Implementing DevOps-at-Scale within the Federal Government

5 Important steps and CIO takeaways in making DevOps a scalable endeavour within the government.

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the development lifecycle and provide continuous delivery and deployment of software. Here are some steps you can take to implement DevOps at scale in your organization:

Step 1: Build a culture of collaboration

Building a ***culture*** of collaboration is ***essential*** for successful DevOps. DevOps relies on close collaboration between development and operations teams, and effective communication and teamwork are key to its success. Here are some ways you can build a culture of collaboration in your organization:

1. Encourage cross-functional teamwork: Encourage development and operations teams to work together, share knowledge, and communicate regularly. This can be achieved through practices like pair programming, code reviews, and daily stand-ups.

2. Foster a culture of continuous learning: Encourage your teams to stay up to date with the latest tools and techniques. This can be achieved through training programs, hackathons, and knowledge-sharing sessions.

3. Emphasize the importance of collaboration: Make it clear that collaboration is a key part of your organization's culture. This can be achieved through incentives, recognition programs, and leadership support.

4. Encourage open communication: Encourage open and transparent communication across your organization. This can be achieved through practices like open office hours, regular team meetings, and company-wide communication channels.

5. Identify key senior manager to be sponsors or champion for collaboration. Show and tell success stories of awesome collaborators within your organization.

6. Staff a dedicated organizational change management expert to develop and implement a successful DevOps adoption strategy.

***CIO Action**: Mandate an executive with an action plan to implement DevOps as a standard practice within the organization*

Step 2: Automate repetitive tasks

Automation can help your teams work faster and more efficiently. Automating repetitive tasks can be a key component of a successful DevOps workflow. Automation can help your teams work faster and more efficiently by taking care of routine tasks, such as code deployment, testing, and infrastructure management.

There are many tools available for automating DevOps tasks, including:

1. Jenkins: an open-source automation server that helps you automate parts of the software development process.

2. Puppet: an automation tool that helps you manage your infrastructure as code. You can use Puppet to automate tasks such as provisioning and configuring servers, and managing software deployments.

3. Ansible: an open-source automation tool that helps you automate tasks such as provisioning and configuring servers, and deploying applications.

4. Terraform: an infrastructure as code tool that helps you automate the process of creating, updating, and managing infrastructure resources, such as servers, databases, and networking components.

5. Docker: a containerization platform that helps you automate the process of building, testing, and deploying applications.

Automating tasks can save your teams time and reduce the risk of errors. It's important to carefully plan and test your automation processes to ensure they are reliable and effective.

***CIO Action**: Make sure key tools required to for automated are readily available for development teams and are approved through internal governance.*

Step 3: Use version control

Version control systems like Git allow you to track changes to your code and collaborate with other developers. They also make it easier to roll back changes if something goes wrong.

Git is a version control system that allows developers to track changes to their code and collaborate with other developers. It is widely used in DevOps to manage code changes and ensure that software is developed and deployed reliably.

Here are some ways you can use Git in your DevOps workflow:

1. Set up a Git repository: A Git repository is a central location where code changes are tracked and stored. You can set up a repository on a server or use a hosted service like GitHub, GitLab, BitBucket and Azure DevOps.

2. Collaborate with other developers: Git allows multiple developers to work on the same codebase at the same time. Developers can use Git to track their own code changes and merge them with the main codebase when they are ready.

3. Track code changes: Git records every change made to the codebase, allowing you to see who made a change and when. This makes it easier to track bugs and understand how the code has evolved over time.

4. Roll back changes: If something goes wrong with your code, you can use Git to revert to a previous version. This can be helpful if you need to fix a bug or revert to a stable version of your code.

Git is a powerful tool that can help your teams work more efficiently and collaborate more effectively.

***CIO Action**: Mandate Chief Enterprise Architect to ensure standards and guidelines are updated to reflect use of Git as a primary version control mechanism.*

Step 4: Set up continuous integration and delivery

Continuous integration involves merging code changes from different developers into a single codebase. Continuous delivery automates the process of testing and deploying code changes. Together, these practices help you deploy code updates more quickly and reliably.

Most organization have some processes established to manage change and releases for their software. However, DevOps practices aim to shorten the development lifecycle and provide standardized mechanism for large software teams to be able to work together to deliver quality software.

Continuous integration involves merging code changes from different developers into a single codebase as often as possible. This allows teams to identify and fix problems early in the development process, rather than waiting until the end.

Continuous delivery takes this a step further by automating the process of testing and deploying code changes. This allows teams to deploy code updates more quickly and reliably, and can help reduce the risk of errors.

Here are some key benefits of using CI and CD:

1. Faster deployment: By automating the process of testing and deploying code changes, teams can deploy updates more quickly and efficiently.

2. Improved quality: By identifying and fixing problems early in the development process, teams can improve the quality of their code.

3. Increased collaboration: CI and CD practices encourage close collaboration between development and operations teams, helping teams work more efficiently and effectively.

4. Reduced risk: By automating the testing and deployment process, teams can reduce the risk of errors and improve the reliability of their software.

There are many tools that developers from across the spectrum have to come love and adopt. However, in Government of Canada setting, Azure Pipelines, a sub-tool of Azure DevOps, have to come to become a leading tool for developing CICD.

Azure Pipelines is a continuous integration (CI) and continuous delivery (CD) platform from Microsoft that allows you to build, test, and deploy your code. It is a cloud-based service that can be used to automate the build, test, and deployment process for a wide variety of applications, including .NET, Java, Node.js, and more.

Azure Pipelines outperform other tools such as Google's Cloud Build, or GitLabs mostly because of the following features:

1. Multi-platform support: Azure Pipelines supports a wide range of platforms and languages, including Windows, Linux, macOS, and Docker containers.

2. GitHub integration: Azure Pipelines integrates seamlessly with GitHub, allowing you to trigger builds and deployments automatically when code changes are pushed to your repository.

3. Build and release pipelines: Azure Pipelines allows you to create build pipelines to automate the process of building and testing your code, and release pipelines to automate the process of deploying your code to different environments.

4. Customization: Azure Pipelines allows you to customize your pipelines using YAML files or the Azure Pipelines user interface.

5. Monitoring and reporting: Azure Pipelines provides real-time monitoring and reporting tools to help you track the progress of your builds and releases.

6. Azure Pipelines can be a useful tool for automating the CI/CD process for your applications. I hope this helps! Let me know if you have any questions about Azure Pipelines.

7. Vast template library: There are so many different technology variations, and building a CICD pipeline for each differs in one way or another. Azure Pipelines come with thousands of well-organized templates that can simplify developing of pipelines for any technology.

***CIO Action**: Mandate Chief Enterprise Architect to use of a CICD Pipeline is a check-point in Architectural Governance.*

Step 5: Monitor and measure your progress

Monitoring and measuring progress is an important aspect of implementing DevOps at scale in your organization. By tracking key performance metrics, you can identify problems, improve your processes, and measure the success of your DevOps efforts.

Here are some ways you can monitor and measure progress when implementing DevOps at scale:

1. Use monitoring tools: Tools like Grafana, Splunk, and New Relic can help you monitor your systems in real-time and track key performance metrics. These tools can alert you to problems and help you identify trends over time.

2. Track key performance indicators (KPIs): Identify the key performance indicators (KPIs) that are most important for your organization and track them regularly. This could include metrics such as deployment frequency, lead time for changes, and mean time to recovery (MTTR).

3. Use dashboards and reports: Use dashboards and reports to visualize your performance data and make it easy to track your progress. This can help you identify trends and areas for improvement.

4. Conduct regular reviews: Schedule regular reviews to discuss your progress and identify areas for improvement. This could be an opportunity to review your KPIs, share best practices, and discuss any challenges you are facing.

By monitoring and measuring your progress, you can stay on track and continuously improve your DevOps practices

---

***CIO Action**: Establish DevOps adoption as in performance management plan for middle management of application development teams.*

Overall, implementing DevOps in IT organization is a game changer! In fact, many organizations are experimenting with DevOps. However, the true value for the organization can be realized by taking focused effort to accelerate adoption of DevOps-at-Scale.