ReleaseIQ

Ten Tenets of DevOps

Provide Strength and Endurance for Engineering Digital Transformations

By Marc Hornbeek & Wayne Greene

Executive Summary

Digital transformations with DevOps are pushing beyond the limits of traditional development and delivery methods. Organizations are releasing with much shorter lead times and much more frequently. As organizations ramp up towards even more accelerated continuous delivery (CD) with automated delivery pipelines, application release decisions depend on having comprehensive, reliable process orchestration and accurate, visible results data for their value streams.

The demand becomes even more critical as organizations that are already well into their transformation journeys need to reach even higher levels of performance and scale for their DevOps and continuous delivery, or their transformations stall.

The 2021 State of DevOps Report indicates that many organizations reach a plateau in their DevOps evolution.

This eBook explains how DevOps solutions, engineered in accordance with ten tenets that include the Nine Pillars framework, coupled with a strong foundation of orchestration and automation capabilities, unstick and accelerate stalled digital transformations.

About the Authors

Marc Hornbeek

Marc Hornbeek, CEO of Engineering DevOps Consulting, and author of the book "Engineering DevOps", has a long history and passion for working with organizations to help them transform to DevOps.

Wayne Greene

Wayne Greene, Head of Marketing at ReleaseIQ, has over 30 years' experience in enterprise class technologies, working across product management and engineering, development, operations, marketing, and strategy. He is passionate

DevOps Pillars

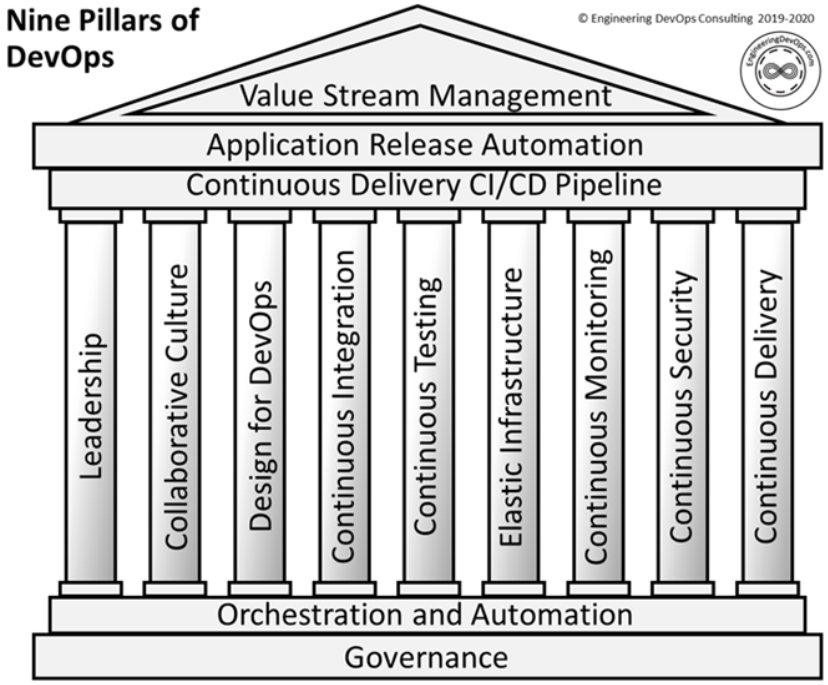
The Nine Pillars framework addresses some key pain-points that cause transformations based in DevOps to stall or fail to reach their goals.

- Leaders and practitioners need a common foundation of understanding of DevOps practices to support communicating goals and to facilitate collaboration. The Nine Pillars is a simple to understand framework that organizes an otherwise complex set of DevOps practices.
- One of the first steps in any transformation is to discover the current state of practices. Each of the Nine Pillars provide specific practices that are useful for practical assessment of the current state and help organize subsequent future state roadmaps.
- Just like a working engine, DevOps works best when engineered to balance the many parts of the solution. The Nine Pillars framework clarifies the foundational parts that need to be kept in balance as DevOps solutions are created, evolved, and maintained.
- DevOps solutions need to be compatible and interwork with other development and operations frameworks including Agile and Site Reliability Engineering (SRE). The Nine Pillars framework provides clarity for communicating and collaborating with those responsible for Agile and SRE.

Nine Pillars of DevOps

Much like a pillar, a column of a building that helps hold the structure up, a DevOps pillar represents a permanent structural part of any well-engineered DevOps. In the book "Engineering DevOps" by Marc Hornbeek, DevOps core practices are organized in Nine Pillars.

As illustrated in the Figure, the Nine Pillars of DevOps are Leadership, Collaborative Culture, Design for DevOps, Continuous Integration, Continuous Testing, Elastic Infrastructure, Continuous Monitoring, Continuous Security and Continuous Delivery. Each DevOps pillar categorizes specific engineering practices that are useful to describe and evaluate DevOps practices an organization needs to engineer and operate DevOps.



In the diagram there are two horizontal foundational structures ("Orchestration and Automation" and "Governance") and three roof structures ("Continuous Delivery CI/CD Pipeline," "Application Release Automation," and "Value-Stream Management"). Horizontal foundational and roof structures cross the Nine Pillars because they are relevant to all the vertical pillars.

Leadership Pillar

The DevOps Leadership Pillar has to do with the aptitudes, attitudes, and actions of people that have leadership roles over teams and organizations that are on a DevOps journey.

- Establish and communicate a long-term vision for organizational and team direction
- Inspirational communication
- Incentives
- Change brings opportunities.
- Demonstrate and encourage empathy.

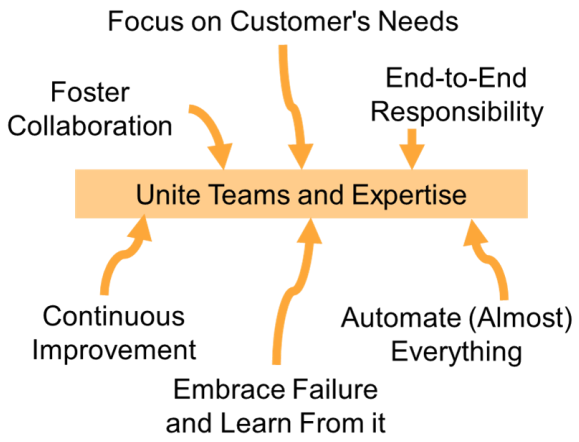
The following are example practices that have been shown to correlate to well-engineered DevOps Leadership:

- Leaders demonstrate a long-term vision for organizational direction and team direction.
- Leaders intellectually stimulate the team by encouraging them to ask new questions and question basic assumptions about the work.
- Leaders provide inspirational communication that inspires pride in being part of the team, says positive things about the team, inspires passion and motivation and encourages people to see that change brings opportunities.
- Leaders demonstrate support by considering others' personal feelings before acting, being thoughtful of others' personal needs and caring about individuals' interests.
- Leaders promote personal recognition by commending teams for better-than-average work, acknowledging improvements in the quality of work and personally complimenting individuals' outstanding work.

Culture Pillar

The DevOps Collaborative Culture Pillar has to do with teams and the organization culture within teams.

- Communication flows fluidly.
- DevOps system is created by a cross-functional team.
- Changes to DevOps systems are led by an expert cross-functional team.
- DevOps System changes follow a phased process.
- Key Performance Indicators (KPIs) monitor the end-to-end DevOps system.

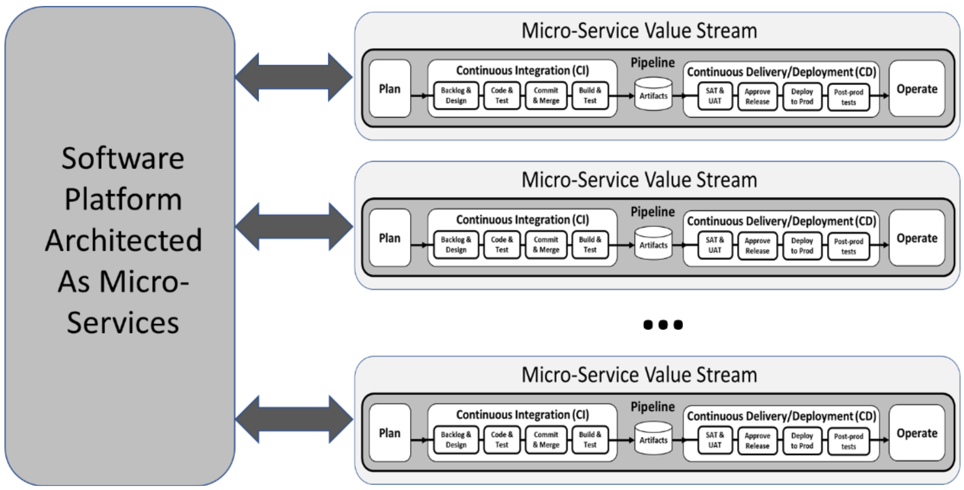


The following are example culture practices correlate to well-engineered DevOps:

- The culture encourages cross-functional collaboration and shared responsibilities and avoids silos between Dev, Ops and QA.
- The culture encourages learning from failures and cooperation between departments.
- Communication flows fluidly across the end-to-end cross-functional team using collaboration tools where appropriate (for example Slack, HipChat, Yammer).
- The DevOps system is created by an expert team, and reviewed by a coalition of stakeholders including Dev, Ops and QA.
- Changes to end-to-end DevOps workflows are led by an expert team, and reviewed by a coalition of stakeholders including Dev, Ops and QA.
- DevOps system changes follow a phased process to ensure the changes do not disturb the current DevOps operation. Examples of implementation phases include proof of concept (POC) phase in a test environment, limited production and deployment to all live environments.
- Key performance indicators (KPIs) are set and monitored by the entire team to validate the performance of the end-to-end DevOps pipeline, always. KPIs include the time for a new change to be deployed, the frequency of deliveries and the number of times changes fail to pass the tests for any stage in the DevOps pipeline.

Application Design Pillar

The Application Design Pillar has to do with how software for applications is designed. Development teams are asked to "just speed it up" while modern distributed application architectures are making processes even more complex. Given this context, it's hardly surprising that speed expectations are challenging unless best design practices are followed.



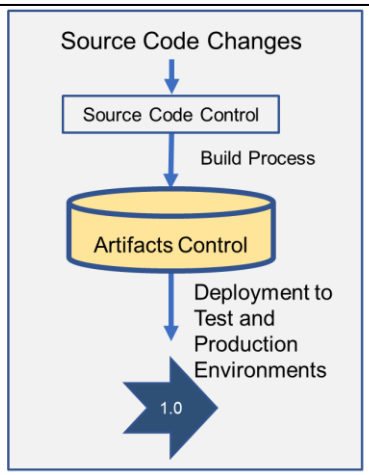
The following are example design practices correlate to well-engineered DevOps:

- Products are architected for modular independent packaging, testing and releases.
- Applications are architected as modular, immutable microservices ready for deployment in cloud infrastructures in accordance with the tenets of 12-factor apps.
- Software source code changes are pre-checked with static analysis tools, prior to commit to the integration branch.
- Software code changes are pre-checked using peer code reviews prior to commit to the integration/trunk branch.
- Software code changes are pre-checked with dynamic analysis tests prior to committing to the integration/trunk branch to ensure the software performance has not degraded.
- Software changes are integrated in a private environment, together with the most recent integration branch version, and tested using functional testing prior to committing the software changes to the integration/trunk branch.
- Software features are tagged with software switches (i.e., feature tags or toggles) during check-in to enable selective feature-level testing, promotion and reverts.
- Automated test cases are committed at the same time code changes are checked in, together with evidence that the tests passed in a pre-flight test environment.
- Developers commit their code changes regularly, at least once per day.

Continuous Integration Pillar

The Continuous Integration Pillar has to do with how changes to software code and build artifacts are “built” or compiled, assembled, or otherwise packaged into executable artifacts for application releases.

- Software version management (SVM) systems control versions of application source code, tools, infrastructure configurations and tests.
- Artifact Repositories control executable images.
- CI manages the process of merging code to a common trunk or integration branch of the code.
- When a code change is “committed” merges, builds and tests are automatically triggered.
- Resources for builds are available on demand, never block a build.
- Incremental builds complete quickly (E.g., less than an hour.

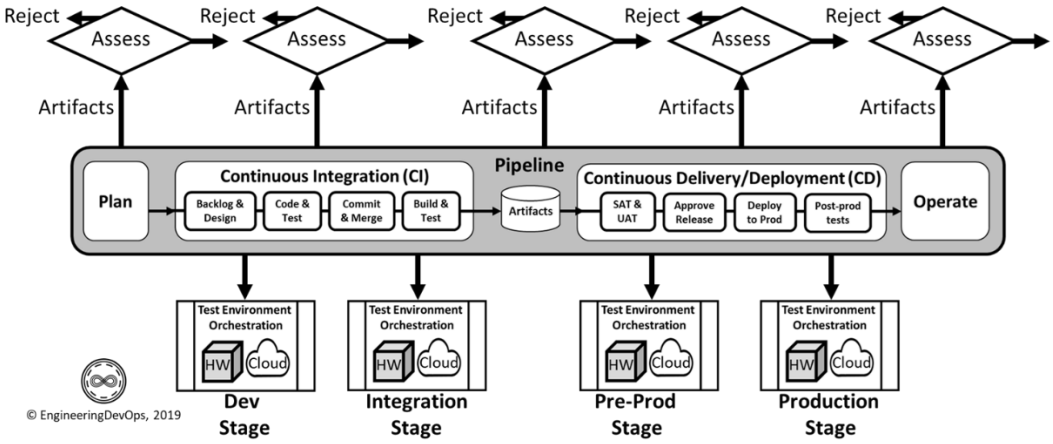


The following are example integration practices that correlate to well-engineered DevOps:

- A software version management (SVM) system manages all source code changes.
- An artifact repository system manages all versions of code images changes.
- A software version management (SVM) system manages all versions of tools and infrastructure configurations and tests that are used in the build process.
- Production software changes are maintained in a single trunk or integration branch.
- Production releases are maintained in a release branch to support software updates.
- Every software commit automatically triggers a build process for all components of the module that has code changed by the commit.
- Once triggered, the software build process is fully automated and produces build artifacts, provided the build time checks are successful.
- The automated build process checks include unit tests.
- Resources for builds are available on-demand and never block a build.
- CI builds are fast enough to complete incremental builds in less than an hour.
- The build process and resources for builds scale up and down automatically according to the complexity of the change. If a full build is required, the CI system automatically scales horizontally to ensure the builds are completed as quickly as possible.

Continuous Testing Pillar

The Continuous Test Pillar has to do with how tests are used to assess software code and build artifacts changes to ensure they meet the requirements for release.

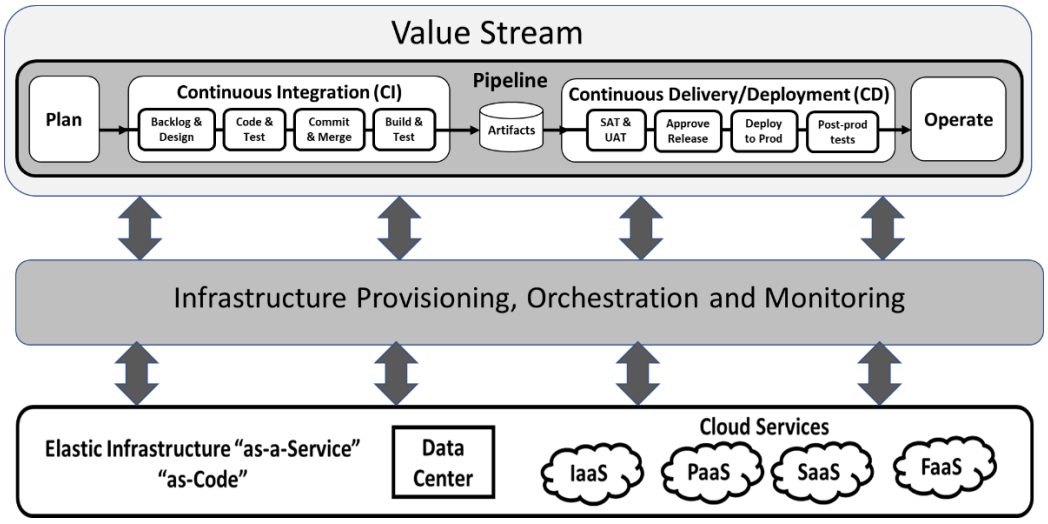


The following are example test practices that correlate to well-engineered DevOps:

- Code changes are tested prior to being integrated to the trunk branch.
- New unit and functional regression tests that are necessary to test a software change are created together with the code and integrated into the trunk branch at the same time the code is.
- A test script standard guide test script creation.
- Tests are selected automatically according to the specific software changes.
- Test resources are scaled automatically according to the resource requirements of specific tests selected and the available time for testing.
- Release regression tests are automated.
- Release performance tests are automated.
- Blue/green testing methods are used to verify deployments in a staging environment before activating the environment to live.
- A/B testing methods are used together with feature toggles to try different versions of code with customers in separate live environments.
- Canary testing methods are used on selected live environments.
- The entire testing life cycle, which may include pre-flight, integration, regression, performance and release acceptance tests are automatically orchestrated across the DevOps pipeline.

Elastic Infrastructure Pillar

The Elastic Infrastructure Pillar has to do with how resource requirements (i.e., computing machines, storage and networks") for builds and testing environments vary in near real time depending on the workload requirements to support specific changes and the variable demand of a constantly changing number of users that need to use resources.

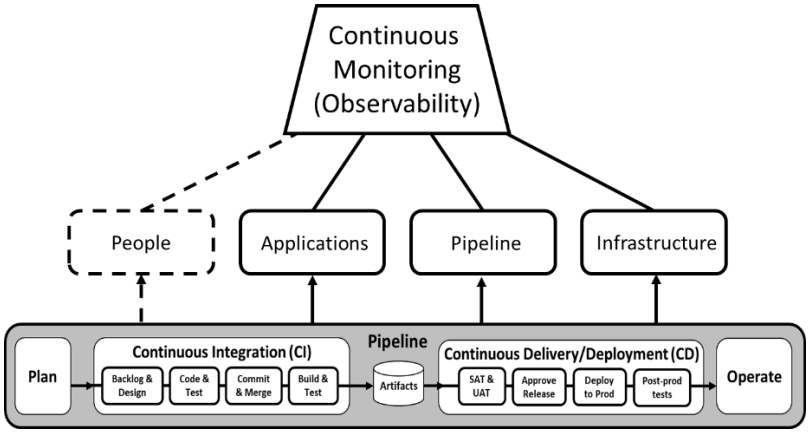


The following are example infrastructure practices that have been shown to correlate to well-engineered DevOps

- Data and executable files needed for building and testing builds are archived frequently and can be reinstated on demand.
- Build and test processes are flexible enough to automatically handle a wide variety of exceptions gracefully. If the build or test process for a component is unable to complete, then the process for that failed component is reported and automatically scheduled for analysis but build and test processes for other components continue.
- Configuration data is managed in a configuration management database (CMDB).
- Infrastructure changes use configuration management tools that assure idempotency.
- Automated tools are used to support immutable infrastructure deployments.
- Equal performance for all. The user performance experience of the build and test processes by different teams are consistent for all users, independent of location or other factors.
- Fault recovery mechanisms are provided. Build and test system fault monitoring, fault detection, system and data monitoring and recovery mechanisms exist. They are automated and are consistently verified through simulated failure conditions.
- Infrastructure failure modes are frequently tested.
- Disaster recovery procedures are automated.

Continuous Monitoring Pillar

The Continuous Monitoring pillar refers to instrumenting, collecting, and analyzing data needed to manage health and performance of applications, databases, pipelines, and infrastructures and to engineer improvements.

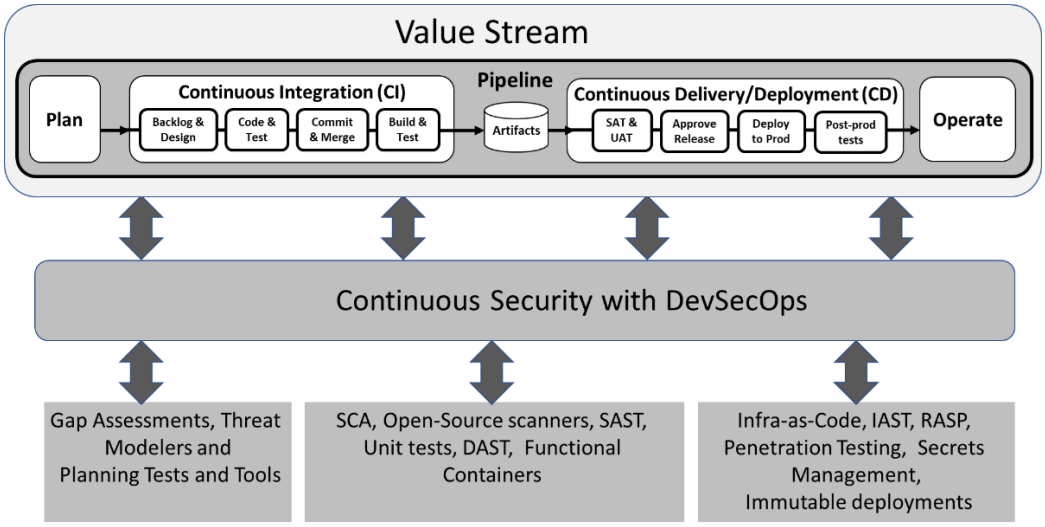


The following are example monitoring practices that have been shown to correlate to well-engineered DevOps:

- Logging and proactive alert systems make it easy to detect and correct DevOps system failures. Logs and proactive system alerts are in place for most DevOps component failures and are organized in a manner to quickly identify the highest-priority problems.
- Snapshot and trend results of each metric from each DevOps pipeline stage (for example, builds, artifacts, tests) are automatically calculated in process and visible to everyone in the Dev, QA and Ops Teams.
- Key performance indicators (KPIs) for the DevOps infrastructure components are automatically gathered, calculated and made visible to anyone on the team that subscribes to them. Example metrics are availability (uptime) of computing resources for CI, CT and CD processes, time to complete builds, time to complete tests, number of commits that fail and number of changes that need to be reverted.
- Metrics and thresholds for DevOps infrastructure components are automatically gathered, calculated and made visible to anyone on the team that subscribes to them. Example metrics are availability (uptime) of computing resources for CI, CT and CD processes, time to complete builds, time to complete tests, number of commits that fail and number of changes that need to be reverted due to serious failures.
- Process analytics are used to monitor and improve the integration, test and release process. Descriptive build and test analytics drive process improvements.
- Predictive analytics are used to dynamically adjust DevOps pipeline configurations.

Continuous Security Pillar

The Continuous Security Pillar, includes DevSecOps practices for integrating security of applications, databases, pipelines, and infrastructures into the continuous delivery pipeline.

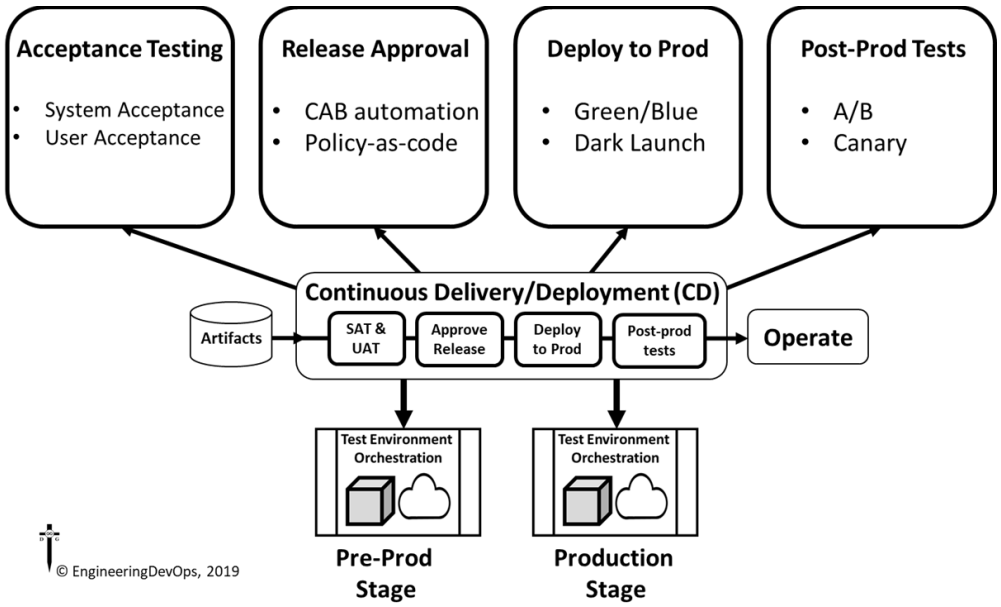


The following are examples of DevSecOps practices that correlate to well-engineered DevOps:

- Developers are empowered and trained to take personal responsibility for security.
- Security assurance automation and security monitoring practices are embraced by the organization.
- Security platforms expose full functionality via APIs for automation capability.
- Proven version control practices and tools are used for all application software, scripts, templates and blueprints that are used in DevOps environments.
- Static Application Security Test (SAST) and Software Composition Analysis (SCA) are automated as part of the DevOps pipeline
- Container are scanned and used for application testing and deployments.
- Immutable infrastructure mindsets are adopted to ensure production systems are locked down.
- Security controls are automated so as not to impede DevOps agility.
- Security tools are integrated into the CI/CD pipeline.
- Source code for key intellectual property on build or test machines are only accessible by trusted users with verified credentials. Build and test scripts do not contain credentials for access to any system that has intellectual property. Intellectual Property is divided such that not all of it exists on the same archive and each archive has different credentials.

Continuous Delivery Pillar

The Continuous Delivery pillar refers to practices for preparing release artifacts for deployment to production.



The following are example Delivery practices that correlate to well-engineered DevOps

- Delivery and deployment stages are separate. The delivery stage precedes the deployment pipeline.
- All deliverables that pass the delivery metrics are packaged and prepared for deployment using containers.
- Deliverable packages include sufficient configuration and test data to validate each deployment. Configuration management tools are used to manage configuration information.
- Deliverables from the delivery pipeline are automatically pushed to the deployment pipeline once acceptable delivery measures are achieved.
- Deployment decisions are determined according to predetermined metrics. The entire deployment process may take hours, but usually less than a day.
- Deployments to production environments are staged such that failed deployments can be detected early and impact to customers isolated quickly.
- Deployments are arranged with automated recovery and self-healing capabilities in case a deployment fails.

Orchestration and Automation Foundation

Nine DevOps Tenets, based upon the Nine Pillars, have in common a foundation for Orchestration and Automation. Orchestration and Automation can be considered a tenth tenet, because DevOps requires efficient connections and collaboration between people, processes, and technology facilitated by orchestration and automation. Orchestration and automation leverages tools, arranged as integrated toolchains, to help DevOps teams build out their environments, systems, and test set ups. Product development teams and associated stakeholders will either have a high or low governance perspective, either letting the individual teams choose their tooling: source code repository, continuous integration, testing, continuous delivery, security etc., or there is more guidance from a central DevOps "platform" team.

Whether or not the tools are centrally chosen, managed, or operated or point tools picked by individual teams, the overall DevOps pipeline needs to operate as if it is a single toolchain system to drive increased organizational productivity, performance and resilience. This implies the need for an orchestrated and automated "uber" controller above the tooling choices. The DevOps Orchestration and Automation Platform or foundation deserves to be considered as a first-class object in the DevOps Architecture.

The following are example Orchestration and Automation practices that correlate to well-engineered DevOps

- Pipelines for a specific application software value stream are composed of toolchains.
- Toolchains for any software platform or application may have persistent tools working in combination with tools that need to be orchestrated and deployed dynamically according to build and test workloads requirement for each stage of the pipeline.
- Automation of pipeline stages and processes are abstracted away from specific tools to enable a variety of tools to be used for each function including manual steps.
- Data and metadata resulting from code commits, releases, project management issues, tests, and verification activities are collected, analyzed, and correlated to support release decisions and to provide information for continuous improvements.
- DevOps orchestration and automation platforms are easily extensible to support different tool integrations and span multiple environments.
- Ensure that the complete full end-to-end process is orchestrated and automated. Leave no step or tool interaction behind.
- Provide both no-code drag and drop composers alongside pipeline-as-code to allow templating and versioning for applicability to environment sized from small to extreme scaled environments.
- Compliance and security standardization in the pipeline should be completely integrated into the automation and orchestration

Applications of the Nine Pillars of DevOps

The Nine Pillars framework can be used to alleviate some key pain-points that cause transformations based in DevOps to stall or fail to reach their goals.

- Leaders and practitioners can use the Nine Pillars help their teams obtain a common foundation of understanding of DevOps practices to support communicating goals and to facilitate collaboration.
- DevOps transformation development planners can use the Nine Pillars to discover the current state of practices. Each of the Nine Pillars provide specific practices that are useful for practical assessment of the current state and help organize subsequent future state roadmaps.
- DevOps implementors can use The Nine Pillars framework to ensure the foundational parts of each DevOps project are kept in balance as DevOps solutions are created, evolved, and maintained.
- Agile and SRE transformation teams can use The Nine Pillars to ensure the solutions are compatible and interwork with Agile and Site Reliability Engineering (SRE) implementations.

DevOps Business Case

DevOps solutions, like any other investment, must be justified against other possible investments. Investments with the highest return are the ones that gain approval. Without a convincing business case, DevOps solutions may seem to some uninformed financial and business leaders to be a significant cost without a quantifiable or justifiable Return-On-Investment.

The primary data points that are most important to a DevOps business case are as follows:

1. What is the cost-equivalent for the aggregate amount of time is being spent by developers, QA and Ops team members waiting for results for each stage of the value stream, for the entire set of applications that the DevOps solution applies to?

2. How much of the cost from #1 can be saved if a DevOps system is put in place?

When considering the above questions, include all the use cases and pain-points described in this document.

To get reasonably accurate data it is necessary to make some measurements. Consider the average, median, minimum and maximum cases. A current-state Value Stream Map is a good way to determine and document the current state and measurement of stage process and wait times.

A business case based on the median rather than the average values is usually more realistic because of the nature of this type of data in which median values tends to skew toward the maximum values. This happens because the minimum is when there are no errors in the value stream, and in a mature environment this should be the case often, but errors that cause long delays are more exaggerated and are important to the business.

What this Means

DevOps is a powerful framework that enables many benefits for organizations that use it. Achieving performance efficiently with DevOps depends on following best practices. By following the nine pillars of practices enumerated in this eBook, organizations can achieve, sustain and improve the performance potential that DevOps has to offer.

References

2021 State of DevOps

Book, Engineering DevOps, Marc Hornbeek, November 2019

https://devops.com/nine-pillars-of-devops-best-practices/

https://containerjournal.com/topics/container-management/9-pillars-of-containers-best-practices/

https://containerjournal.com/editorial-calendar/best-of-2021/9-pillars-of-engineering-devops-with-kubernetes/

https://devops.com/9-pillars-of-continuous-security-best-practices/

https://devops.com/27-factor-assessments-lead-to-success-with-devops/

Call to Action

Organizations that desire to advance their digital transformations, unblock their DevOps and continuous delivery pipeline bottlenecks through improvements in their DevOps solutions are encouraged to contact ReleaseIQ. ReleaseIQ can provide more detailed evaluations, assessments, and solution roadmaps that will unblock and accelerate digital transformation progress.

www.releaseiq.io

info@releaseiq.io