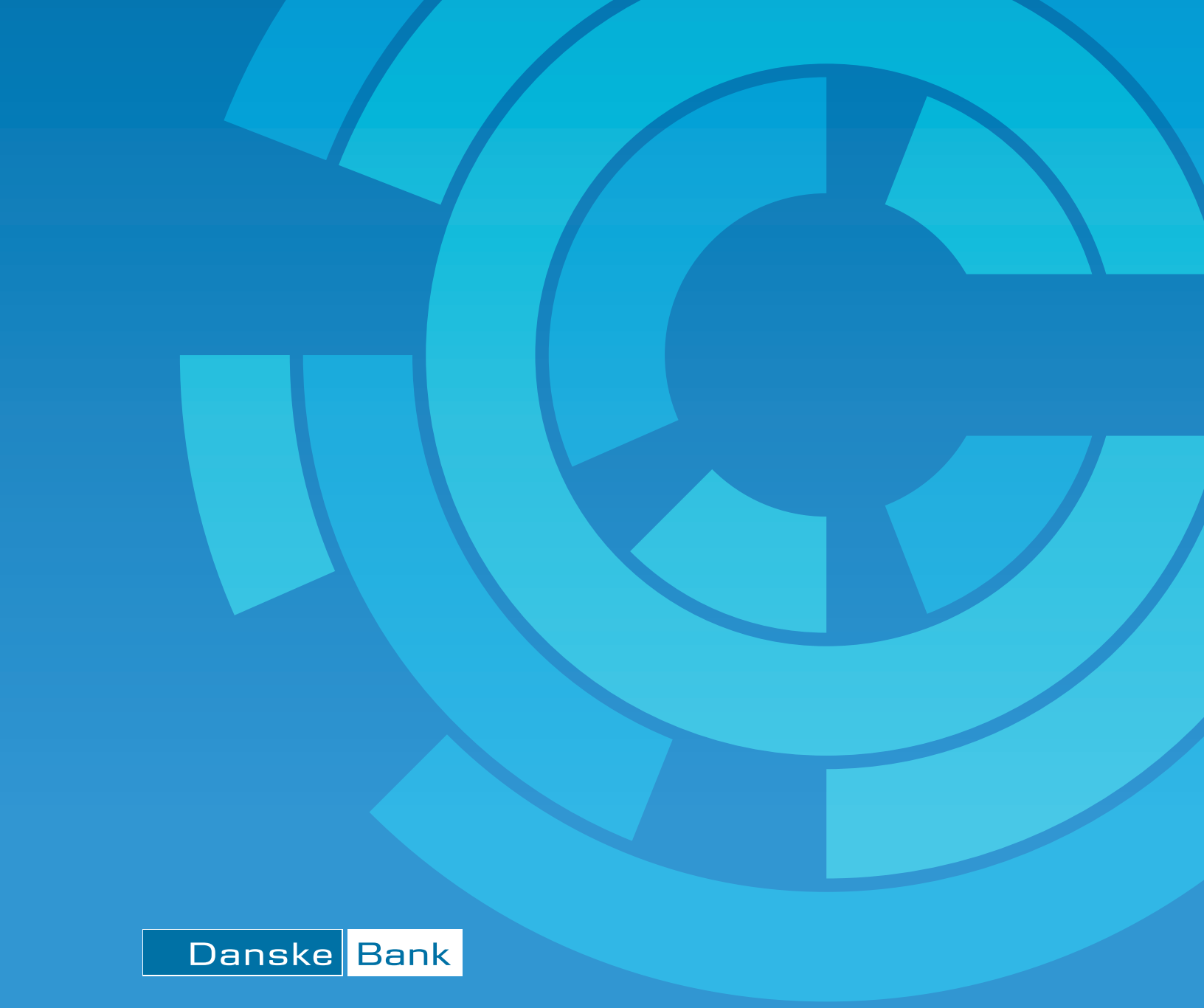

Danske Bank

DANSKE BANK CHEF JOURNEY

3 Years in Green Field Automation

Liudas Baksys - Development Team Manager - Danske Bank

Published July, 2019

This Danske Bank customer journey was captured from a Chef Community Summit presentation in October 2018. Liudas Baksys who runs the Linux Development Team at Danske Bank explained the bank's three year journey with adopting Chef starting from configuration management and leading to a comprehensive and engaging Chef Community serving the entire Linux infrastructure at the bank.

Danske bank's journey was progressive and can be broken down into three main phases.

1. **Configuration Automation** - Converting manual and long process into an automated, repeatable process.

2. **Iterative Improvement** - Making further automation enhancements not only to automate configuration processes, but validation processes as well.

3. **Building Community** - Building a joint sense of ownership with a broader set of users was extremely important for the bank's continued automation success.

Danske Bank's lessons learned from their Chef journey revolved around:

1. **User Focused:** Having a strong focus on Chef users and making sure they were successful.

2. **Standardization:** Creating a set of standards early in the process is extremely important.

3. **Testing:** Testing frequently and early provides confidence of changes made and their impacts.

Here is a more in depth look at Danske Bank's Chef adoption journey.

ABOUT DANSKE BANK

One of the largest financial institutions in the Nordic countries, Danske Bank is currently present in 16 countries with 245 branches and over 20,000 employees, serving personal, business and institutional customers and in addition to banking services, offer life insurance and pension, mortgage credit, wealth management, real estate and leasing services.

Phase One:
CONFIGURATION AUTOMATION

Danske Bank's Chef journey began in 2015, with the formation of a new team initially responsible for maintaining a few Linux servers, but shortly thereafter retasked with productizing the Linux offering in the bank's private cloud.

Before Chef, the process for provisioning a Linux server was manual, time consuming, and difficult to scale effectively. In order to provision a Linux server, a request would first need to be submitted using a manually created Word Document. The document would then be given to the virtualization team, responsible for building a container that was then again handed over to the Linux/Virtualization team who would install the operating system, configure different agents, adjust compliance settings, and perform simple verification tasks. Because there was no programatic way to apply or validate server configurations, setup tasks were manual and error-prone, and verification was rudimentary -- initially no more than ensuring a count of installed packages on a newly-configured system matched the expected number in the Word Document.

The Bank's automation journey with Chef then began with the team converting the Word Document into code by creating Chef Cookbooks that translated each requirement into executable Chef Recipes. This project resulted in a collection of these Cookbooks that defined an automated, repeatable process for applying a base OS configuration, installing required software and agents, and hardening the system in accordance with any compliance requirements. What once required multiple handoffs and manual effort could be achieved with Chef at the push of a button.

Phase Two:
ITERATIVE IMPROVEMENT

Once Danske Bank had created a baseline for their automation, the next phase of their journey centered on how they could improve the efficiency of their automation processes to iterate with even higher velocity.

The first challenge became how to deal with the single-threaded nature of server configuration processes. For systems and applications that needed to be clustered together, provisioning servers one-by-one meant that not all the data required for their configurations would be available until the entire cluster had been instantiated, so that members could open firewall ports, establish connections, and so on. To address this, the team started expressing any dynamic or tunable configuration parameters as attributes, which are variables that can be used to alter the behavior of Chef Cookbooks without needing to write a separate recipe for each permutation. Implementing attributes in this way allowed Danske Bank to make further efficiency improvements, such as allowing individual stakeholders to define which package repositories are required in each implementation without needing to write any net-new Chef code. They simply needed to assert which repositories were required as attributes, and the Cookbooks provided by the Linux team would take care of the rest.

As Danske Bank continued to expand its use of Chef, it became apparent that they needed a way not only to automate their configuration processes, but their validation processes as well. For example, when the team implemented standardized naming conventions for Chef Cookbooks, they needed a way both to audit existing content to ensure everything was updated to conform to the new standard, and enforce those standards on any new content created going forward. This was achieved by implementing InSpec to codify those requirements such that they could be executed as automated tests. This way, any time a cookbook was created or changed, any deviation from expectations could be captured immediately, preventing costly rework later on.

Phase Three:
GROWING THE COMMUNITY

Automation profoundly changes the way work is done. Instead of applying updates in an ad-hoc, system by system fashion, changes are instead made to the policy that manages their collective configurations. The obvious benefit of this shift is that less time must be spent repeating work, and that work is applied consistency with fewer opportunities for configuration drift due to human error.

Less obvious is how that shift can impact parts of an organization not yet automating their work. At Danske Bank, teams responsible for managing the day-to-day operation of applications found that when they identified and attempted to remediate configurations on the servers they managed, their changes would be undone the next time Chef was run on those systems. This is because Chef had become the canonical source of truth for those configurations, but the teams responsible for the resulting environments were still updating configurations by hand. From Chef's perspective, these changes constituted a drift from desired state, and from the end users' perspective, Chef was undoing their hard work, and applying outdated configurations.

This friction led to teams finding ways to circumvent or outright disable Chef, which could have undone all the efficiency gains made in previous efforts if left unchecked. The Linux team understood that fighting these fights each time they showed up did not lead anywhere, or solve the underlying adoption problem. This was an inflection point in the adoption journey where Chef was no longer an internal tool for the Linux team alone, but had grown such that impacted teams across the organization. Therefore, the Linux team would have to work with other bank system users to educate them on how to use Chef effectively, and ultimately drive adoption beyond the core team.

And so, the Linux team did just that! They started working with users to educate them on how Chef works; what it does, and what it doesn't do. Workshops with different groups were delivered to see how Chef could help them, or at least not get in their way.

As an outcome of this educational effort the team created a Chef Development Guide and also provided a CICD pipeline for cookbook development. The team provided a Cookbook Template, allowing those that wanted to create their own cookbooks could do so following the bank's standards; including Naming, READ ME Files, change logs, sample tests, and Chef Kitchen files to test on the Bank's Linux image including all necessary compliance settings.

Some users became so comfortable with Chef, they even started to put Pull Requests to the cookbooks to help improve them, in order to add features they needed. By consequence of this engagement, Danske Bank essentially created a Chef Community within the bank.

3 YEARS IN RETROSPECT

Looking back through the first three years, Danske bank faced different challenges, admittedly they could have done some things better or even earlier. Some things were overdone as well. The bank learned a lot, and they consider themselves now Master Chefs where they can today easily review complex resources such as Library or Helper. The main source of documentation for the bank is now Chef Source Code, considered as the only source of truth.

Three advice points Danske Bank Chefs give from their experience:

1. **Focus on Users** - "Users are the ones you work for, they generate the value. You should work with them and help them achieve their goals. Not just having the Chef tool for yourself, but by working with them, the users will start helping and adopting Chef as their own."

2. **Have Standards** - The bank learned the hard way by overcoming the mentioned challenges they had to create standards. If they had created the standards early on, they might have been able to move faster. Either way, having standards is important.

3. **Test Early** - The early testing helped the bank guarantee that their cookbook rework tasks were much easier, and provided them the confidence that if any changes were needed they would not break things. Test Often and Test Constantly.

To find out more about other Chef customer successes visit our Customer Page: https://www.chef.io/customers/

ABOUT CHEF:

Chef is the leader in DevOps, driving collaboration through code to automate infrastructure, security, compliance and applications. Deployed broadly in production by the Global 5000 and used by more than half of the Fortune 500, Chef enables coded enterprises to express infrastructure, security policies and the application lifecycle as code, modernizing development, packaging and delivery of any application to any platform.

For more visit https://www.chef.io