WHITEPAPER

DevOps the NetApp Way

Definitions and best practices for DevOps by NetApp IT

NetApp

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Overview

DevOps has been practiced for some time now, but it's still a relatively new methodology for many large global enterprises. DevOps is more than just a trend, and it's not going away anytime soon. It's an evolutionary step in the way we develop and operate software, which brings many benefits.

Although different teams in NetApp IT had started to adopt DevOps practices, we lacked a common approach to how DevOps teams should be set up and how they should operate to meet a true definition of DevOps. We needed to understand what differentiates a DevOps team from a regular development team and what processes DevOps teams should follow. Therefore, we created our definition of a DevOps team based on ten rules.

As with any journey, we will continue to evolve our rules and definitions to make DevOps successful here at NetApp.

Ten Rules for DevOps Team

According to our definition, any team that applies the following 10 rules can be called a DevOps team. Teams that don't follow these rules are called regular development teams.

A DevOps team must:

1. Own one or more products.
2. Be responsible for the development, QA, and operations of the products they own.
3. Have one assigned team lead, who leads and contributes to the team.
4. Not be smaller than three people and not larger than eight people.
5. Be able to change the size and scope of the team over time.
6. Apply agile methodologies to continuously enhance the products they own.
7. Actively observe their products to assess and improve the operational health.
8. Embrace failures as opportunities to learn and improve.
9. Be able to release new versions of their products independently and at any time.
10. Maintain collective ownership of the products and practice knowledge-sharing to prevent single, silo-like domain ownership.

"Every decade has its own trendy software methodology. While they all seem to feel better, history proves them to be ineffective. However, we see continued evidence that DevOps delivers value, and for six consecutive years, we have statistically verified key capabilities and practices that help organizations improve their software development and delivery using DevOps methods. DevOps is not a trend and will eventually be the standard way of software development and operations, offering everyone a better quality of life."

—2019 Google State of DevOps Report

About the Author

Florian Lippisch leads Services Enablement Solutions in support of enterprise application solution management and delivery for the worldwide IT organization at NetApp.

1. Product ownership

A DevOps team owns one or more products.

A product is typically a microservice or the front end of an application. A DevOps team truly owns one or more such products. That ownership comes with both the responsibility and the power to drive the future of the products.

The consumers of products have specific requirements that the DevOps team must understand and fulfill. At the same time, products must also adhere to platform-specific guidelines, such as those from governance and review boards that evaluate architectural proposals. Within these boundaries, the team has the freedom to design and implement the product to their preferred design.

Permitted DevOps products—that is, services and tools—owned by a DevOps team are listed in the Viable Products section of this paper. A team can own multiple viable products. Nonviable products must be managed in a different way, which can distract teams if they must operate in different modes.

2. Team responsibilities

A DevOps team is responsible for the development, QA, and operations of the products they own.

In NetApp IT, a DevOps team is composed of developers who actively develop the products they own in a way that adheres to high-quality standards. Automated tests are the ideal way for teams to ensure that these standards are met. In certain cases, such as UI products, automated tests can't cover everything. In these cases, the team can include QA specialists who write and execute manual test scripts. This means that some DevOps teams might not contain dedicated QA resources at all. They can consist only of developers who write the necessary automated tests.

The DevOps team is also responsible for the operations of the products they own. This responsibility includes the following tasks:

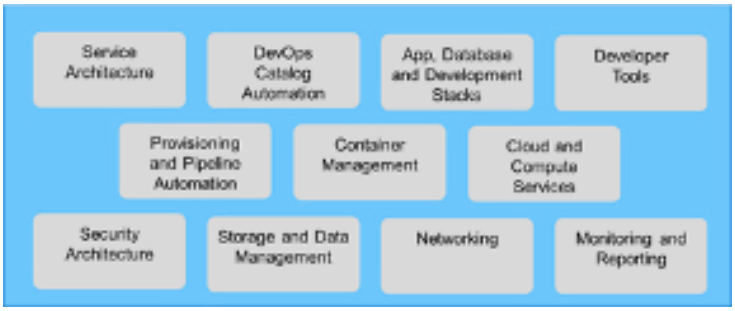


Figure 1) The Platform Team Inside NetApp IT.

- **Provisioning the required environments.** Typically, this task is as simple as requesting a new pipeline in the CloudOne DevOps management console.

  If the team has special requirements, it is up to the DevOps team to work with the Platform Team (shown in Figure 1) to get what they need.

- **Executing the build and deployment pipelines.** The team owns the configuration of their pipelines and the execution of the deployment.

- **Observing log files and health information of the environment** to detect active and potential problems, then mitigate them. As part of this task, teams should improve the observability of their products where possible. This includes the logging of more details or getting rid of false-positive error entries. This active monitoring and continuous improvement help not only the DevOps team, but also the L2 teams that support the products, because it reduces the workload and number of incidents.

- **Responding to incidents** and consumer inquiries about the products they own.

Today, responsibility for operational activities does not necessarily mean that the team must have the internal resources to perform the tasks. The NetApp Operational Level 2 (L2) support team in IT operates separately from the delivery teams, often leveraging shared resources to provide coverage 24/5. DevOps teams can collaborate with the L2 team to carry out these tasks. That is, the L2 team can execute the production deployment or handle incoming incidents before they reach the DevOps team. However, ultimate accountability remains with the DevOps team.

3. Team lead

A DevOps team has one assigned team lead, who leads and contributes to the team.

Each team has one assigned lead who manages the members of the team. She or he represents the team and acts as the primary point of contact for others. It's important for the lead to also be a member of and contributor to the team. The lead has the following responsibilities:

- Primary point of contact for the DevOps team.
- Facilitate the daily stand-up meeting.
- Facilitate the grooming sessions for the technical review of the user stories.
- Oversee the Kanban board and ensure it is up-to-date and the necessary data points are entered.
- Represent the team in the DevOps status meetings, providing updates as required.
- Assess and monitor the team's capacity to do the work in the backlog and the impact of incoming projects, and work with the management team to obtain resources as required.

For larger DevOps teams, some of the above responsibilities can be delegated to a dedicated scrum master[1] who supports the team. However, the team lead remains accountable.

The team lead must ensure that there is a backup person on the team who takes on these responsibilities when the lead is unavailable.

4. Team size

A DevOps team should not be smaller than three people and not larger than eight people.

Large teams become inefficient by nature, as the number of communication paths increases polynomially[2]. Therefore, in NetApp IT, we want DevOps teams to remain small, not exceeding eight people on a permanent basis (Table 1).

For peak workloads during projects, it may be acceptable for a team to grow over the recommended size, but it is an indicator that the team has grown too big and should be restructured.

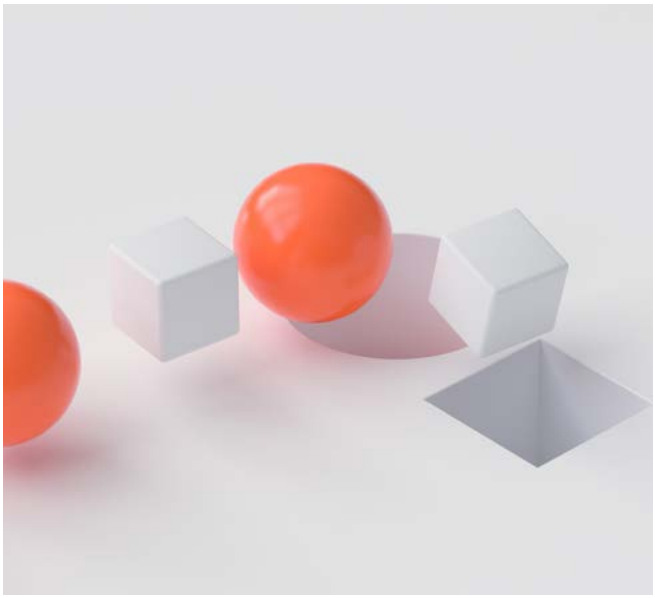
Teams with less than 3 people may be a "waste" of lead resources. We recommend increasing the scope and size of that team for optimal use.

Table 1) Recommendations on DevOps Team Size.

Size	Recommendation
1	• A team that relies on a single person is too small. If that person is unavailable, no support is available, and knowledge may be lost. • Consider merging the team with an existing team.
2	• A two-person team is small but functional. Teams that are just starting can run like this for a limited time. They should not be responsible for any production services. • Consider merging the team with an existing team or increasing its scope.
3	• A three-person team is functional but still small. The team can remain at this size permanently, and with a good lead could probably take on more responsibilities. • Consider increasing the scope of the team.
4-7	• Four to seven people is the optimal team size.
8	• An eight-person team is at the upper limit of manageability and efficiency. This size is workable if the workload is expected to be reduced in the future, for example, when a project is complete. • Consider decreasing the scope or splitting the team.
9+	• A team of more than eight people is too large and can no longer operate efficiently. • Consider decreasing the scope or splitting the team.

1 Even though we do not follow the Scrum model explicitly, we practice certain ceremonies and as such the title of a "Scrum Master" is still viable for a team support role.

2 Combinatorial Explosion (Communication)



5. Team adjustments

A DevOps team size and scope can be changed over time

Workloads for products are variable over time. Some products may require an occasional change, while others may need to be constantly enhanced. The product landscape also changes as new products are required or others reach end of life. There is no perfect size for a team for all times; size and scope must be adjusted to accommodate needs over time.

To optimize the use of our resources in NetApp IT, the management team reviews the size and scope of each team regularly. Taking various metrics into account, along with the outlook and overall priorities of the business, the management team may adjust the size or scope of a DevOps team in collaboration with the team lead.

Adjustments may result in the reassignment of a resource or moving the ownership of a product to another team. Such a shift is disruptive for the existing teams, so it should be done only after careful assessment of the situation and as infrequently as possible.

The consistent use of the same technology and the same processes enables the flexibility to make these adjustments. Our teams use the CloudOne Business Resilient Applications (COBRA) framework, deviate from it only when necessary.

6. Agile practices

A DevOps team applies agile methodologies to continuously enhance the products they own

DevOps teams need to be agile and respond quickly to upcoming needs and changing environments. They are not dedicated to a project, only to the products they own, which they are continuously improving and delivering value, albeit in small increments.

In NetApp IT, the preferred method for DevOps teams is the use of a Kanban board, where work will flow from the left to the right. Teams may also apply Scrum-like methodologies such as sprints, sprint planning, or grooming sessions.

DevOps teams should:

- Split work into small increments of deliverable features.
- Release changes often, e.g. as they are completed or on a weekly basis.
- Avoid spending a lot of time on long-term planning of the work. Instead, focus on maintaining a healthy backlog for the next 30 days. Anything beyond that is frequently subject to change.

Waterfall projects still exist

NetApp IT still runs some projects in a waterfall manner. Although this method is addressed separately, our DevOps teams do need to contribute to these projects. In these cases, the teams should still handle all the work as items on the Kanban board and prioritize the work to align with the project schedule. It is up to the team to use good judgment to support the project in the best way possible, while maintaining the agile workflows of the team.

7. Product monitoring

A DevOps team actively observes their products to assess and improve their operational health

At NetApp, a DevOps team owns products that they are responsible for throughout their entire lifetime. This ownership includes the need for the DevOps team to continuously observe their products by checking logs and health data.

Before a product goes live, the team should make sure that the product produces proper log information that can be analyzed with the tools provided by our internal CloudOne platform. No matter how good the initial logging is, there will always be room for improvement after the product goes live. Therefore, it is very important for the teams to spend sufficient time on continuously improving the observability of their products.

Teams should:

- Proactively review log and health data on a weekly basis.
- Report the operational health in reoccurring DevOps team status meetings.
- Improve log details where a gap is identified.
- Remove false-positive log error entries.
- Address error log entries by either fixing the problem or, if there is not really an error, downgrading its severity.
- Consider implementing active notifications that the service sends should a critical process fail; for example, sending an email message to the team if something is seriously wrong.

8. Failures are viewed as opportunities

A DevOps team embraces failures as opportunities to learn and improve

In the DevOps world, failure can't be avoided entirely. We expect failures to happen, and we build systems and processes to deal with them. DevOps teams don't punish failures, especially those that lead to either avoiding changes or hiding things that went wrong. Instead, DevOps teams openly share when something goes wrong, allowing everyone to learn from a mistake so that we can collectively prevent it from happening again.

Products that are built and maintained by DevOps teams also need to anticipate the possibility of failures of other systems they connect to. When such a failure occurs, the products need to deal with it as gracefully as possible. Failures happen. Recognizing this fact, DevOps teams prepare for them, learn from them, and recover from them quickly.

9. Release independence

A DevOps team can release a new version of their product independently and at any time

Products owned by a DevOps team, especially microservices, are used by other consumers. The way these consumers use the product is treated like a contract between the DevOps team and the consumers. This contract must adhere to compatibility guidelines. These guidelines demand that the owning team of a product must not break the contract when they release a change. It is the responsibility of the DevOps team to test the validity of the contracts in each release, ideally with fully automated tests.

On the other side, consumers of services should react gracefully to changes. Ideally, the consumer can work with different versions of services without the need for manual intervention.

Read about compatibility and ways for compatible extensions in our COBRA REST API Guidelines.

By adhering to these requirements, DevOps teams can release updates to their products at any time, without waiting for other dependencies or approvals.

10. Collective ownership

A DevOps team maintains collective ownership of the products and shares the knowledge with the team.

Any software that is created in an enterprise the size of NetApp is likely to be used for a long time. The developers who originally created that software may not be the ones who maintain it throughout its lifetime. Even though they are set up for long-term ownership of products, DevOps teams also see developers leave and take with them the knowledge that they have acquired.

That's why it's important for DevOps teams to collectively own their products and avoid building knowledge silos within the team. Every developer on the team should be able to work on all products, and every QA engineer should be able to test all parts of the product. Sharing the knowledge among the members should be part of the DNA of the team.

As the collective owner of the products, the team members should work together to groom new stories and to drive the direction of the development and architecture of the products.

DevOps inside NetApp IT
Viable products

Not every product that NetApp IT manages is a viable product for a DevOps team. Our legacy systems are an example. They were architected for a different model of development and operations, so it's best to continue to manage them as done in the past.

Viable DevOps products follow certain characteristics that are necessary to manage them in a DevOps fashion. One example is the need to be able to release with a high frequency. A high proportion of automated tests is necessary to have the confidence that a change won't break the product or other dependent products.

Generally, we differentiate between front-end and back-end products. DevOps teams typically specialize in one of the two categories, owning only products of that category.

Front-End products

Types of front-end products include:

- Web applications running in a browser on a desktop, tablet, or mobile device
- Mobile-native applications installed on a mobile device or tablet
- Desktop applications that include an auto-update mechanism

The requirements are:

- The product needs to be small enough to be managed by a single DevOps team, or provide a mechanism that allows the modularization in a way that multiple teams can own one or more modules that can be handled like a standalone product.
- The product must have a high degree of automation and end-to-end tests so that the team can ensure that changes don't break existing behaviors.

Back-End products

Types of back-end products include:

- Microservice—a stand-alone product that offers APIs to its consumers.
- Proxy services—a microservice that provides access to another service or data source without adding additional business logic.

The requirements are:

- The API design must follow the NetApp RESTful API Guidelines.
- Automated tests must be implemented to ensure that changes don't break the logic. Services should have a test coverage of at least 95%, ideally 100%.
- A published API is a contract with consumers that must not be broken. There are compatible extensions to an API, as described in the NetApp RESTful API Guidelines, but the key is to ensure that changes or updates don't break any application or service that consumes it. No application or service should be forced to update to a new API version.
- Teams need to monitor the usage of their APIs to figure out when a legacy API is no longer used so that it can be retired.
- Back-end products need to be small enough to be managed by a single DevOps team.
- Back-end products are standalone products that could potentially have more than one consumer. They should never be built specifically to be used by only one consumer.
- Back-end products must have sufficient documentation available to any other team that wants to consume the service.

Nonviable products

Any product that does not fit the description of a viable product is considered nonviable. Examples include:

- Monolithic applications, such as NetApp's legacy ERP and quoting systems
- Customized SaaS applications such as Ascend
- Any tool that needs to be tested and deployed manually after each change

Front-End and Back-End Product Ownership

Front-end and back-end products should not be owned by the same team. The only exception is the UI Gateway service, which can be owned by the corresponding front-end team. The reasoning behind this rule can be explained by a 1967 quote from Melvin E. Conway, known as Conway's Law: "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."

We learned from experience that teams owning both the UI and the services creates a tightly coupled system that is closer to a monolithic architecture. Each service needs to be a product on its own, with an independent lifecycle. The team can ensure desired outcomes by separating the consumer and producer.

A team can own more than one product, so that there is a meaningful workload for the team. As of now, there is no upper limit for how many products a team can own. The limit is currently defined only by the size of the team.

The number of products should keep the team busy at about 80% utilization. The remaining 20% of time is required for the team to handle peak workloads, to perform administrative tasks, or use the time for training and learning. After all, it's important for teams to remain healthy and to operate with a good work-life balance.

Composing the right products for each team is an art. The number of teams is limited by the number of available team leads and resources. The management team meets regularly to decide on changes to the team landscape based on current and future workloads, resource availability, and inflow of new products and teams.

Ideally, the team's products have something in common. For example, they might address the same business domain or integrate with the same core enterprise systems, such as ERP or CRM. Given the nature of small teams, this will remain a balancing act that requires ongoing refinement.

Conclusion

DevOps is a journey. The rules in this document apply to where we are in our journey today. As we further mature and develop our DevOps skills, we will continue to evolve these rules. Measuring the success of our model and processes is vital to advance us in our journey. Equally important is everyone's feedback, suggestions, and contributions to evolving this model. If you have feedback, we are happy to hear your thoughts. Email us at ng-CloudOne-DevOps@netapp.com.

About NetApp

In a world full of generalists, NetApp is a specialist. We're focused on one thing, helping your business get the most out of your data. NetApp brings the enterprise-grade data services you rely on into the cloud, and the simple flexibility of cloud into the data center. Our industry-leading solutions work across diverse customer environments and the world's biggest public clouds.

As a cloud-led, data-centric software company, only NetApp can help build your unique data fabric, simplify and connect your cloud, and securely deliver the right data, services and applications to the right people—anytime, anywhere.

NetApp

©2020 NetApp, Inc. All Rights Reserved. NETAPP, the NETAPP logo, and the marks listed at http://www.netapp.com/TM are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners. WP-7334-1120

Copyright Information
Copyright © 2020 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:
THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

Data contained herein pertains to a commercial item (as defined in FAR 2.101) and is proprietary to NetApp, Inc. The U.S. Government has a non-exclusive, non-transferrable, non-sublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b).