DevOps Governance Engineering

This paper is derived from selections from my book.



The book “Engineering DevOps” can be obtained here: [**mybook.to/engineeringdevops**](http://mybook.to/engineeringdevops)

This white paper and other useful papers, glossary checklists, slides, blueprints, templates and engineering practices can be downloaded from [**https://engineeringdevops.com/documents**](https://engineeringdevops.com/documents)

About the Author:

**Marc Hornbeek** a.k.a. DevOps\_the\_Gray esq.

[linkedin.com/in/marchornbeek](https://www.linkedin.com/in/marchornbeek)

June 18, 2020

© Engineering DevOps Consulting 2019, 2020

All rights reserved

**How do you Engineer Governance for DevOps?**

**DevOps Governance Engineering**

IT governance is defined as the processes that ensure the effective and efficient use of IT in enabling an organization to achieve its goals. With DevOps, the rate of request for change is continuously increasing, and the cadence of governance decisions needs to respond to this. Uncontrolled proliferation of solutions leads to greater financial cost, a likely divergence in process, and potentially a loss of the feedback and process control that is critical to successful DevOps. DevOps relies on practitioner empowerment, while DevOps governance processes rely on central control.

**Why Is Governance Engineering Important for DevOps Engineering?**

Proper DevOps governance provides the following benefits:

• Taxonomy for communicating governance policies

• Statutory and regulatory compliance (evidence for audits)

• Reinforce enterprise culture, ethics, and behaviors

• Sets direction for use of technology

• Enforce standard operating procedures

• Risk management • Cost management

• Visibility and process control

• Performance management

• Protect confidentiality

• Facilitates trust with stakeholders

**How Is Governance Engineered for DevOps?**

DevOps governance is a quite different approach than traditional IT governance. It requires rethinking how work is funded and the Lines of Defense framework as follows:

**First Line—Who Owns the Risk**—Individual developers and engineers

**Second Line—Who Sets Policy and Monitors the Risk—** Governance and Risk functions that set policy and monitor risk daily

**Third Line—Independent Assurance**—Internal audits that provide independent insurance and report directly to the audit committee or the board

**Fourth Line—External Partners**—Auditors and regulators who must be brought into the conversation and given full transparency into development processes and risk management

**A sound governance strategy should cover organizational, financial, and operational requirements**. The process must be designed to be repeatable, as the DevOps journey is not just a one-off deal but an ongoing process to facilitate continuous improvement.

**Organizational**—Define your business services and the organizational roles to deliver those services and break down the traditional silos between development and operations. Then create SLAs to guarantee those as required to support your business services and applications.

**Financial**—Define your budget for research and development, business support, operational support, network, security (cyber and physical) systems, and IT operations. Then wrap up the process by creating procurement methodologies that will support financial governance.

**Operational**—Define the processes, procedures, and policies that govern how you will operate and who will operate critical functions. This requires aligning your business goals across your entire organization to ensure that continuous integration and development are achieved.

***Figure 1 —DevOps Governance Engineering Blueprint*** illustrates a DevOps governance engineering blueprint that provides for control with practitioner participation. For **The First Way of DevOps**, Continuous Flow control gates use control logic at the boundary of each pipeline stage to control flow.

**Example of control parameters for Continuous Flow are listed as follows**:

• Incident and issues management; thresholds for work items, incidents, and issues. For example, a threshold could be that no priority zero bugs exist.

• Seek approvals outside automated pipelines. For example, approvals from legal approval departments or auditors could be a requirement.

• Quality validation. Query metrics from tests on the build artifacts such as pass rate or code coverage and deploy only if they are within required thresholds.

• Security scan on artifacts. Ensure security scans such as anti-virus checking, code signing, and policy checking for build artifacts have completed.

• User experience relative to baseline. Ensure the user experience hasn’t regressed from the baseline state.

• Change management. Wait until change management procedures are complete before deployment.

• Infrastructure health. Execute monitoring and validate the infrastructure against compliance rules after deployment.



**Figure 1—DevOps Governance Engineering Blueprint**

**The Second Way of DevOps** describes the principles that enable fast and Continuous Feedback from right to left at all stages of the value stream (i.e., pipeline). Second Way DevOps systems are essential as an enterprise scales DevOps in complexity to support many parallel pipelines and deployment targets. Measurement and feedback capabilities embedded in DevOps infrastructures and pipelines help shorten and amplify feedback loops. When working in complex systems, real-time feedback provides the earliest opportunity to detect and correct errors before a catastrophic event develops. As continuous security, continuous testing, and continuous deployment environments scale up, governance must be upgraded to include control gates between pipeline stages that utilize control parameters from feedback sources.

**The Third Way of DevOps** prescribes Continuous Improvement, which involves continual experimentation, which requires taking risks, learning from success and failure, and understanding that repetition and practice are the prerequisites of mastery. Third Way DevOps systems are essential for an enterprise to accelerate its pace of innovative changes while controlling reliability through SRE systems and practices. Governance practices for Third Way DevOps systems build on governance practices of The First Way and The Second Way DevOps systems. In The Third Way, SREs ensure reliability of Service Level Agreements (SLAs) in production by continuously monitoring Service Level Indicators (SLIs) during the pipeline to ensure candidate releases that do not meet Service Level Objectives (SLOs) are not released to production. Changes that do not meet SLOs are rolled back for remediation.

**Recommended engineering practices for DevOps governance** are broken into three categories: **DevOps governance for People, Process, and Technology**.

**Recommended engineering practices for DevOps governance for people are as follows:**

• The Governance Policy Board includes cross-functional representation (e.g., Executive, Legal, Product Owners, Security, Human Resources, Architects, Product Developers, Operations, QA, Infrastructure, Finance, Product Management, Vendor Management, and Project Management) to ensure various stakeholder requirements are covered by the policies.

• Governance policies are reviewed and updated regularly with Development team inputs to keep them current with the rapid changes required for DevOps.

• Product teams deliver ongoing value, and the business can count on improvements to meet market demand.

• Roles of the users of the pipelines include responsibilities to respond to gate criterion exceptions.

**Recommended engineering practices for DevOps governance for process are as follows:**

• The scope of individual governance policies is defined for all domains (e.g., geographic regions).

• Resource consumption dashboarding, trend analytics, and optimization scenarios views enable you to see what users are consuming and whether certain components can be reduced if they’re not needed.

• Every project (test, automation, DevOps, etc.) must adhere to the defined reference architecture using the security blueprints included in a prescribed catalog of services.

• Code changes are monitored to ensure that a “two sets of eyes” peer review take place.

• Unauthorized change monitoring is used as a detective control. All change events are logged. Governance functions and management review what changes are made and determine if they are authorized.

• Access to environments (e.g., production) is limited specific roles. When needed, access is given through time-limited tokens granted under access approval rules (just-in-time admin).

• Gate criteria are defined for each pipeline stage including thresholds for control parameters such as incident and issues, approvals outside automated pipelines, quality validation, security, user experience, change management, and infrastructure health.

• Workflows are implemented for the pipelines to utilize gate information as part of the promotion of changes from stage to stage.

• Gate criteria, tools, and workflows are maintained.

**Recommended engineering practices for DevOps governance for technologies are as follows**:

• Governance focus is shifted away from a project focus towards a product focus.

• Developers leverage automation, such as building on every commit, implementing static code analysis on every build, scanning for open-source vulnerability, performing static security scanning, and running automated tests.

• A “clean room model” is adopted, where all product pipelines— whether they are application, test, or infrastructure code—are identified and registered under source control.

• Tools for collect information needed for gate criterion decisions are available in the pipelines.

Tools process gate criteria data that is collected. Gate criteria data collection and processing tools are integrated into the pipeline.

**What Is Needed to Engineer Governance for DevOps?**

The following steps list phases of activities are recommended for implementing governance in accordance with recommended engineering practices.

**Phase 1: Deﬁne Engineering Governance**

• Define gate criteria for each pipeline stage, including thresholds for control parameters such as incident and issues, approvals outside automated pipelines, quality validation, security, user experience, change management, and infrastructure health.

• Identify tools for collecting information needed for gate criteria decisions.

• Identify tools capable to process the gate criteria data collected.

• Production reliability targets (Error budgets = SLO) are determined for each application and service. SREs and Dev teams determine the targets together. This forms a contract between Dev and Ops by which release decisions are governed.

**Phase 2: Implement Engineering Governance**

• Integrate the gate criteria data collection and processing tools into the pipeline.

• Implement workflows for the pipelines to utilize gate information as part of the promotion of changes from stage to stage.

• Update roles of the users of the pipelines to include responsibilities and respond to gate criteria exceptions.

• Update the gate criteria, tools, and workflows as needed for continuous improvement.

**Phase 3: Operations with Engineering Governance**

• Site Reliability Engineering practices provide the means (tools, automation, and workflows) to monitor SLIs and act when SLOs are not met.

• SLOs are reviewed periodically to make sure they make sense, as situations tend to change over time during continuous improvement cycles.

• SLIs data collection and processing tools are integrated into the pipeline.

As indicated in the last phase, Site Reliability engineering practices are recommended to help govern DevOps because SRE provides the metrics and disciplined methodologies consistent with operating a well-engineered DevOps environment. In the next chapter, SRE is explained in more detail.

**Summary**

In summary, the answer to the question: “**How do you engineer Governance for DevOps?”** has been explained in this white paper. DevOps governance is a quite different approach than traditional IT governance. With DevOps well-engineered governance implements Team Governance, Pipeline Governance and Site governance. DevOps practices to implement governance include a taxonomy for communicating governance policies, statutory and regulatory compliance (evidence for audits), reinforce enterprise culture, ethics, and behaviors, sets direction for use of technology, enforce standard operating procedures, risk management, cost management, visibility and process control, performance management, protect confidentiality and facilitates trust with stakeholders.

**Learn More**



The book “Engineering DevOps” can be obtained here: [**mybook.to/engineeringdevops**](http://mybook.to/engineeringdevops)

This white paper and other useful papers, glossary checklists, slides, blueprints, templates and engineering practices can be downloaded from <https://engineeringdevops.com/documents>

About the Author:

**Marc Hornbeek** a.k.a. DevOps\_the\_Gray esq.

[linkedin.com/in/marchornbeek](https://www.linkedin.com/in/marchornbeek)

© EngineeringDevops 2019, 2020

All rights reserved