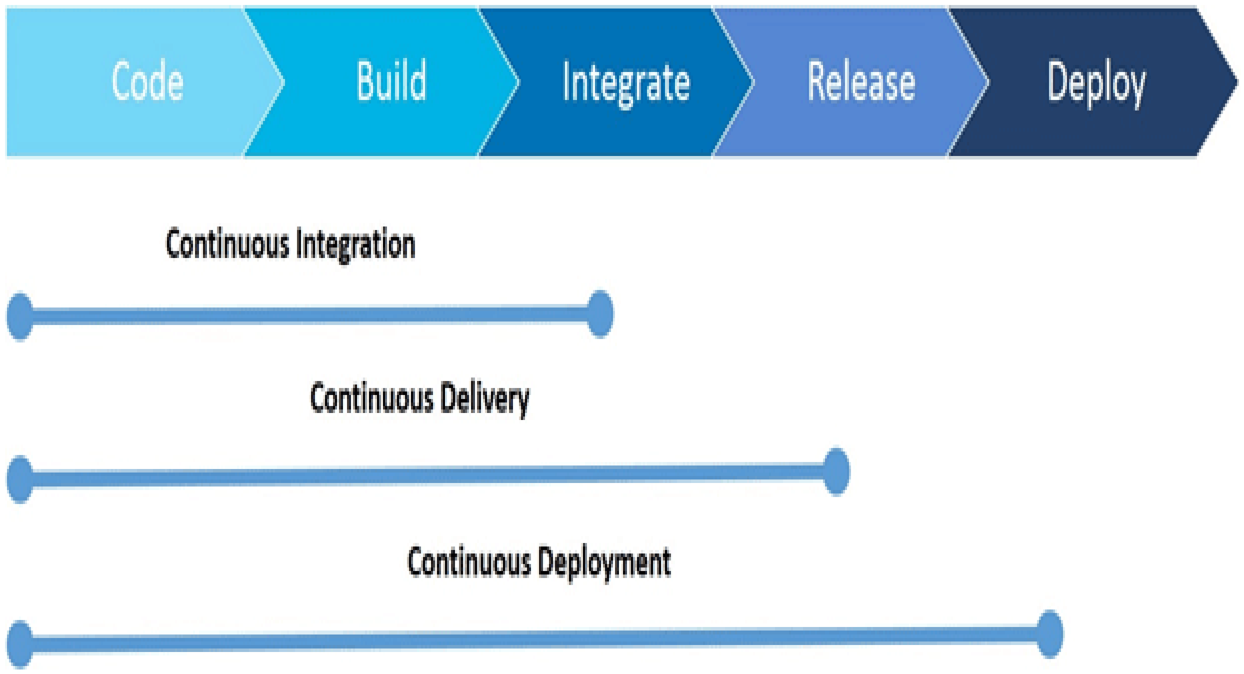
Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.



Source: saviantconsulting.com

Continuous Deployment (CD)

Continuous Deployment (CD) is the process to build, test, configure and deploy from a build to a production environment.

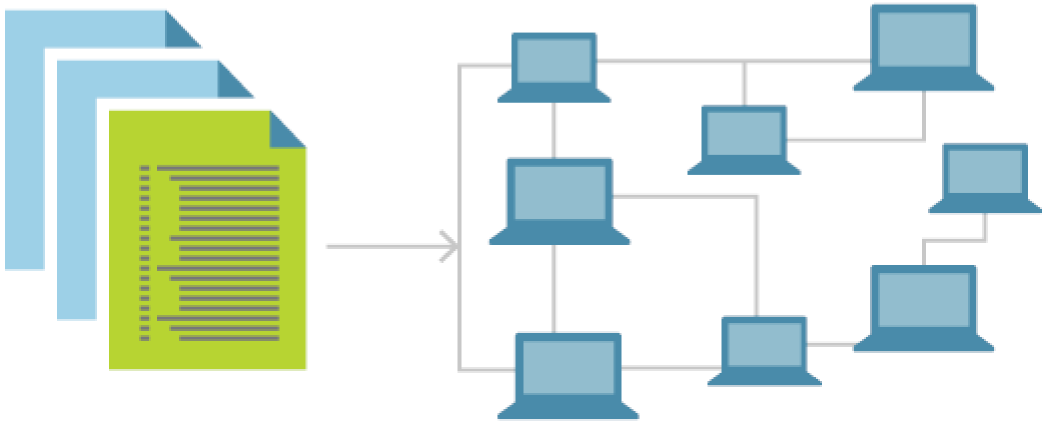


Source: saviantconsulting.com

Infrastructure as Code (IaC)

Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as DevOps team uses for source code.

- Virtual Machine
- Storage
- Software Defined Network

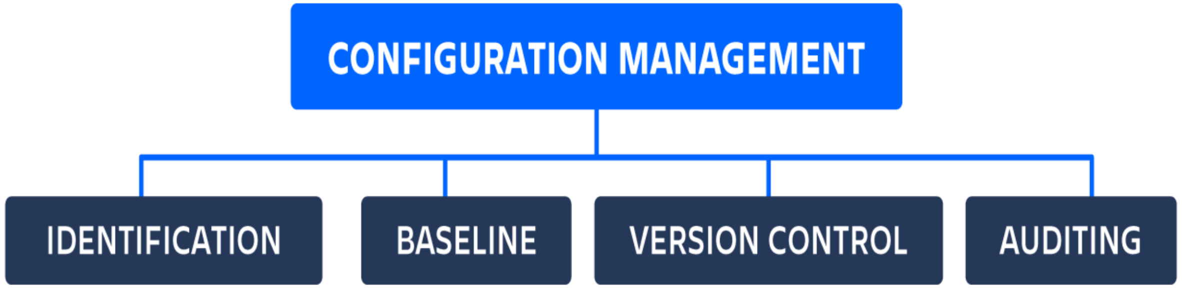


Source: Microsoft.com

Configuration Management

Configuration management is a systems engineering process that tracks and monitors changes to a systems configuration metadata.

- Config file with Service endpoints
- Secrets in a vault

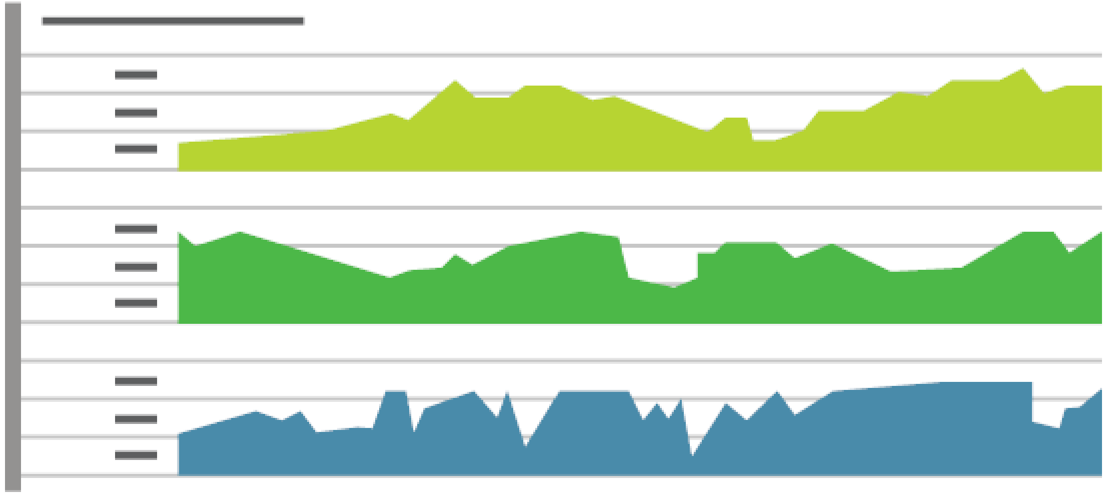


Source: atlassian.com

Monitoring

Monitoring provides feedback from production. Monitoring delivers information about an application’s performance and usage patterns.

- Time to Detect (TTD)
- Time to Mitigate (TTM)
- Time to Remediate (TTR)

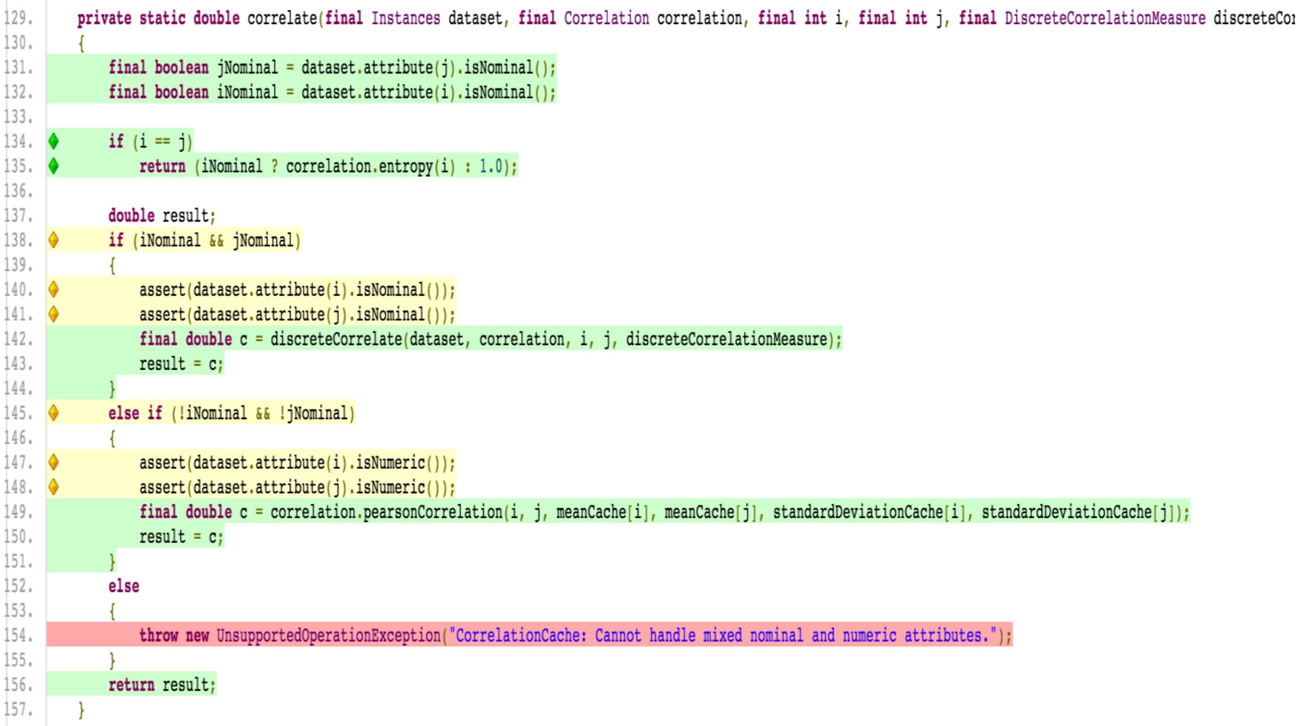


Source: Microsoft.com

Static application security testing (SAST)

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities SAST scans an application before the code is compiled. It’s also known as white box testing.

- Secrets testing
- Code Quality



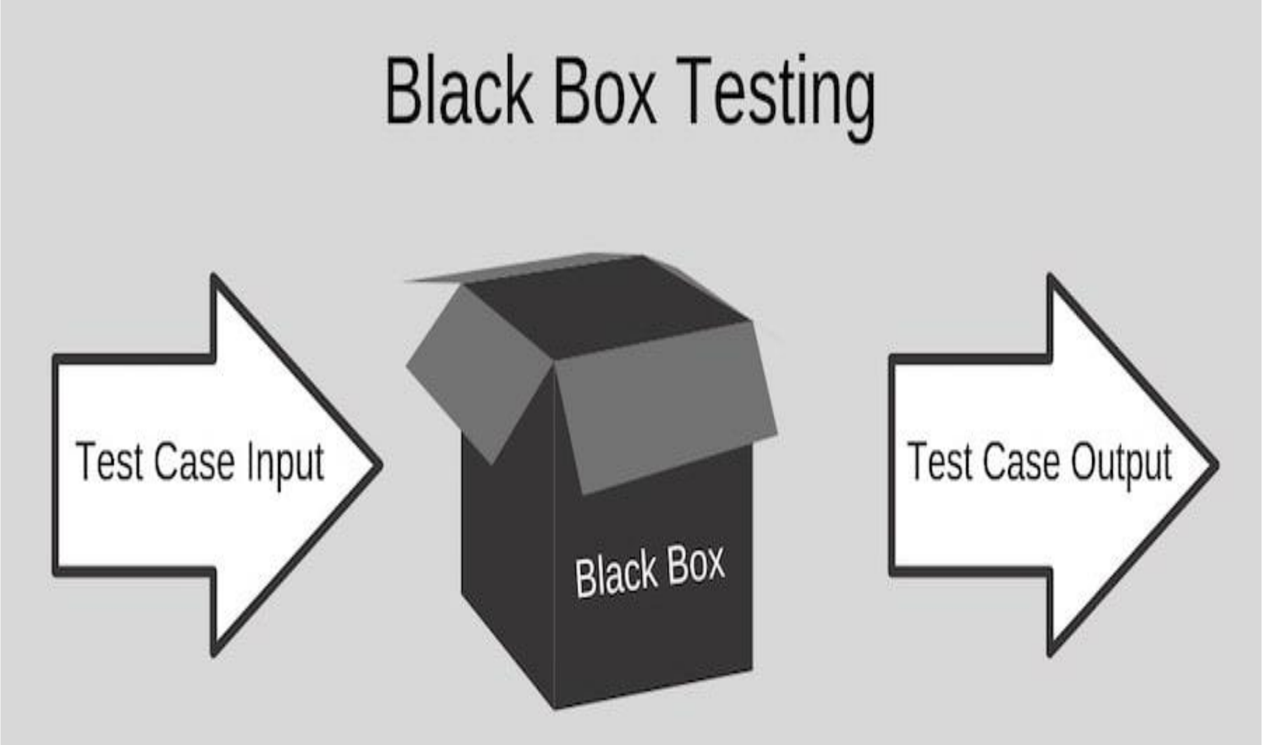
```
private static double correlate(final Instances dataset, final Correlation correlation, final int i, final int j, final DiscreteCorrelationMeasure discreteCo
{
    final boolean jNominal = dataset.attribute(j).isNominal();
    final boolean iNominal = dataset.attribute(i).isNominal();

    if (i == j)
        return (iNominal ? correlation.entropy(i) : 1.0);

    double result;
    if (iNominal && jNominal)
    {
        assert(dataset.attribute(i).isNominal());
        assert(dataset.attribute(j).isNominal());
        final double c = discreteCorrelate(dataset, correlation, i, j, discreteCorrelationMeasure);
        result = c;
    }
    else if (!iNominal && !jNominal)
    {
        assert(dataset.attribute(i).isNumeric());
        assert(dataset.attribute(j).isNumeric());
        final double c = correlation.pearsonCorrelation(i, j, meanCache[i], meanCache[j], standardDeviationCache[i], standardDeviationCache[j]);
        result = c;
    }
    else
    {
        throw new UnsupportedOperationException("CorrelationCache: Cannot handle mixed nominal and numeric attributes.");
    }
    return result;
}
```

Dynamic Analysis Security Testing (DAST)

DAST helps to find certain vulnerabilities in web applications while they are running in production. It's also known as black box testing.

It essentially uses the same techniques that an attacker would use to find potential weaknesses.

- SQL injection
- Cross-site scripting



Source:phoenixnap.com

Software Composition Analysis (SCA)

Software Composition Analysis (SCA) is the process of automating the visibility into open source software (OSS) use for the purpose of risk management, security and license compliance.

- License analysis
- Vulnerability scanning (SCAP)

Source: scart2015.disim.univaq.it/

Deployment Mode

Blue-Green Deployment

It is a release technique that reduces downtime and risk by running two identical production environments called Blue and Green.

Only one of the deployments will serve requests at a time.

Maintaining two identical environments will have significant cost impact.

Canary Deployment

It is a software deployment technique that helps to reduce risk of deploying application to production.

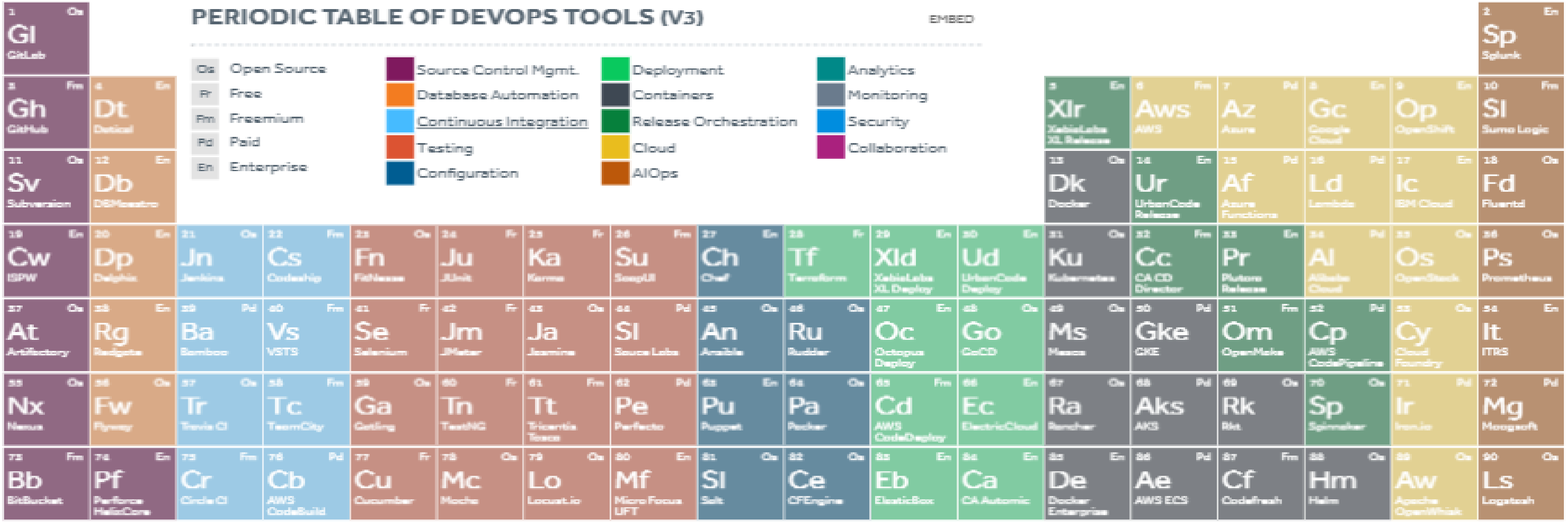
Application will be rolled out to subset of users, monitored for unexpected issues, and is either rolled out to all users or rolled back based on monitoring results.

Supports A/B Testing.

Workload can be partitioned either based on users or servers.

Tools

XebiaLabs
Enterprise DevOps



PERIODIC TABLE OF DEVOPS TOOLS (V3)

EMBED

Os	Open Source
Fr	Free
Fm	Freemium
Pd	Paid
En	Enterprise

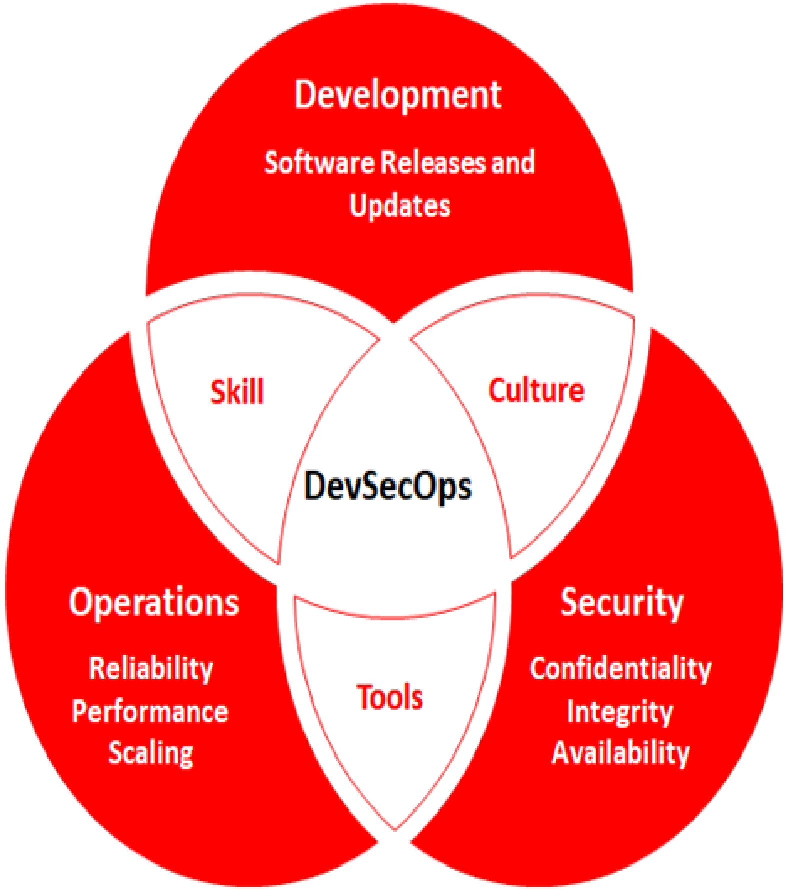
🟪	Source Control Mgmt.
🟧	Database Automation
🟦	Continuous Integration
🟥	Testing
🟦	Configuration
🟩	Deployment
⬛	Containers
🟩	Release Orchestration
🟨	Cloud
🟫	AIOps
🟩	Analytics
⬛	Monitoring
🟦	Security
🟪	Collaboration

1 Os Gl GitLab																	2 En Sp Splunk
3 Fm Gh GitHub	4 En Dt Datical											5 En Xlr XebiaLabs XL Release	6 Fm Aws AWS	7 Pd Az Azure	8 En Gc Google Cloud	9 En Op OpenShift	10 Fm Sl Sumo Logic
11 Os Sv Subversion	12 En Db DBMaestro											13 Os Dk Docker	14 En Ur UrbanCode Release	15 Pd Af Azure Functions	16 Pd Ld Lambda	17 En Ic IBM Cloud	18 Os Fd Fluentd
19 En Cw ISPW	20 En Dp Delphix	21 Os Jn Jenkins	22 Fm Cs Codeship	23 Os Fn FitNesse	24 Fr Ju JUnit	25 Fr Ka Karma	26 Fm Su SoapUI	27 En Ch Chef	28 Fr Tf Terraform	29 En Xld XebiaLabs XL Deploy	30 En Ud UrbanCode Deploy	31 Os Ku Kubernetes	32 Fm Cc CA CD Director	33 En Pr Plutora Release	34 Pd Al Alibaba Cloud	35 Os Os OpenStack	36 Os Ps Prometheus
37 Os At Artifactory	38 En Rg Redgate	39 Pd Ba Bamboo	40 Fm Vs VSTS	41 Fr Se Selenium	42 Fr Jm JMeter	43 Os Ja Jasmine	44 Pd Sl Sauce Labs	45 Os An Ansible	46 Os Ru Rudder	47 En Oc Octopus Deploy	48 Os Go GoCD	49 Os Ms Mesos	50 Pd Gke GKE	51 Fm Om OpenMake	52 Pd Cp AWS CodePipeline	53 Os Cy Cloud Foundry	54 En It ITRS
55 Os Nx Nexus	56 Os Fw Flyway	57 Os Tr Travis CI	58 Fm Tc TeamCity	59 Os Ga Gatling	60 Fr Tn TestNG	61 Fm Tt Tricentis Tosca	62 Pd Pe Perfecto	63 En Pu Puppet	64 Os Pa Packer	65 Fm Cd AWS CodeDeploy	66 En Ec ElectricCloud	67 Os Ra Rancher	68 Pd Aks AKS	69 Os Rk Rkt	70 Os Sp Spinnaker	71 Pd Ir Iron.io	72 Pd Mg Moogsoft
73 Fm Bb BitBucket	74 En Pf Perforce HelixCore	75 Fm Cr Circle CI	76 Pd Cb AWS CodeBuild	77 Fr Cu Cucumber	78 Os Mc Mocha	79 Os Lo Locust.io	80 En Mf Micro Focus UFT	81 Os Sl Salt	82 Os Ce CFEngine	83 En Eb ElasticBox	84 En Ca CA Automic	85 En De Docker Enterprise	86 Pd Ae AWS ECS	87 Fm Cf Codefresh	88 Os Hm Helm	89 Os Aw Apache OpenWhisk	90 Os Ls Logstash

DevSecOps

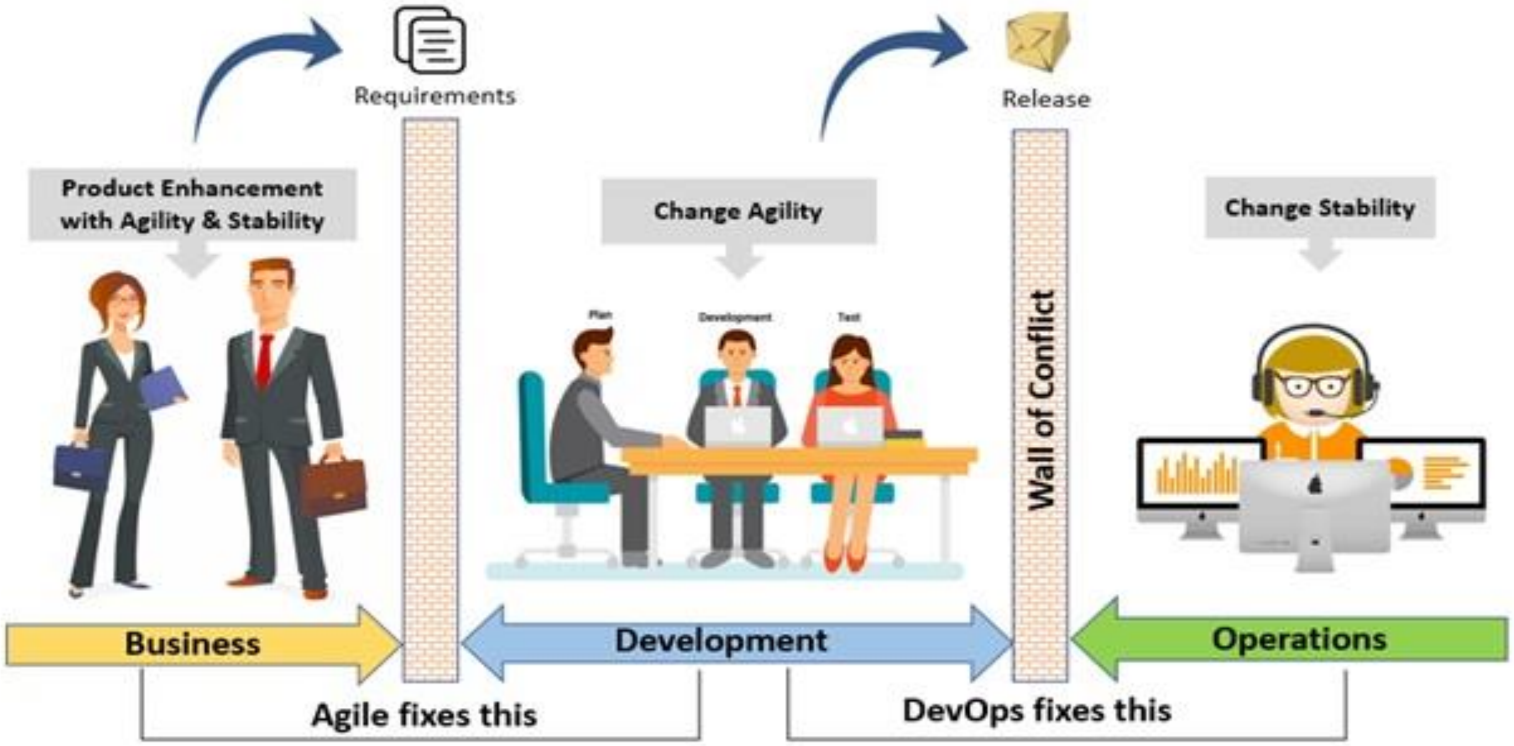
It is the integration of Development, Security, and Operations into a unified, self-directed, self-correcting team completely accountable for all aspects of creating and maintaining the software and digital infrastructure in its portfolios.

- A Mindset that everyone is responsible for security
- A collaborative approach
- A collection of procedure and tools
- A means of building security and compliance into software with focus on continuous improvement
- A strategy driven by heuristics



Source: i.blackhat.com

Organizational Silos

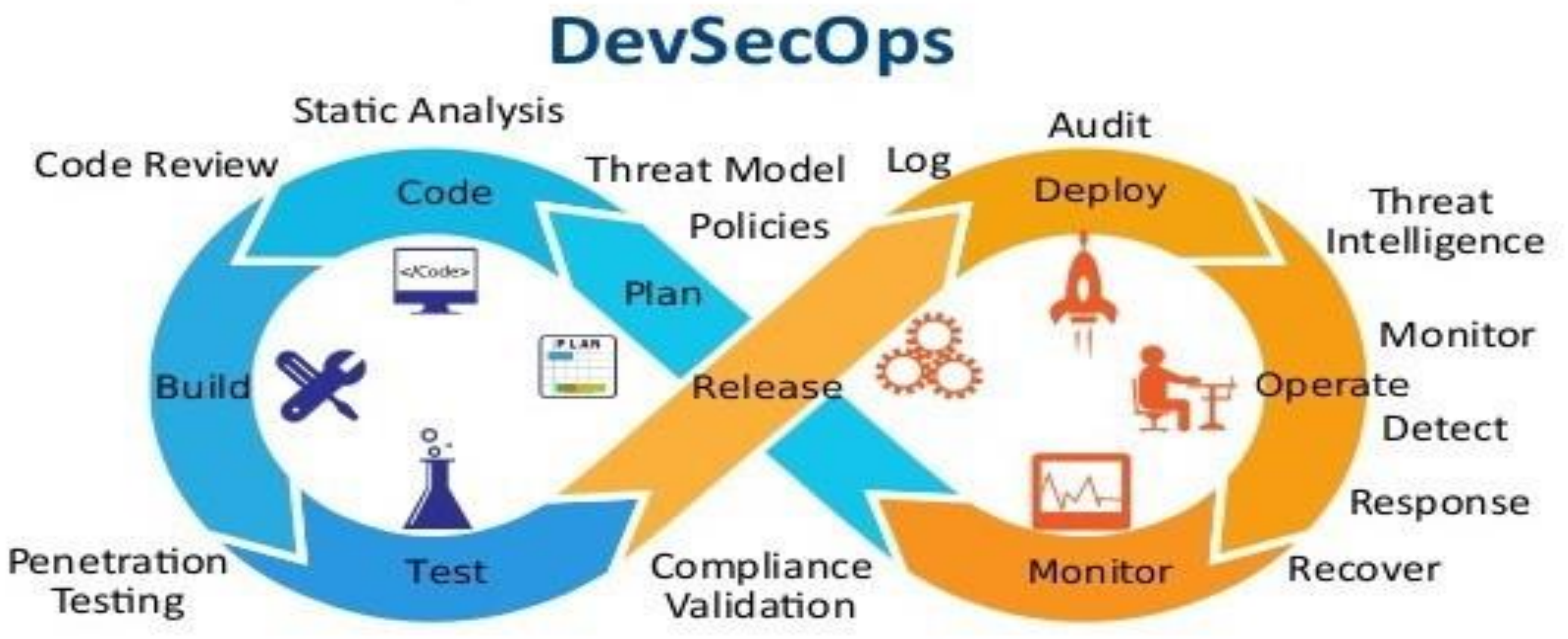


Source: Accenture

Values

- **Leaning in** over Always Saying “No”
- **Data & Security Science** over Fear, Uncertainty and Doubt
- **Open Contribution & Collaboration** over Security-Only Requirements
- **Consumable Security Services with APIs** over Mandated Security Controls & Paperwork
- **Business Driven Security Scores** over Rubber Stamp Security
- **Red & Blue Team Exploit Testing** over Relying on Scans & Theoretical Vulnerabilities
- **24x7 Proactive Security Monitoring** over Reacting after being Informed of an Incident
- **Shared Threat Intelligence** over Keeping Info to Ourselves
- **Compliance Operations** over Clipboards & Checklists

Process



Source: https://media-exp1.licdn.com/dms/image/C4D12AQGsUiRxmNAE5g/article-cover_image-shrink_600_2000/0?e=1587600000&v=beta&t=AFqNGHh9dxUxX5zGNSeXicQ3D1xNO6fjCyKtVThrzrc

Adoption

According to Cloud Security Alliance, six pillars of DevSecOps adoption are

- Collective Responsibility
- Collaboration and Integration
- Pragmatic Implementation
- Bridging the divide between Compliance and Development
- Automation
- Measurement, Monitoring, Report and Action

Transformation Practices

According to Sans, following seven practices are imperative to DevSecOps transformation.

1 Embed automated tests and validation of controls into the deployment cycle.

2 Inventory and analyze reusable code to avoid reintroducing flaws.

3 Monitor code and results continuously in production.

4 Create “triggered” responses that can roll controls back to a known good state if there’s a problem.

5 Evaluate AppSec tools for DevOps capabilities and automation; replace them as needed.

6 Align and coordinate with Dev, Sec and IT Ops teams, and keep communication constant between them.

7 Commit to a culture of process descriptions, automation, continuous monitoring and remediation.

Metrics

- Deployment frequency
- Change lead time – commit time to production deployment
- Change volume – # of user stories
- Change failure rate - # of failed deployments
- Mean time to recovery (MTTR) – production restoration interval
- Availability – uptime/downtime
- Customer issue volume
- Customer issue resolution time
- Time to value
- Time to patch vulnerabilities

Continuous Improvement

“a chain is only as strong as its weakest link”

Theory of Constraints

- Identify the system’s constraint
- Exploit the constraint
- Subordinate everything else to the constraint
- Alleviate the constraint
- Repeat with the next constraint

References

1) https://www.devsecops.org/
2) https://www.ranorex.com/blog/understanding_devsecops/
3) https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/
4) https://www.amazon.com/Continuous-Delivery-Deployment-Automation-Addison-Wesley/dp/0321601912
5) https://tech.gsa.gov/guides/dev_sec_ops_guide/

6) 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr
https://www.youtube.com/watch?v=LdOe18KhtT4