

Efficient automation with Ansible

Table of contents

Disclaimer.....	3
Version history	3
1. Introduction	3
2. Ansible as the common language for automation.....	4
3. Plan efficient improvements with automation	5
3.1. Let's discover what we have first.....	6
3.1.1. Popular event-centric practices	6
3.1.1.1. What is "Event Storming" discovery practice?.....	7
3.1.1.2. Example of concept-to-technology "zoom-in" with events.....	8
3.1.2. A traditional lens for domain discovery: "Domain Storytelling" practice	10
3.1.3. Sample project for "Sample Org": Motivation Mapping.....	12
3.1.4. Sample: The initial discovery	13
3.1.4.1. Sample: Event Storming for "Deployment of app new version"	14
3.1.4.2. Sample: Domain Story for "Deployment of app new version" (current state).....	15
3.2. What are our issues? What are the bottlenecks in our processes?.....	16
3.2.1. What is "Metrics-based Process Mapping" practice?	16
3.2.2. Sample: MBPM – today's process flows, owners, metrics	17
3.2.3. Sample: Issues and risks today	19
3.3. Are we preparing to deliver according to the real needs?.....	20
3.3.1. What is "Target Outcomes" discovery practice?	20
3.3.2. Sample: Strategy for the first delivery iteration.....	20
3.3.2.1. What is MVP?	20
3.3.2.2. Sample: MVP ideas and choice	21
3.4. Is everything ready to start the first delivery iteration?	22
3.4.1. What are "User Story Mapping" and "Value Slicing" practices?	23
3.4.1.1. Terminology.....	23
3.4.1.2. "User Story Mapping" and "Value Slicing"	24
3.4.1.3. Are your User Stories ready to start implementation?	25
3.4.2. Sample: Epics, features, and initial User Story Map	25
3.4.2.1. Sample: Domain Story for "Deployment of app new version" (desired state)	26
3.4.2.2. Sample: Initial User Story Map	27

3.4.3. What features go into the first delivery iteration?.....	29
3.4.3.1. Practices of prioritization.....	29
3.4.3.2. Sample: Scope of MVP defined, Value Slices.....	30
3.4.3.2.1. Sample: Kano Model applied.....	30
3.4.3.2.2. Sample: Value Slices.....	32
4. Deliver automation efficiently.....	33
4.1. Introducing the “Delivery Loop” and agile practices.....	33
4.1.1. Backlog refinement (options practice).....	34
4.1.2. “Increment Planning” or “Sprint Planning” (delivery practice).....	34
4.1.3. Daily standup (foundational practice).....	35
4.1.4. Showcase or demo (delivery practice).....	35
4.1.5. Retrospectives (foundational practice).....	35
4.1.6. “Definition of Done” (foundational practice).....	35
4.2. Sample: Let’s deliver according to the plan of our first iteration.....	35
4.2.1. Sample: Decisions for iteration.....	36
4.2.2. Sample: Action plan for iteration.....	36
4.2.3. Sample: During the iteration.....	37
4.2.4. Sample: The Showcase (demo) for iteration.....	41
5. What Red Hat offers to support customers?.....	42
5.1. Better confidence in your automation or container adoption journey with Red Hat Open Innovation Labs.....	42
5.2. Lots of moving parts at scale? Consider using Red Hat Ansible Automation Platform.....	42
6. Conclusion.....	43
7. About the authors.....	44
Appendix A. Event Storming for “App in use – requesting access privileges”.....	45
Appendix B. Event Storming for “Patching the Linux box of app”.....	47

Disclaimer

This guide based on a case study describes solely personal, individual opinions of this guide's authors. The information in it is given "as is," without any warranty. It can be outdated at the time when you read this. The authors are not liable for anything related to this information. The authors **guess** that this guide can be helpful in general.

Version history

Version	Date	Description	Authors and Contributors
1.0	12 October 2023	First public version	Authors: Nicholas Wong, Gineesh Madapparambath, Michael Knyazev Contributors: Donna Benjamin, Jeremie Benazra, Denis Mikhalkin, Frederick Son, Valentine Schwartz

1. Introduction

In this guide based on a case study, we are aiming to support the widest community of automation specialists and people who want to actually benefit from automation. This guide is for DevOps Engineers and Architects as well as for Product Owners, IT and Operations Managers. We designed it in a way that many of you can start using the mentioned resources and new knowledge, the next day, after reading this document. The style of this guide is very practical. The introductions into corresponding practices and focused recommendations are alternating with a real-life "sample project" of automation which is progressing from start to its Minimum Viable Product (MVP) delivery throughout the chapters. A possible MVP implementation can be found in the following personal repository published under MIT license:¹ <https://github.com/mikhailknyazev/automation-samples>

Everyone can find something unique related to their particular environment in this guide. If you ask us what is unique about it, we would say it is because it shows how engineering, product and agile practitioners can work on automation projects together in a structured way, enjoying harmonious work ethics and delivering valuable outcomes for customers.

The related offerings by Red Hat are grouped in a special section named "What Red Hat offers to support customers?" in the end.

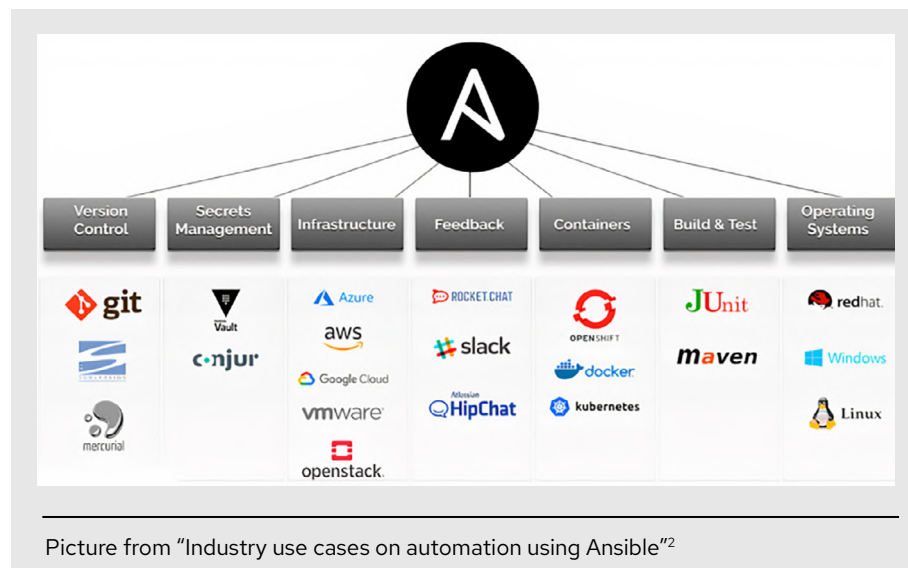
¹ A possible MVP implementation (personal repository published under MIT license)
<https://github.com/mikhailknyazev/automation-samples>

2. Ansible as the common language for automation

The need for orchestration in complex IT environments is not new and you'll find that many ecosystems already have their own orchestrators. Tools like OpenStack's Heat, Amazon's CloudFormation, Azure Resource Manager, Jenkins or HashiCorp's Packer, and Terraform are all about orchestrating tasks and realization of the "Infrastructure-as-Code" approach.

But what are the chances that the customer is going to be able to constrain their orchestration to just one environment? That's where Ansible® comes in. Ansible's library of modules and easy extensibility make it simple to orchestrate different "conductors" in different environments, all using one structured Ansible syntax. When choosing what to use when implementing new scripted functionality, Ansible is often an efficient replacement for "bare" Bash/PowerShell/Python scripting.

Ansible works with the customer's existing SSH and WinRM infrastructure, so there are no extra network ports to open. Ansible works as an universal "glue" for IT artifacts in organizations. It can work with compute nodes, storage devices, networks, load balancers, monitoring systems, web services, and other devices. For example, you can add or remove servers from your load balancing pool and disable monitoring alerts for each machine that is being updated.



We've seen it many times in real-life projects – Ansible-driven infrastructure provisioning, deployment, and testing as well as operations automation for Kubernetes and Red Hat® OpenShift®. You can even add Kubernetes as an event source into Event-Driven Ansible (EDA) to trigger automation as desired.³ You can deploy Ansible Automation Platform on Red Hat OpenShift.⁴

Below are some examples of automation ideas which might be applicable to the customer's organization:

- ▶ Reduce drift, outages, and downtime by ensuring consistent system settings
- ▶ Minimize human error by automating operational tasks
- ▶ Reduce skill requirements across multiple domains
- ▶ Establish and maintain desired baselines for systems to meet compliance
- ▶ Lower risk of security breaches and other security-related incidents through patch management

² Industry use cases on automation using Ansible <https://www.linkedin.com/pulse/industry-use-cases-automation-using-ansible-anand-kumar/>

³ "Kubernetes meets Event-Driven Ansible" <https://www.ansible.com/blog/kubernetes-meets-event-driven-ansible>

⁴ "New reference architecture: Deploying Ansible Automation Platform 2 on Red Hat OpenShift" <https://www.ansible.com/blog/new-reference-architecture-deploying-ansible-automation-platform-2-on-red-hat-openshift>

Speaking about enterprise security in particular – it is not a homogeneous entity. It is normally a portfolio of multi-vendor solutions run by disparate and often siloed teams. With so many different layers, automation proves to be efficient in helping security operations teams to integrate and share accountability and chain multiple security technologies together.

IT team members often find themselves repeating the same tasks day after day. Examples include:

- ▶ A system engineer builds servers and virtual machines, installing packages, patching old systems, configuring firewall devices, and more.
- ▶ A developer struggles to build a coding environment every time a new version of the programming language or software library is released. They are also spending a lot of time testing code and waiting for test results.
- ▶ A database administrator spends valuable time provisioning disk space and configuring storage devices, experiencing delays provisioning a new database server, or facing issues with network or system readiness.
- ▶ The operation team struggles with a flood of events and alerts, spending their time filtering out false positives, resulting in lost productivity.
- ▶ The security team works hard to fix violations and make sure systems comply with relevant security standards.

3. Plan efficient improvements with automation

We want to add more value and spend less effort. Before jumping into implementing automation, it is worth pausing to ask a few questions:

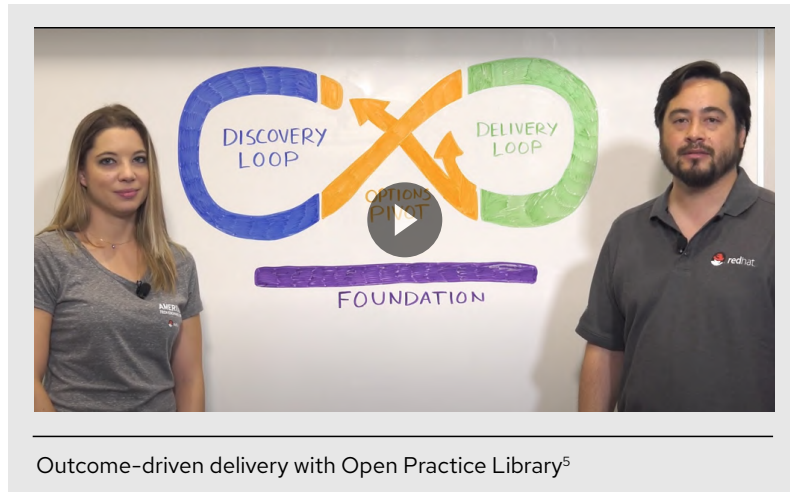
- ▶ How complex is the use case?
- ▶ Can I reduce human error?
- ▶ Can I reduce the deployment time and speed up my tasks?
- ▶ How frequently am I doing this task?
- ▶ How much time can I save by automating this task?
- ▶ What is return on investment? Can I save some money?

It is a common misunderstanding that the responsibility for finding use cases and implementing automation only falls to the systems team, platforms team, or infrastructure team. When we explore our work environment and day-to-day tasks, we will find thousands of tasks that we can automate using Ansible. It could be the database team managing database servers and instances, the network team handling network operations, or the application team who wants to deploy their application updates more effectively. Implementing automation in the environment is a collaborative journey, and we need support and guidance from different teams.

There are well-known open practices that facilitate taking a structured approach to adopting automation. This section has the following logical order in terms of the covered practices and examples:

- ▶ The “Discovery Loop”
 - ▶ Motivation Mapping
 - ▶ Big Picture Event Storming
 - ▶ Domain Storytelling
 - ▶ Metrics-based Process Mapping (MBPM)
 - ▶ User Story Mapping
 - ▶ Target Outcomes
- ▶ The “Options Pivot” phase
 - ▶ Kano Model
 - ▶ Value Slicing
 - ▶ MVP

The bigger context in terms of the practices we touched in this guide is explained in the following short video:



The Open Practice Library is available here:⁶ <https://openpracticelibrary.com>
You will find many references to its sections in this guide, but not only to them.

The next section introduces an approach of “structured brainstorming” where members of the different teams with different expertise share their knowledge with each other – **Event Storming**. This is needed to understand the big picture of the context for automation.

We also introduce the “**sample project**” we use throughout this guide.

3.1. Let’s discover what we have first

When we want a holistic discussion involving business people as well as technical experts, we need to make their communication as seamless as possible. It turns out that **event-centric focus** helps to transcend the business/tech boundary. Events of all sorts relevant to the domain in question can be recognised by all the parties. As usual with brainstorming, we want to prioritize moving forward over precision.

3.1.1. Popular event-centric practices

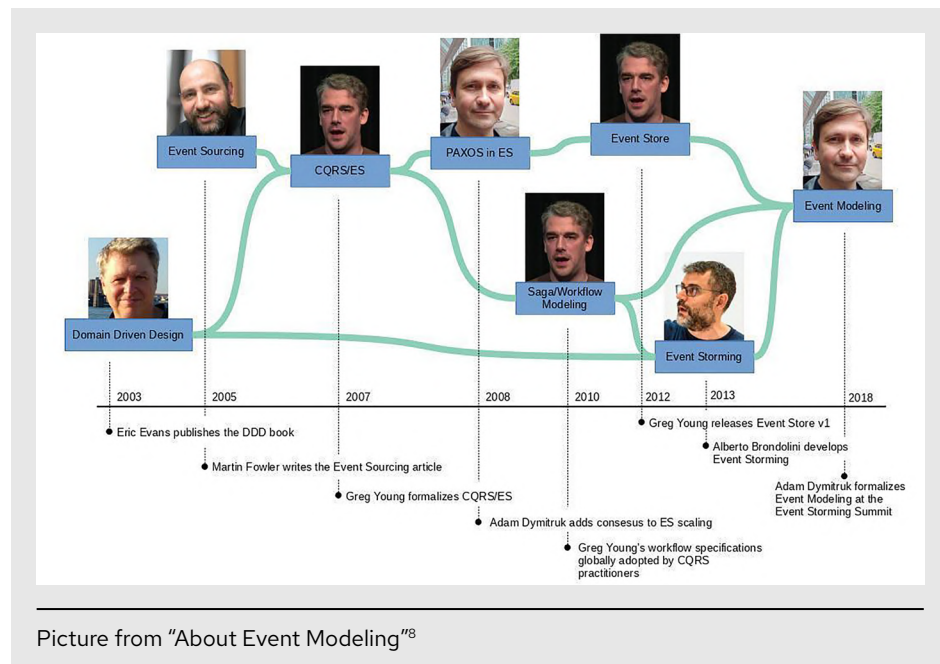
Described in the next section “**Event Storming**” practice focuses on discovering the problem space. Another practice that incorporates an event-centric view (as opposed to a system or time based view) is “Event Modeling”.⁷ The latter creates a blueprint for a solution based on concepts of “Command Query Responsibility Segregation” and “Event Sourcing” (CQRS and ES design patterns). In other words, Event Modeling can be very efficient in building solutions of some types, but it is not for everyone.

Here is a summary of event-centric practices evolution, with “Event Storming” and “Event Modeling” on the right:

⁵“Outcome-driven delivery with Open Practice Library” <https://www.youtube.com/watch?v=N4mBIZq8MnQ>

⁶“Open Practice Library” <https://openpracticelibrary.com/>

⁷“Event Modeling” <https://openpracticelibrary.com/practice/event-modeling/>



In this guide, we are focusing on the "Event Storming" practice. If you are interested in more information about "Event Modeling," then this "deep dive" with its author Adam Dymitruk should be helpful "Event Modeling Deep Dive".⁹ You might also like the following Event Modeling resources:

- ▶ A good sample-based explanation¹⁰
- ▶ Some Miro-friendly resources can be found here¹¹
- ▶ How "Event Modeling" process can be bootstrapped from "Big Picture Event Storming"¹²

3.1.1.1. What is "Event Storming" discovery practice?

There are different types of Event Storming workshops. The "Big Picture Event Storming" is usually used to discover the business domain and share knowledge. The "Software Design Event Storming" is more focused on system design.

The "Event Storming" practice is a synthesis of facilitated group learning practices from Gamestorming¹³ and the principles of domain-driven design (DDD).

The knowledge is expressed on colorful sticky notes – either virtual during online sessions with tools like Miro,¹⁴ or on-site in a room with an 8-10 meters-long wall, the bigger the better. Below is a picture from where you can get the color and purpose of each.



⁸ "About Event Modeling" <https://eventmodeling.org/about/>

⁹ "Event Modeling deep dive with Vaughn Vernon and Adam Dymitruk" <https://www.youtube.com/watch?v=ufKqwisD1l8>

¹⁰ "What is Event Modeling? (with example)" <https://www.qoeeven.com/blog/event-modeling/>

¹¹ "Event Modeling cheat sheet" <https://eventmodeling.org/posts/event-modeling-cheatsheet/>

¹² "Event Storming vs Event Modeling by Rafal Maciag @DDD Istanbul" <https://youtu.be/LDPlvmD9upk?t=2274>

¹³ Gamestorming on Wikipedia <https://en.wikipedia.org/wiki/Gamestorming>

¹⁴ Miro <https://miro.com/>

Here we introduce the core concepts only, you are welcome to follow the links below in this section to read about it all in more detail.

Event

Any Event Storming starts with collecting the knowledge about your business, expressed in a form of domain events, which are stuck to the wall (electronic or physical) in time order from left to right. A domain event is any event that is important for your business and makes an explicit impact on it. The event could happen inside or outside your business system. By convention, each event is expressed in a past term verb, written down on an orange sticky note.

Examples: "Booking is requested", "Booking is accepted", "Ride is started", "Ride is finished."

Command

After you feel that most of the domain knowledge is reflected in events on the wall, focus on the commands that trigger appropriate events. Write down commands, important for the business, on a light blue note and place left to the event they spawn. A command is a message that represents the intention of a user, and could be expressed as an action.

Examples: "Request booking", "Cancel booking", "Request refund."

Policy

Policy artifact is used to document conditions and policies for events to happen. So, on a storming wall a policy stays between a domain event and a command. Policies are formalized like "Whenever...X, then...Y" or "If...X, then...Y."

Example: "Whenever booking is created, then create an offer."

See also:

- ▶ Event Storming in Open Practice Library¹⁵
- ▶ The Big Picture in Open Practice Library¹⁶
- ▶ Big Picture Event Storming¹⁷

3.1.1.2. Example of concept-to-technology "zoom-in" with events

As we mentioned previously, event-centric focus helps to transcend the business/tech boundary during design workshops and following implementation iterations. In this section, we show how events in the user/business context – the domain events – can help people "zoom into" particular technology realizations, while using the same "events language."

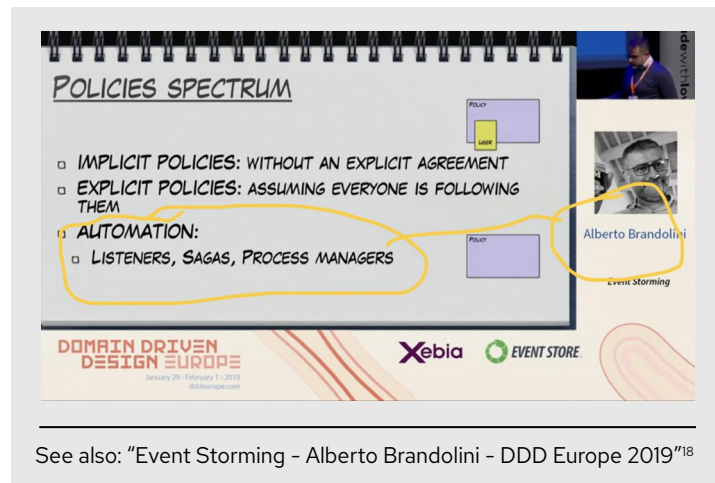
The original author of popular Event Storming practice Alberto Brandolini mentions the following types of "Policies" (in "Event Storming" context) for automation:

- ▶ Listeners
- ▶ Sagas
- ▶ Process managers

¹⁵ Event Storming in Open Practice Library <https://openpracticelibrary.com/practice/event-storming/>

¹⁶ The Big Picture in Open Practice Library <https://openpracticelibrary.com/practice/the-big-picture/>

¹⁷ Big Picture Event Storming <https://medium.com/@chatuev/big-picture-event-storming-7a1fe18ffabb>



POLICIES SPECTRUM

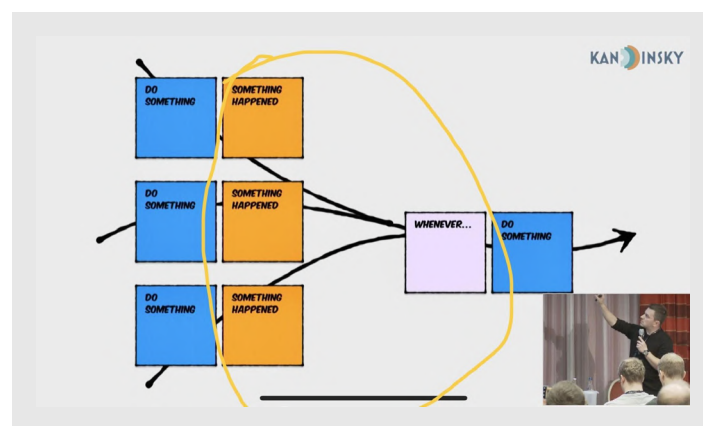
- ◻ IMPLICIT POLICIES: WITHOUT AN EXPLICIT AGREEMENT
- ◻ EXPLICIT POLICIES: ASSUMING EVERYONE IS FOLLOWING THEM
- ◻ AUTOMATION:
 - ◻ LISTENERS, SAGAS, PROCESS MANAGERS

Alberto Brandolini
Event Storming

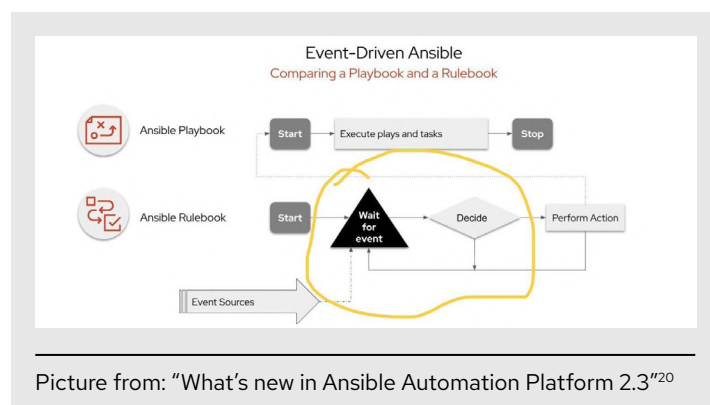
DOMAIN DRIVEN DESIGN EUROPE
January 29 - February 1, 2019
ddd@europa.com

Xebia EVENT STORE

See also: "Event Storming - Alberto Brandolini - DDD Europe 2019"¹⁸



Another expert on this subject Mariusz Gil also elaborates the process manager pattern in his talk "Modeling complex processes and time with saga/process manager patterns".¹⁹ In particular, Mariusz shows a demo implementation of the "process manager" automation pattern. In layman's terms, it is about how three flows of domain events get merged into one such flow. Speaking about the technical side of things, in the context of this guide, the EDA (Ansible Rulebooks) fits the "Process Manager" pattern very well. Here is a diagram showing how it can work:



¹⁸ Event Storming - Alberto Brandolini - DDD Europe 2019 <https://www.youtube.com/watch?v=mLXQIYEwK24>

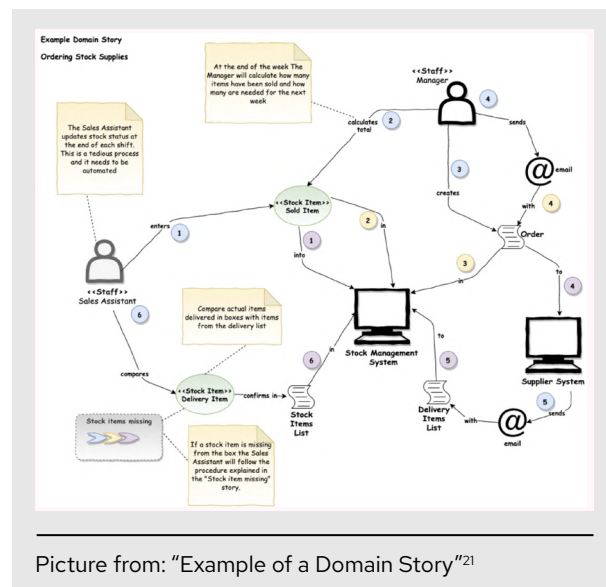
¹⁹ Modeling complex processes and time with Saga/Process Manager patterns - Mariusz Gil <https://www.youtube.com/watch?v=WvjTCmeGIGA>

²⁰ What's new in Ansible Automation Platform 2.3 <https://www.ansible.com/blog/whats-new-in-red-hat-ansible-automation-platform-2.3>

3.1.2. A traditional lens for domain discovery: “Domain Storytelling” practice

One way to picturize a project’s domain may be to use “Domain Storytelling.” For example, you might come across parts of the process where several people or systems are actively involved. If these parts are critical to your analysis of a domain, you might want to go the extra mile to get another perspective on the process – model those parts additionally as Domain Stories.

A picture is worth a thousand words. Here is an example of Domain Story “Ordering Stock Supplies.” As you can see, graphical notation of this particular example resembles UML component/deployment diagrams. Hence, it can be “friendly” for say engineers in the project team.



Picture from: “Example of a Domain Story”²¹

An important aspect of “Domain Storytelling” is that it bridges the current state and the desired state. Hence, applying this practice is helpful to further detail the project goals. Here are the suggested steps:

1. Have a subject matter expert start explaining the business process (current state)
2. The facilitator or supporting team record each step in workflow diagrams
3. For each recorded step, number it and add detailed explanations below adjacent to the corresponding number
4. Repeat the process for desired state

You can read more about “Domain Storytelling” here.²²

3.1.3. Sample project for “Sample Org”: Motivation Mapping

As we know, business challenges are often dynamic, that is they can be quite seasonal and subject to rapid and repeated change. One way to stay current with stakeholder requirements is to deeply understand and “map” their motivations applying the “Motivation Mapping” practice.²³ In this section we are using this practice to give structure to the sample project of this guide. In real life, the filled “Motivation Mapping” visual canvas radiates “synchronized motivation” for the project team and all the stakeholders. It helps everyone to keep an eye on whether their actual activities fit the planned strategic direction.

²¹ Example of a Domain Story <https://medium.com/domain-driven-stories/example-of-a-domain-story-d20ade05831e>

²² Domain Storytelling in Open Practice Library <https://openpracticelibrary.com/practice/domain-storytelling/>

²³ Motivation Mapping in Open Practice Library <https://openpracticelibrary.com/practice/motivation-mapping/>

In summary, “Motivation Mapping” uses a simple canvas to organize the following:

- ▶ **Context** – What is the environment surrounding us or our customer? What platform are we operating on? What are the factors impacting our environment?
- ▶ **Goals** – What goals do we or our customers have? What do we want to achieve?
- ▶ **Reasons for the goals** – Identify why we set this goal or objective.
- ▶ **Obstacles** – What is preventing us from making progress on our objective? What are the impediments?
- ▶ **Target outcome** – What state do we need to be in to be able to say that our goal was achieved?

In the context of automation, we recommend checking if the following benefits are relevant for your “Motivation Mapping” canvas:

- ▶ Automation introduces greater consistency in execution
- ▶ Better task completion time
- ▶ Increased team efficiency and capacity for more important tasks
- ▶ Increased confidence in vulnerability management process
- ▶ Streamlined deployment across environments
- ▶ More scalable and extensible processes
- ▶ More “Infrastructure-as-Code” with Ansible & co.
- ▶ Improved functional tests
- ▶ Reduction in manual work/overhead and less errors
- ▶ Some automation content is reusable in other business workflows

The “sample project” of this guide is happening for an imaginary organization “Sample Org,” which has recently become subject to extended regulations and compliance requirements. It is happening because the company is expanding its business into the financial sector as well as into new geographical regions. For example, the PCI-DSS set of standards²⁴ require the installation of latest patches while setting the maximum time allowed for critical patch installations. The “App” in this guide is now crucial for the “Sample Org” operations. It is the primary software used to request access privileges across the “Sample Org,” so its stability and availability are of highest importance.

Let us also comment in terms of the “sample project” team at this initial point of the project. The following preparational activities would be timely:

- ▶ Onboarding activities, testing the needed remote collaboration tools, setting up meetings for the week, preparing the walls for whiteboard sessions (if onsite activities are planned)
- ▶ Introductions and checking everyone is on the same page
- ▶ Discuss what a high performing team is. Its benefits and phases it needs to go through (the Tuckman’s “forming–storming–norming–performing” model)
- ▶ Discuss and agree on ways of working and how we will work as a team (also, potentially creation and agreement on the “social contract”)

Here is how the “Motivation Mapping” canvas looks for the “sample project” of this guide. It was produced by the “sample project” team and the informed stakeholders working for “Sample Org” – during an one-hour session:

²⁴ PCI DSS on Wikipedia https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard

Context

Category "App" (an important for business app)

A Linux® box in the data center running an important business app, which is used to request access privileges across the organization. Number of its users increased recently, unstable behavior reported. The app should stay on Linux box(es) because of compliance reasons. The app deployment is currently managed manually.

Category "Patching the fleet" (patching and security remediation for the Linux fleet)

The company is expanding into the new markets, now they need more capacity. The remaining manual steps are becoming bottlenecks every day. Amount of work and completion timeframe requirements for the entire Linux fleet within the organization are expected to rapidly grow in the nearest months.

Goals

App

Business users should have stable, 24/7 access to the app.

Patching the fleet

Patching should be fully automated. This automation will vastly increase the speed and quality of the updates delivery to internal customers, while allowing the specialists to focus on more satisfying work.

Reasons for the goals

App

With expansion into the new markets across time zones, the business operates nearly 24/7. In particular, more users need the app during night and over weekends (in the original timezone). It is getting harder to find a "maintenance window" for the app upgrades/maintenance. Also, newly hired executives feel unsatisfied with the slow responsiveness of the app.

Patching the fleet

Full automation is required because of increasing difficulty with meeting regulatory compliance controls for the Linux boxes. This process is taking staff away from more important activities. Human error probability is high because of a lot of manual work. The infrastructure team wants to shift their ways of working toward the "Site Reliability Engineer" model.

Obstacles

Both categories "App" and "Patching the fleet":

Onboarding for engineers who should implement the high availability (HA) for the app, improvements to the patching process will take two-four weeks because of security reasons. Also, the compute capacity for the development (non-production) environment in the regional data centers, which are available for the project team, is scarce right now. Extra capacity can be provisioned in three-six weeks.

Target outcome

App

- ▶ Updates to new app versions in the production environment as well as security patching and maintenance of corresponding Linux system(s) should happen without observed disruption of the service for the business users.
- ▶ The app should be responsive even during peak usage hours. Up to three-second response time is satisfactory in edge cases. However, more than 95% of user requests should complete in under one second.

Patching the fleet

- ▶ For mission-critical Linux workloads, automated patching and security remediation should happen without observed downtime of the service. It should be achieved in six weeks for the app and in two-three months for several other mission-critical Linux workloads.
- ▶ For non-critical Linux workloads, automated patching and security remediation for up to 3000 servers should complete within 24 hours time frame (with allowed planned downtime). That level of “patching throughput” should be achieved in three months time.

3.1.4. Sample: The initial discovery

It is important for Event Storming workshops to have the right people present. This includes people who know what questions to ask and the informed stakeholders (domain experts, product owners). It works smoothly together with some help from a good moderator and facilitator – Scrum Master, Agile Coach.

During the preparation for Event Storming workshops, the “sample project” team decided to focus on the following knowledge areas:

- ▶ App in use – requesting access privileges
- ▶ Deployment of app new version
- ▶ Patching the Linux box of app

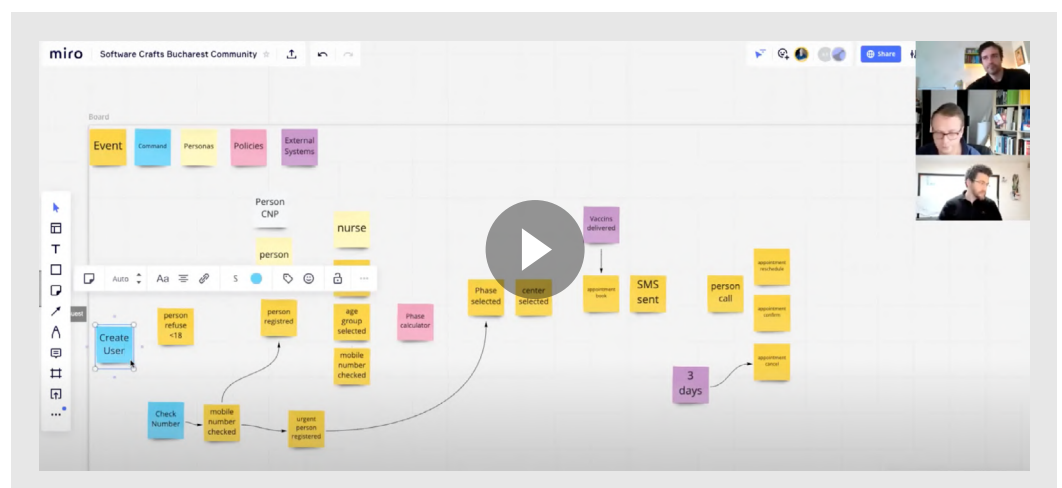
The project team went through the following steps for each of the above-mentioned knowledge areas:

- ▶ Step 1 - Collect domain events (Big Picture)
- ▶ Step 2 - Refine domain events (Big Picture)
- ▶ Step 3 - Track causes (Process Modeling)

In the context of this guide, the most important knowledge area is “Deployment of app new version.” The next section has the “sample project” team’s Event Storming results for it, for all the three above-mentioned steps. The results for the other two areas can be found in the following appendices correspondingly:

- ▶ Appendix A. Event Storming for “App in use – requesting access privileges”
- ▶ Appendix B. Event Storming for “Patching the Linux box of app”

If you are interested in a complete example of Event Storming session from scratch, the following video can be helpful: “Event Storming Workshop @Bucharest Software Craftmanship Community”²⁵

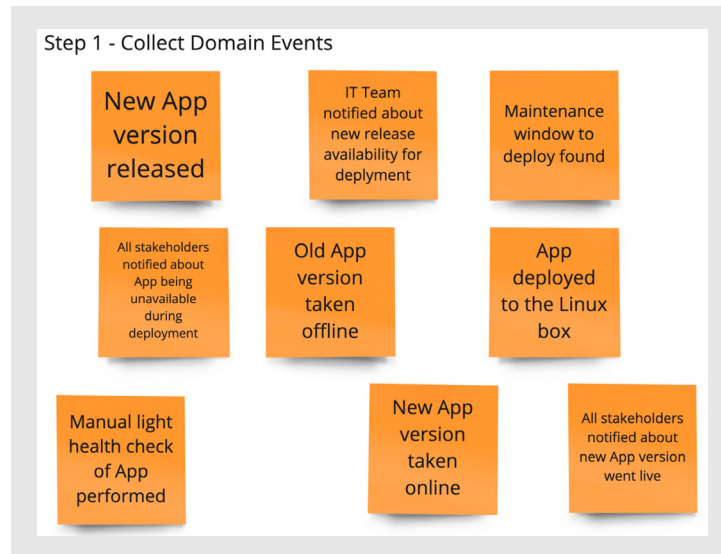


²⁵ Event Storming Workshop @Bucharest Software Craftmanship Community <https://www.youtube.com/watch?v=xV5aDdj3PVE>

3.1.4.1. Sample: Event Storming for “Deployment of app new version”

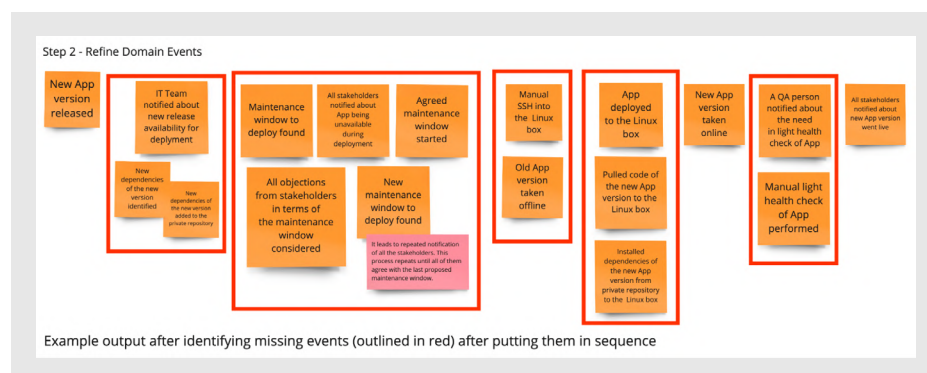
During the “Step 1 - Collect domain events (Big Picture),” each participant uses only orange post-its. Each orange post-it stands for a professional event. A professional event is a technically relevant fact that happened in the course of business. The verb on the post-it must therefore be in the past.

The first round is a pure brainstorming process about the existing domain events. Ask people to hang the events in the chronological order in which they occur.



During the “Step 2 - Refine domain events (Big Picture),” go through the domain event post-its with the participants. Ask the participants to explain what each event means. Check for syntactical correctness.

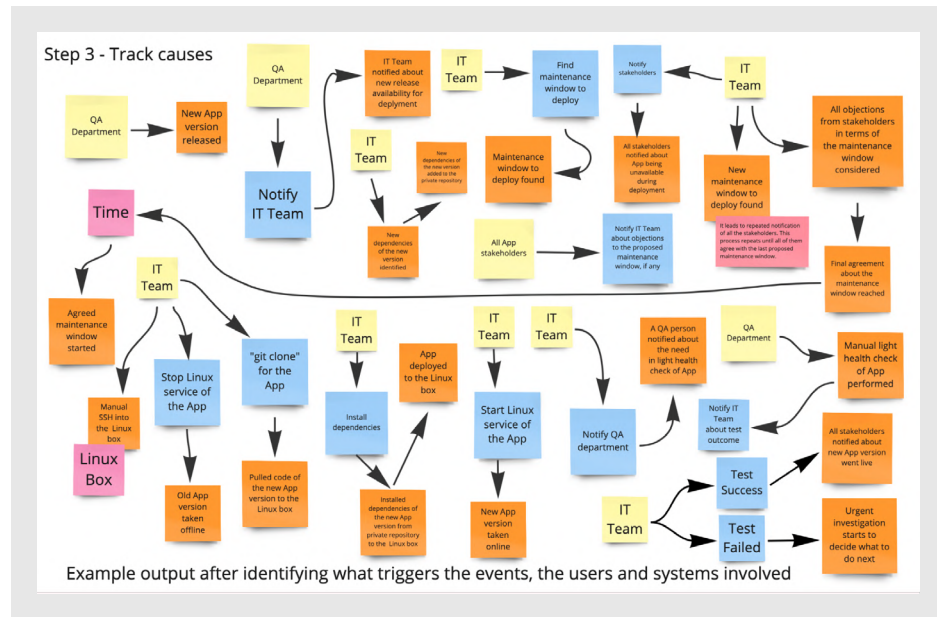
Also discuss again whether the events are in the right order in terms of time. Unify occurring synonyms (different terms for the same thing) and sharpen differences if the same term was used to describe different things.



During the “Step 3 - Track causes (Process Modeling),” get into the cause analysis. Where do the domain events come from? There are four main causes:

- ▶ User actions (Commands)
- ▶ External systems
- ▶ Time (for example, appointment elapsed); business processes
- ▶ Other domain events (through automatic reactions; via policies/business rules)

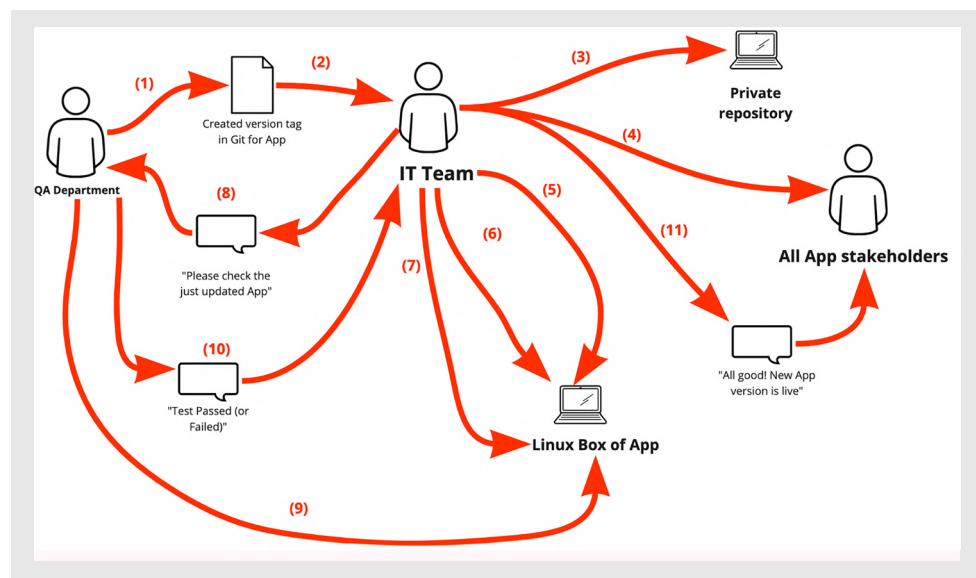
Ask the participants about the triggers of the domain events.



3.1.4.2. Sample: Domain Story for “Deployment of app new version” (current state)

The “sample project” team decided they also wanted to go the extra mile to get another perspective on the process of “Deployment of app new version” – they modeled the cooperative part of the process additionally as a Domain Story. The team went through the following steps to produce the diagram and its description below:

- ▶ Have a subject matter expert start explaining the business process current state
- ▶ The facilitator or supporting team record each step in workflow diagrams
- ▶ For each recorded step, number it and add detailed explanations below adjacent to the corresponding number



1. The QA department tags a new release of app in the Git version control system
2. The QA department notifies IT team about new release availability for deployment
3. IT team identifies new dependencies and adds the artifacts to the private repository
4. IT team agrees with all app stakeholders about the maintenance window to deploy the new version of app
5. When the agreed maintenance window starts, the IT team manually gets into the Linux box and stops the Linux service of the app (old version)
6. IT team deploys the new version of app to the Linux box: (a) pulls code of the new app version to the Linux box; (b) installs dependencies of the new app version from private repository to the Linux box
7. IT team starts the Linux service of the app, taking it online
8. IT team notifies the QA department about the need in light health check of just updated app
9. The QA department performs a manual light health check of the just deployed app
10. The QA department notifies the IT team about their testing outcome
11. If the new app version light testing had been reported “PASSED”, then IT team notifies all the stakeholders about new app version went live

3.2. What are our issues? What are the bottlenecks in our processes?

We now want to refine our understanding of the issues, problems, inefficiencies and risks we have today. What improvements would be most impactful today and strategically? We want to align our “project team” on the process improvement efforts. With clear understanding of our process bottlenecks, “project team” can focus on process improvements which are most relevant today and strategically. Let us see how the MBPM practice can help with it in the next sections and summarize all the relevant information we collected so far after it.

3.2.1. What is “Metrics-based Process Mapping” practice?

Let us introduce MBPM. MBPM is a lean process improvement technique. An MBPM exercise helps identify possible process improvements and their impact. The MBPM practice creates a detailed map of a process that delivers an outcome/service to customers/stakeholders. The map displays process steps, responsible actors, lead time metrics, and quality metrics. It helps identify the described below potential improvements to the process.

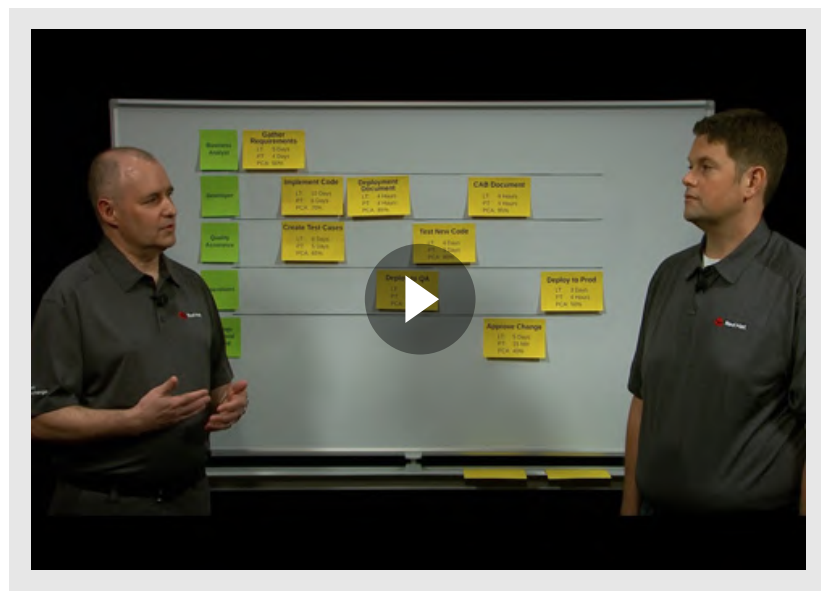
So, while “doing MBPM,” for each **step** in a **process**, project participants identify or estimate the following:

- ▶ A **verb-noun phrase** to describe the process step action, **for example, “test code”** (extra details are OK for clarity if needed)
- ▶ The **Team** or **Actor** that executes the step
- ▶ The **Process Time (PT)**, or the time that the team or actor actively, actually executes the action in a step, that is doing corresponding productive work
- ▶ The **Lead Time (LT)**, or the time elapsed:
 - ▶ **from** when some chunk of work is ready to get started by a Team/Actor at this particular step (in other words, there are no reasonable dependencies; for example, the “Definition is Ready” is met for a Backlog Item)
 - ▶ **to** when the outputs/outcomes of this particular step get delivered to the next “work steps” (as an input correspondingly)
- ▶ The **“Percent Complete and Accurate” (PCA or “%C&A”)**, or the percentage of a particular step’s outputs that are fully complete and accurate. An output is complete and accurate if it does not require any further corrections, additions, or clarifications for downstream “work centers” (Teams/Actors) to start their work on downstream chunks of work.

For example, such “mapping” of work activities in a particular process helps identify:

- ▶ Percent of deliverables with no defects, which are the output of the process. The LT for these deliverables.
- ▶ Percentage of deliverables which require fixes or rework at some intermediate process step(s), thus contributing to slowness of the process.
- ▶ Steps which have large LT while having short PT, which is indicative of a bottleneck. Such steps are candidates for high-impact improvements or even “refactoring” via some alternatives with better (higher) PT/LT ratio and lower PT in general.

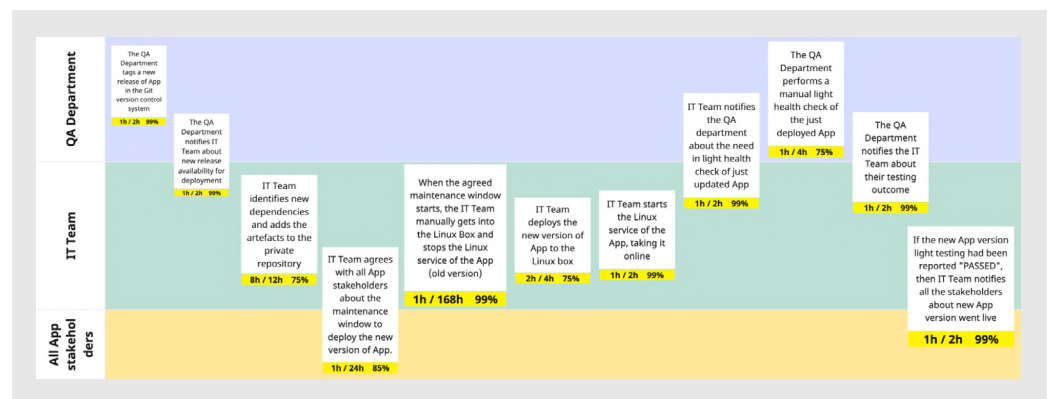
If you are interested in an extra example of MBPM outside of this guide scope, the following video can be helpful: “Visualizing, measuring and optimizing your processes with Metric Based Process Mapping.”²⁶



3.2.2. Sample: MBPM – today’s process flows, owners, metrics

The following diagram gives an idea how MBPM can look like for the “Deployment of app new version.” Each of the process steps has the following figures in the bottom:

Process Time (PT)/Lead Time (LT) “Percent Complete and Accurate” (PCA %)



²⁶ Visualizing, measuring and optimizing your processes with Metric Based Process Mapping <https://www.youtube.com/watch?v=g1XSbEwR3bU>

Let's summarize the PT-LT-PCA data collected/estimated in a table for simpler analysis. It will show how the MBPM practice can suggest improvements to processes in a quantitative way.

Step name	PT (hours)	LT (hours)	PT/LT %	PCA %
1. Tag a new release	1	2	50%	99%
2. Notify IT team about new release availability	1	2	50%	99%
3. Identify new dependencies and update the private repository	8	12	67%	75%
4. Agree with everyone about the maintenance window to use for deployment	1	24	4%	85%
5. Wait for the the agreed maintenance window start and stop the Linux service of the app	1	168	0.5%	99%
6. Deploy the new version of app	2	4	50%	75%
7. Start the Linux service of the app	1	2	50%	99%
8. Notify the QA department about the need in light health check	1	2	50%	99%
9. Perform a manual light health check of the just deployed app	1	4	25%	75%
10. Notify the IT team about testing outcome	1	2	50%	99%
11. Notify all the stakeholders about new app version went live	1	2	50%	99%
Summary	Summary PT: 19 hours	Summary LT: 224 hours	Summary PT/LT: 8.5% "Summary PT" divided by "Summary LT"	Summary PCA 33% All the PCA% values multiplied

The preceding sample process map and table display the following significant results:

- ▶ 33% of deliverables “flow” through the process and emerge complete and accurate with no defects. The lead time for these deliverables is 224 hours.
- ▶ Most deliverables (67%) require fixes or rework at some intermediate process step, which contributes to the lead times longer than “expected” 224 hours.
- ▶ The **fourth** and **fifth** steps have both large Lead Times and a low PT/LT ratio, which is indicative of bottlenecks. Improvements to those steps will maximize improvement to the overall process Lead Time.
- ▶ While the **ninth** step has relatively moderate Lead Time, the PT/LT ratio is still too low. Also, its PCA of 75% suggests that it has a relatively high risk of increased LT due to inaccuracy/incompleteness.

3.2.3. Sample: Issues and risks today

So far, the “sample project” team has performed:

- ▶ “Motivation Mapping”: overall goals, context, reason for the goals, obstacles, and target outcomes and analyzed the following specific knowledge areas further in the context defined via “Motivation Mapping”
- ▶ App in use – requesting access privileges
 - ▶ Event Storming
- ▶ Deployment of app new version
 - ▶ Event Storming
 - ▶ Domain Story (current state)
 - ▶ MBPM
- ▶ Patching the Linux box of app
 - ▶ Event Storming

After it, the “sample project” team had a couple of sessions summarizing the issues, problems, inefficiencies, and risks in the context. Here are the results collected during those analytical sessions:

(a) Issues analysis: “The important for business app”

- ▶ There are no logical issues found in how the app works internally. The app on the single Linux box is unresponsive at times because of increased user load. It is because of the lack of processing power of the single Linux box (likely CPU overutilized)
- ▶ Deployment of a new version is not easy
- ▶ There is no proper visibility into the health of the app, user-facing errors get detected post factum
- ▶ Maintenance windows: take lots of time to negotiate, lots of time to wait for its start
- ▶ Manual tests require cross-department coordination; lead time is considerably longer than process time
- ▶ Risk: manual steps and manual tests are prone to human mistakes

(b) Issues analysis: “Patching and security remediation for the Linux fleet”

Patching is just partially automated. Drawbacks:

- ▶ Requires manual intervention too often; the amount of such manual participation is often proportional to the number of Linux boxes
- ▶ Too slow to complete
- ▶ Downtime can be avoided at times
- ▶ Maintenance windows: take lots of time to negotiate, lots of time to wait for its start
- ▶ Risk: manual steps and manual tests are prone to human mistakes

In summary, the issues and risks identified above are related to the low level of process automation or insufficient compute capacity allocated to perform the processes.

So what is the strategy to improve it all? In the next section, we introduce some helpful relevant practices and see what “sample project” team decides accordingly.

3.3. Are we preparing to deliver according to the real needs?

In section 3.1.3. “Sample: Motivation Mapping...,” we introduced the sample project’s overall target outcomes. When we are done with discovery practices for the current interaction, we want to tie the priorities back to a real business/customer needs, because these needs will change over time and can even make redundant the just prioritized item. Basically, on every iteration, we are checking that we are not about to spend effort for nothing. “Output” does not equal “Outcome!” That is, “all results of actual effort spent” are not the same as “only the valuable for the stakeholders results/achievements.”

In other words, when the “Target Outcomes” session is held during a non-initial iteration of “Discovery/Development,” its primary focus are the results/outcomes of the just completed iteration. We want to cross-check and verify we are on the most efficient track here.

The described below “Target Outcomes” practice helps us perform such cross-checking in a structured way.

3.3.1. What is “Target Outcomes” discovery practice?

The “Target Outcomes” practice is the main practice in the outcomes element of the discovery iteration. It summarizes the findings and learnings from other discovery practices into a finite set of statements that can be publicly displayed for team and stakeholders to regularly reference. See also: Target Outcomes.²⁷

Let us comment in terms of the project team at this point. Any of the following scenarios might indicate that a team will benefit from establishing target outcomes:

- ▶ The scope for a team deliverable changes unexpectedly
- ▶ Your teams celebrate a successful delivery before customers interact with the product or provide product feedback
- ▶ Daily team interactions focus on completing the feature instead of completing a version of the feature that achieves desired results

3.3.2. Sample: Strategy for the first delivery iteration

We are preparing for the very first “delivery iteration” of the “sample project.” It is not uncommon to focus on building an MVP at this stage. What is MVP?

3.3.2.1. What is MVP?

In general, an MVP aims to achieve the following goals:

- ▶ Deliver something that the users/stakeholders can touch/feel/interact with
- ▶ Demonstrate the idea of the product in action, make the stakeholders more interested in the project, and attract more potential users
- ▶ As early as possible, validate that the product idea is worth investing more effort/time. Otherwise, make it clear that spending more resources on the product is not reasonable
- ▶ Collect valuable feedback from real users/stakeholders

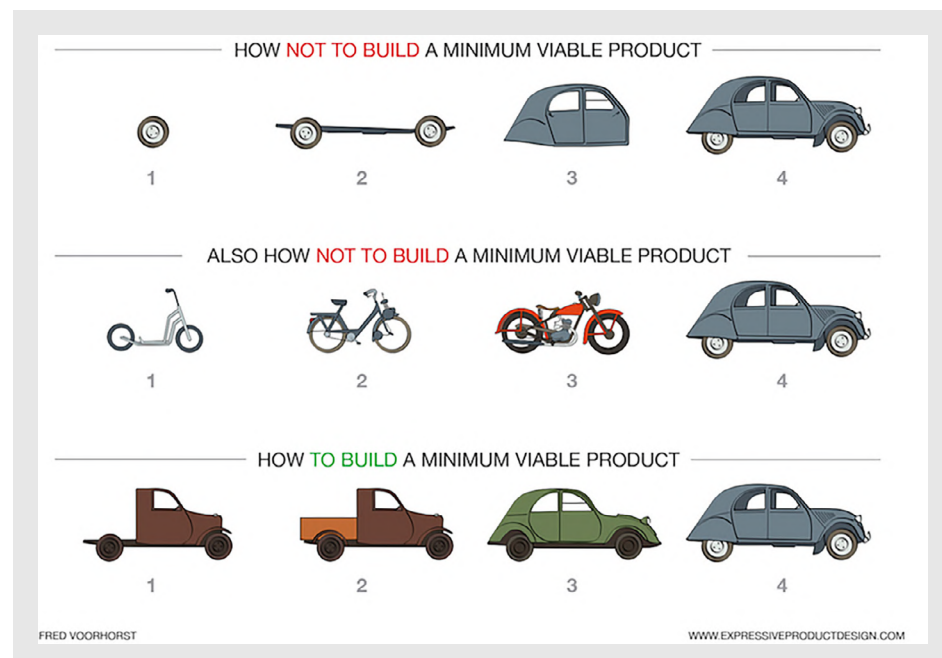
Simply put, it is sort of “shortest path to complete the entire journey,” where the “entire journey” is the overall future outcome of the project. That is, no “bells and whistles” are in focus for MVP.

²⁷ Target Outcomes in Open Practice Library <https://openpracticelibrary.com/practice/target-outcomes/>

With an MVP development, you need to both:

- A. Efficiently move through the overall project plan execution closer to the ultimate, overall ideal project outcome
- B. Right on the MVP release already, soon, have an outcome which is highly valuable/usable for the ultimate user of the future overall project outcome

In the picture below, the third row depicts an example when both (A) and (B) points from the above are met. Speaking about the other two examples in the picture, the second one does not fit (A), while the first row does not fit point (B) above.



In the next section, we outline the MVP for the sample project.

3.3.2.2. Sample: MVP ideas and choice

In general, the sample project's team agreed that it makes sense to start delivering value in the form of an MVP matching the goals of the sample project. Once the MVP proves it is all valuable to the stakeholders at a reasonable cost, the sample project's team will be able to add features based on team vote/priority later.

In other words, whatever features the team/stakeholders come up with now, they can be classified into two big categories:

- ▶ in-MVP-scope
- ▶ post-MVP

It is important to mention that all the features/items preliminary planned for some "post-MVP" release, they may or may not end up being implemented. The feedback from users/stakeholders of the MVP will be collected shortly after its release, the requested improvements will likely be considered along with the pre-existing "candidates" for inclusion into the next release.

In section 3.1.3. “Sample: Motivation Mapping...,” we introduced the sample project’s overall goals along with other contextual details. Here are they per corresponding domain:

(a) The important for business app

Business users should have stable, 24/7 access to the app

(b) Patching and security remediation for the Linux fleet

Patching should be fully automated

Having the goals in mind, the project team decided to have an “Let’s outline MVP” session. The team members proposed some ideas placing them on a whiteboard/a Miro board. It took like 10 minutes of their collective effort. The ideas were further grouped and a bit refined by the facilitator of the session. That grouping activity went along with some semi-informal discussions around the points, which targeted overall clarity for all the participants. That took 25 more minutes.

Here are the resulting refined ideas for the MVP to deliver for the sample project in the first delivery iteration:

- A. Automated patching with downtime (for non-critical Linux boxes)
- B. Replatforming of the app to Kubernetes/Red Hat OpenShift
- C. Improvements to the app deployment, patching, observability, and performance. (We will automate patching of the app only. We will try to expand this idea to more Linux boxes with HA requirements later.)

During the discussion over the above-mentioned items, the team highlighted the following:

- ▶ **Idea (A)** is covering the “Patching should be fully automated” in detail, but it does not give reasonable attention to the app, for example, its availability expectations in the sample project.
- ▶ **Idea (B)** has a potential of being a breakthrough in operational readiness for the app, leveraging recent developments in containerised applications orchestration, observability, and availability. However, the “Automated patching of Linux fleet” aspect is missing here, while the own Linux fleet is a natural and important part of IT assets of the organization benefiting from the sample project’s outcomes.
- ▶ **Idea (C)** covers both the relevant app improvements as well as patching of Linux boxes. A disadvantage of this option, at this point, is that it is not immediately clear what exactly will need to be done. The original author of that idea was able to persuade the team it is not that bad via giving some insights into the technical design though.

The facilitator decided to run a voting session to agree on the resulting MVP idea, that is on the idea the team decides to deliver. It took seven minutes. Here are the voting results:

- ▶ **1st place – Idea (C):** Improvements to the app deployment, patching, observability, and performance
- ▶ **2nd place – Idea (B):** Replatforming of the app to Kubernetes/Red Hat OpenShift
- ▶ **3rd place – Idea (A):** Automated patching with downtime (for non-critical Linux boxes)

As mentioned earlier, a disadvantage of the winning Idea (C), at this point, is that it is not immediately clear what exactly will need to be done to achieve the MVP results. In the next section, we introduce several practices which help us get the needed clarity.

3.4. Is everything ready to start the first delivery iteration?

As we discussed previously, the sample project’s team agreed that it makes sense to start delivering value in the form of the MVP matching the goals of the sample project. The MVP is:

Improvements to the app deployment, patching, observability and performance

(We will automate patching of the app only. We will try to expand this idea to more Linux boxes with HA requirements later.)

Such improvements must progress along with the overall project goals:

- ▶ Business users should have stable, 24/7 access to the app
- ▶ Patching should be fully automated

It is now time for the sample project's team and the informed stakeholders to come up with particular features to deliver in the simple project. They should be then classified into two big categories:

- ▶ in-MVP-scope
- ▶ post-MVP

The in-MVP-scope features will effectively form the Scope-of-Work (SoW) for the MVP.

By now, the team understands the "Sample Org" domains well. They have the big-picture Event Storming results and the "Domain Story" diagrams. They have identified "bottlenecks" in the domain processes via applying the MBPM practice. Plus, they summarized current issues as described in section "3.2.3. Sample: Issues and risks today."

It all informs in terms of specific functionality as well as non-functional requirements which the team/product owner wants to get delivered. In the next section, we will quickly revise some relevant terminology and concepts, and then focus on identifying features for the first delivery iteration of the "sample project."

3.4.1. What are "User Story Mapping" and "Value Slicing" practices?

Let us first double-check we are on the same page when it comes to the relevant terminology.

3.4.1.1. Terminology

In general, "Epics" can have nested "Features," while "Features" can have nested "User Stories." They can be all used to represent functional as well as non-functional requirements for a product/system. "User Stories" are just a technique for "requirements by conversation." Non-functional requirements here are just a category of conversation, along with functional requirements. They are all an aspect of an item we are aiming to deliver.

Epic:

- ▶ Large initiatives delivering new products, solutions, services to stakeholders/customers
- ▶ Comprised of a large collection of features

Feature:

- ▶ Capabilities that the product owner is interested in
- ▶ Provides values to users/stakeholders
- ▶ Realized by some number of user stories

User Story:

- ▶ Represents an individual need of users/stakeholders
- ▶ Describes a chunk of functionality/non-functional aspect that will be of value to a user/stakeholders
- ▶ Serves as an atomic item for project planning
- ▶ Represents the smallest increment of value in project
- ▶ Convenient to serve as the focus of a targeted conversation by the project team

3.4.1.2. "User Story Mapping" and "Value Slicing"

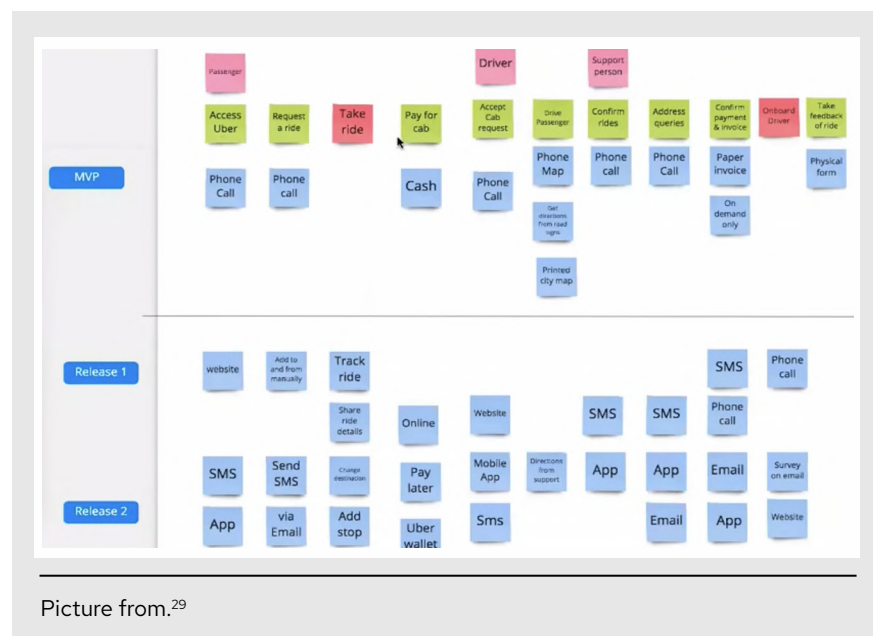
"User Story Mapping" is an evolution of the traditional agile backlog practice. It is an effective practice for creating lightweight release plans that can drive standard agile delivery practices. When correctly applied, it gives you the following:

- ▶ A backlog of scope items (captured as stories or simply feature titles) the project team believes can be delivered in the planning window
- ▶ The backlog "sliced" into about three "delivery iterations," such that it forms the outline of nearest plan for delivery
- ▶ Enough detail for the first "delivery iteration" of the plan to get started with the work

See also: User Story Mapping & Value Slicing.²⁸

Here is an example of User Story Mapping with an initial (MVP) delivery iteration identified via "Value Slicing." **Note:** The domain in the picture below is different from the "sample project" of the guide, this particular example is for an imaginary modern taxi service domain. We will apply the practices to the "sample project" in the next sections.

The mapping below was identified by a team, along with some "Value Slices" preliminary allocated to the iterations of "Release 1" and "Release 2" correspondingly.



The green cards in the picture above represent features of the product (product capabilities). The blue cards are the proposed User Stories (smallest increments of value in project).

In general, there are two very different ways to "slice" your agile backlog:

- ▶ **Horizontal slicing** where the focus is on working on architectural layers one by one
- ▶ **Vertical slicing** where the work is sliced by end-to-end features cutting across the whole architectural stack

If your top priority is the **real value delivery**, then, in common case, **we recommend using vertical slicing rather than horizontal slicing** while forming "chunks of work."

²⁸ Target Outcomes in Open Practice Library <https://openpracticelibrary.com/practice/target-outcomes/>

²⁹ Story Map, MVP, Prioritization, Estimation – Part 6/7 | Agile Product Journey (Idea to Inception) <https://www.youtube.com/watch?v=DI8HFmVLXQc>

Horizontal slicing should be normally considered as a fallback option. For example, when only team members who can work on only a single layer of a system are available over the next delivery iteration. For example, only on the business-logic layer of an imaginary system, so that the DB layer as well as the presentation layer of the system cannot be improved over the next iteration physically – there are no specialists to work on them. It means that no “vertical slices” requiring changes to the other two layers can be worked on over the next iteration. In this case, a horizontal slice pertaining to the business-logic layer can be included into the iteration to keep the project work moving forward.

See also: Horizontal or vertical slicing for agile.³⁰

3.4.1.3. Are your User Stories ready to start implementation?

How can we get confident a User Story is ready for implementation? Just having something like “Cash” (a prototype of a User Story) under feature “Pay for cab” does not seem enough. It is where the following checklist “Definition of Ready” (“ready” for a User Story description to be used to start its implementation) can be handy for the project team to detail User Stories:

- ▶ User Story is understood by the team
- ▶ User Story has clear business value
- ▶ User Story is estimated
- ▶ User Story’s dependencies identified
- ▶ User Story is small
- ▶ User Story’s acceptance criteria is defined

Another famous “Three ‘C’-s” criteria set for “good User Story” is as follows:

Card

- Written on note cards
- Can be annotated with estimates, values, notes, etc.

Conversation

- Details of the story come out through conversations with the customer

Confirmation

- Acceptance test is defined to confirm the implementation of story is complete

Last but not least, here are a couple of reasonable templates to use for the primary text of “good User Story”:

- ▶ **As a** {role}, **I can** {do or have something with measurable qualities} **to** {achieve a business goal}
- ▶ **To** {achieve a business goal}, {roles} **can** {do or have something with measurable qualities}

3.4.2. Sample: Epics, features, and initial User Story Map

As described in the earlier sections, the “sample project” team decided to build the following MVP:

Improvements to the app deployment, patching, observability, and performance

The original author of that MVP idea had given the team some insights into the technical design they had in mind. The following two Epics have been identified for the initial scope of the sample project:

- ▶ **Epic: Rolling update of app**
For example, functionality of rolling update of the Linux box(es) with a new app version (with integrated into the process Linux patching)
- ▶ **Epic: Remote health-checker**
The health-checker app to observe the app health (a user-like, remote component)

³⁰ Horizontal or vertical slicing for agile? <https://www.youtube.com/watch?v=jQg27pFGmWA>

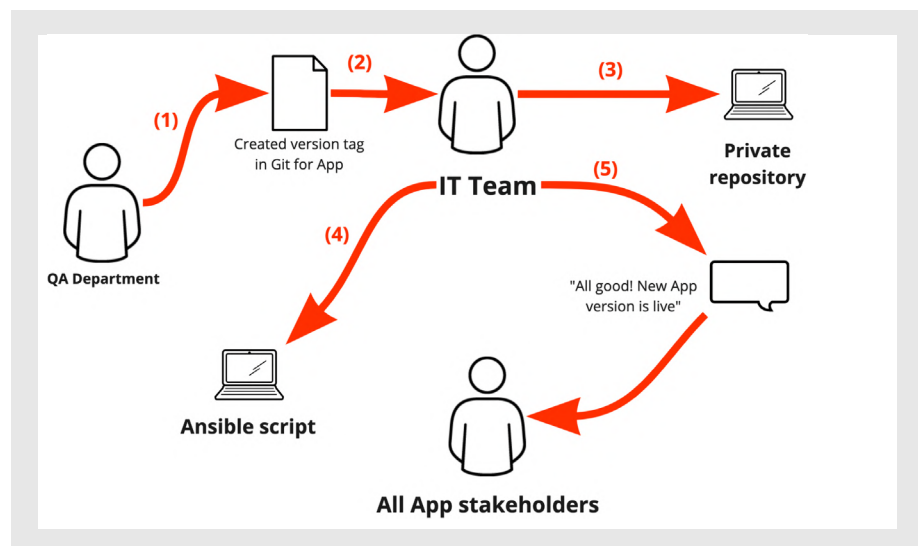
At this point, the “sample project” team decided that it makes sense to have a Domain Storytelling session “Round II” – about the DESIRED STATE. The idea of the session is to further clarify what features should/can to be done as part of the MVP or later.

3.4.2.1. Sample: Domain Story for “Deployment of app new version” (desired state)

As described in section “3.2.2. Sample: MBPM – today’s process flows, owners, metrics,” the “sample project” team identified the following “bottlenecks” in the deployment process:

- ▶ ...
- ▶ Agree with everyone about the maintenance window to use for deployment
- ▶ Wait for the the agreed maintenance window start and stop the Linux service of the app
- ▶ ...
- ▶ Perform a manual light health check of the just deployed app
- ▶ ...

The “sample project” team has invited the subject matter expert – the original author of the MVP idea. First of all, they want to focus on the improvements related to the currently identified “bottlenecks.” They have had a session and came up with the following diagram describing the DESIRED STATE of the process.



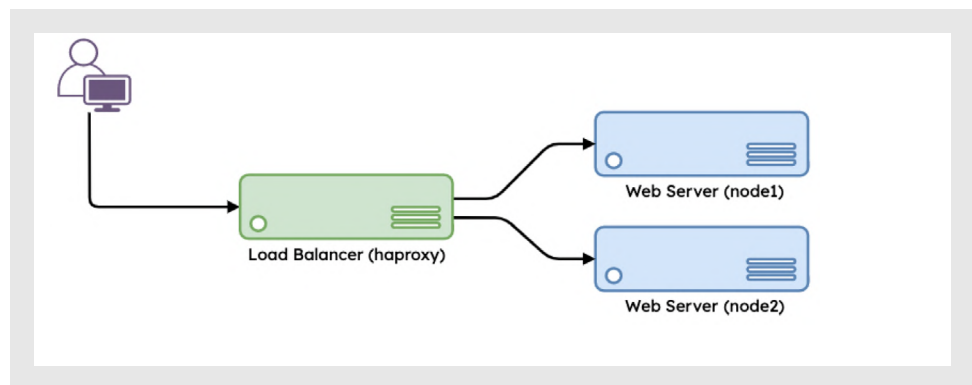
1. The QA department tags a new release of app in the Git version control system
2. The QA department notifies IT team about new release availability for deployment
3. IT team identifies new dependencies and adds the artifacts to the private repository
4. IT team executes the automation Ansible script to deploy the new app version
5. If the new app version automated testing had been reported “PASSED” by the script, then IT Team notifies all the stakeholders about new app version went live

As you can see, the new diagram above is free from the previously identified “bottlenecks.” It is thanks to the Ansible script and some architectural changes described by the idea’s author as follows:

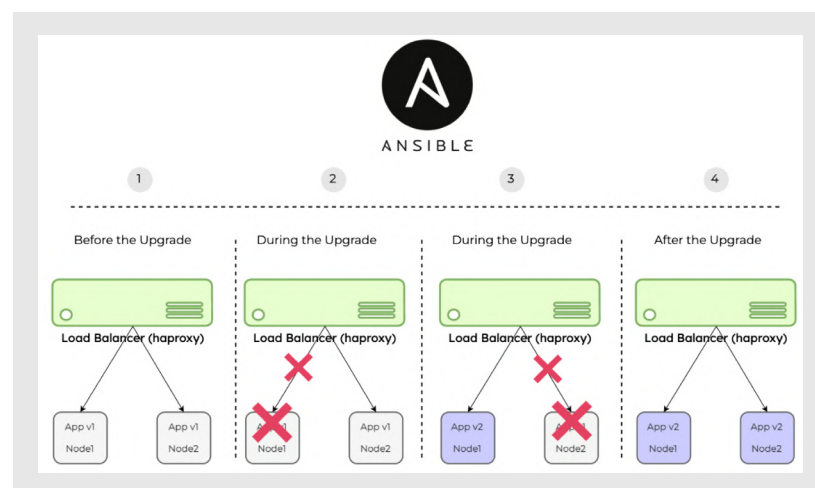
- ▶ Several instances of app running on corresponding dedicated Linux boxes are behind a load balancer (for example, HAProxy).³¹
 - ▶ All the instances are serving the user requests, so the service has better availability for the user. For example, if one of the instances goes down, the others still serve the requests;
 - ▶ User requests processing is faster because more CPUs are utilized for it
- ▶ Rolling deployment of the new versions of app instances is happening behind the load balancer. In particular, the instances get taken offline one-by-one (and then returned to “active” state), thus allowing the other instances to serve the user requests during the overall rolling update process.

Here are a couple more diagrams visualizing what was explained by the expert to the “sample project” team (pictures from Ansible for Real-Life Automation):³²

App hosted on multiple Linux servers with a load balancer in front



Rolling update of app using Ansible

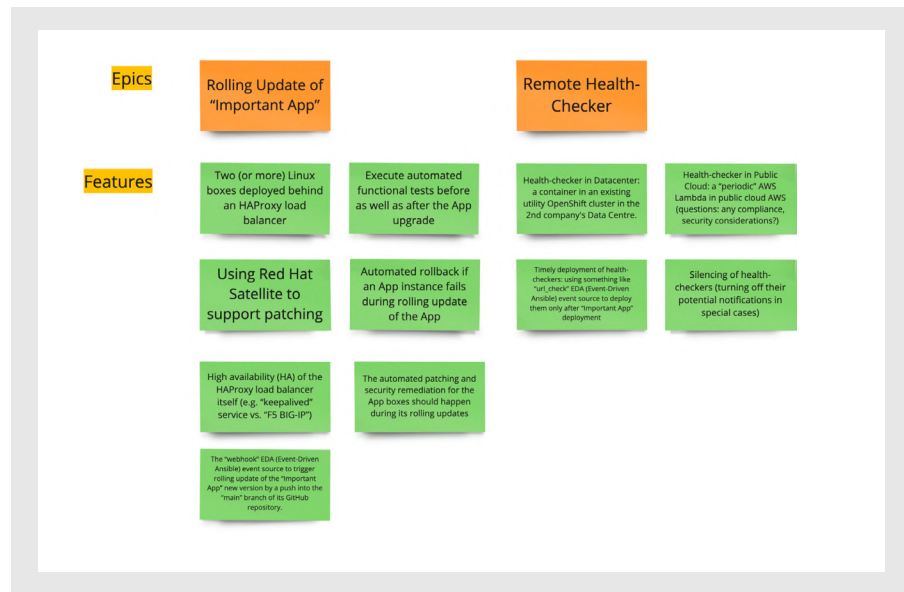


3.4.2.2. Sample: Initial User Story Map

“Sample project” team took a day to further investigate the improvements discussed in the previous section. After it, they had another session where they started building the User Story Map as follows:

³¹ HAProxy on Wikipedia <https://en.wikipedia.org/wiki/HAProxy>

³²“Ansible for Real-Life Automation” by Gineesh Madapparambath <https://amzn.asia/d/flHSDjh>



Epic: Rolling update of app

Features (capabilities that the stakeholders are interested in):

- ▶ Two (or more) Linux boxes deployed behind an HAProxy load balancer
- ▶ Execute automated functional tests before as well as after the app upgrade
- ▶ The automated patching and security remediation for the app boxes should happen during its rolling updates
- ▶ Using Satellite to support patching
- ▶ Automated rollback if an app instance fails during rolling update of the app
- ▶ The "webhook" EDA (Event-Driven Ansible) event source to trigger rolling update of the app new version by a push into the "main" branch of its GitHub repository. See also: Event-Driven Ansible + Gitops.³³
- ▶ HA of the HAProxy load balancer itself (for example, "keepalived" service vs. "F5 BIG-IP")

Epic: Remote health-checker

Features (capabilities that the stakeholders are interested in):

User-like (remote) health-checker component(s) which can help detect user-facing issues before real users get affected. It is similar to New Relic's "synthetic" clients or "canaries" possible with AWS CloudWatch.

- ▶ Health-checker in data center: a container in an existing utility Red Hat OpenShift cluster in the second company's data center
- ▶ Health-checker in public cloud: a "periodic" AWS Lambda in public cloud AWS (questions: any compliance, security considerations?)
- ▶ Timely deployment of health-checkers: using something like "url_check" EDA event source to deploy them only after app deployment
- ▶ Silencing of health-checkers (turning off their potential notifications in special cases)

³³ Event-Driven Ansible + Gitops <https://www.youtube.com/watch?v=Bb5IDftLbPE>

3.4.3. What features go into the first delivery iteration?

Automation is almost always helpful, but its implementation comes at cost, so we need to prioritize planning items. Such prioritization is needed for the following:

- ▶ To **maximize the value** of effort spent
- ▶ To **maximize “work-not-done”** (that is, do not spend effort when there is no point in it)
- ▶ To **reduce the amount of “waste” outputs** which are not real value. “Output” does not equal “Outcome!” Some outputs are actually “waste” because they require some maintenance effort over time, while they are not giving real value to the stakeholders/customers.

In the next sections we cover four practices to support prioritization, with more attention given to the “Kano Model” practice. We then see that the “sample project” team decides to use the latter as the most customer-focused one.

3.4.3.1. Practices of prioritization

The “**Impact and Effort Prioritization (Matrix)**” practice **evaluates tasks on impact and required effort to implement**. Teams can choose to deliver quick wins over improvements with a bigger impact that require more effort if this suits the team’s needs. See also: [Impact & effort prioritization \(Matrix\)](#).³⁴

The “**Priority Sliders**” practice is a tool that facilitates conversations about **relative priorities** to focus upcoming activities. See also: [Priority Sliders](#).³⁵

The “**How-Now-Wow**” matrix-based practice is graphically similar to the above one. It is especially valuable for **categorizing innovative ideas**. See also: [How-Now-Wow Prioritization \(Matrix\)](#).³⁶

A more textured, “**Kano Model**” practice, it is the **most customer-focused** among the four practices mentioned here. Let us elaborate more on the Kano Model right below.

Noriaki Kano, who is a professor emeritus of the Tokyo University of Science, invented this simple ranking scheme which distinguishes between essential and differentiating for customers/stakeholders attributes of products. According with Prof. Kano, the consumer’s preferences can be classified in the following five categories:

- ▶ **Must Have/Must-be (aka musts, basic needs, but dissatisfiers when absent)** - When done well, customers are neutral, when not present, or done poorly customers are very disappointed
- ▶ **One-dimensional quality (aka wants, satisfiers, wow factors)** - Leading to satisfaction when present and dissatisfaction when not present
- ▶ **Attractive (aka delighters)** - Provide satisfaction when realized, but do not cause dissatisfaction if not present
- ▶ **Indifferent (aka neutrals)** - Often these are overlooked features taking effort to achieve, but for nothing – they are of no importance to customer
- ▶ **Reverse** - When realized, they are causing dissatisfaction actually with some customers, for example, unneeded overcomplications.

³⁴[Impact & effort prioritization \(Matrix\) in Open Practice Library](https://openpracticelibrary.com/practice/impact-effort-prioritization-matrix/) <https://openpracticelibrary.com/practice/impact-effort-prioritization-matrix/>

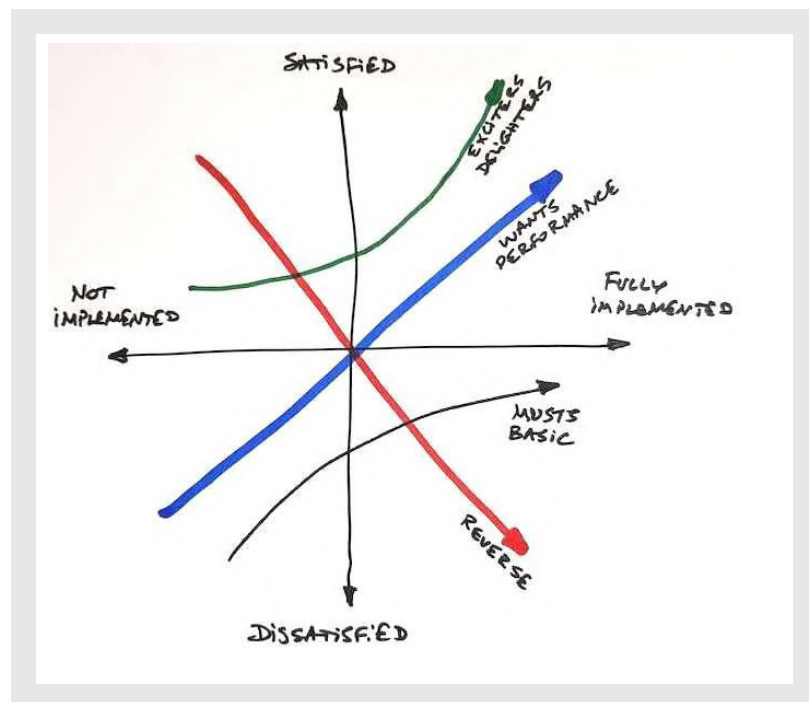
³⁵ [Priority Sliders in Open Practice Library](https://openpracticelibrary.com/practice/priority-sliders/) <https://openpracticelibrary.com/practice/priority-sliders/>

³⁶ [How-Now-Wow Prioritization \(Matrix\) in Open Practice Library](https://openpracticelibrary.com/practice/how-now-wow-prioritization-matrix/) <https://openpracticelibrary.com/practice/how-now-wow-prioritization-matrix/>

Hence, here is the general “Kano-based” recommendation when planning a product:

- ▶ **Include all must-have** needs. These are mandatory.
- ▶ **Include a satisfactory level of “One-dimensional Quality” (“Wants”) and “Attractive” (“Delighters”)** features and optimize them just enough. It’s okay if customers/stakeholders are asking for more – you don’t have to optimize these features right away if customers still find value in the product. You can optimize/add accordingly during the next few iterations.

The following “Kano Matrix” can be used to plot the product features being planned and discuss the correct categories for the features with the project team, customers, stakeholders. Then just follow the above-mentioned “Kano-based” recommendation to include the right features into the next “delivery iteration.” See also: Kano Model.³⁷



3.4.3.2. Sample: Scope of MVP defined, Value Slices

“Sample project” team decided to apply the “Kano Model” practice to prioritize/classify the initial features identified so far. They liked it is truly customer-centric. What goes into the MVP and what can be implemented later?

3.4.3.2.1. Sample: Kano Model applied

In particular, the team decided to employ Kano’s standardized questionnaire to measure participants’ opinions in an implicit way. It should help to understand the sample project’s stakeholders in a deterministic, “finite” way. The participants therefore need to answer two questions for each product feature, from which one question is formulated in a “positive” way and the other is in a “negative” way.

Here is the questionnaire given to each of the participants for each of the features outlined in section “3.4.2.2. Sample: Initial User Story Map”:

³⁷ Kano Model in Open Practice Library <https://openpracticelibrary.com/practice/kano-model/>

	I like it	I expect it	I am neutral	I can tolerate it	I dislike it
"Positive" questions					
How would you feel if the product had ...?					
How would you feel if there was more of ...?					
"Negative" questions					
How would you feel if the product <i>did not</i> have ...?					
How would you feel if there was less of ...?					

Such questionnaires were then collected by the facilitator, and the "vote" of that participant for placing the feature into a "Kano Category" was calculated according with the following table proposed by professor Kano:

"Positive" questions		"Negative" questions		Category
I expect it	+	I dislike it	→	Must Have
I like it	+	I dislike it	→	One-dimensional quality (wants, satisfiers)
I like it	+	I am neutral	→	Attractive
I am neutral	+	I am neutral	→	Indifferent
I dislike it	+	I expect it	→	Reverse (causing dissatisfaction)

For example, for the following feature:

- ▶ The "webhook" EDA event source to trigger rolling update of the app new version by a push into the "main" branch of its GitHub repository.

One of the participants filled the questionnaire as follows:

	I like it	I expect it	I am neutral	I can tolerate it	I dislike it
"Positive" questions					
How would you feel if the product had ...?	✓				
"Negative" questions					
How would you feel if the product <i>did not</i> have ...?					✓

So that participant's vote got calculated as follows:

"I like it" (to a "positive" question) + **"I dislike it"** (to a "negative" question) => **One-dimensional quality (aka wants, satisfiers)** – Leading to satisfaction when present and dissatisfaction when not present.

In summary, the features have been classified as follows by the relevant participants of the questionnaire:

Epic: Rolling update of app

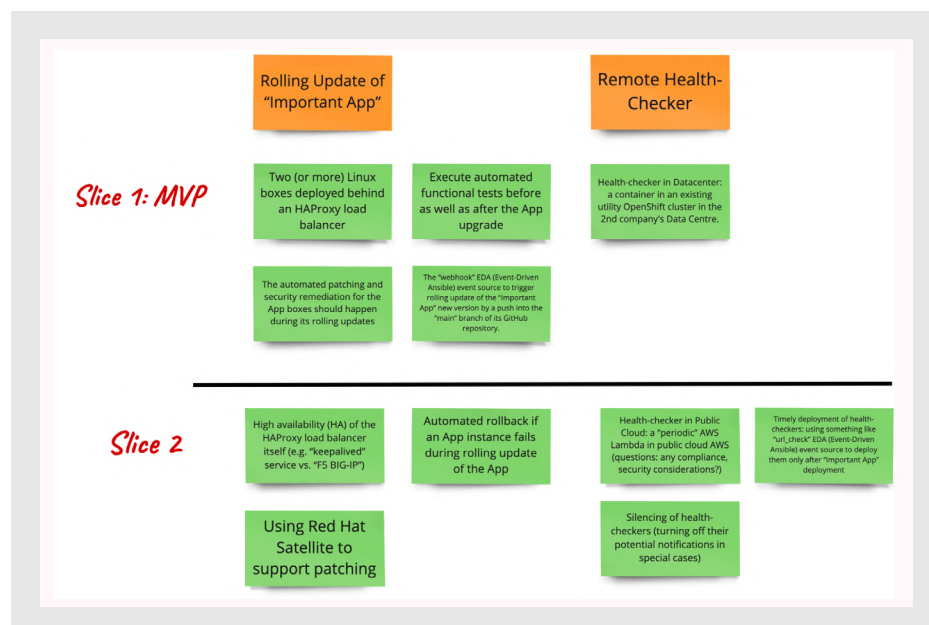
- ▶ **[Must have]** Two (or more) Linux boxes deployed behind an HAProxy load balancer
- ▶ **[Must have]** Execute automated functional tests before as well as after the app upgrade
- ▶ **[Must have]** The automated patching and security remediation for the app boxes should happen during its rolling updates
- ▶ **[One-dimensional quality]** Using Satellite to support patching
- ▶ **[One-dimensional quality]** Automated rollback if an app instance fails during rolling update of the app
- ▶ **[One-dimensional quality]** The “webhook” EDA event source to trigger rolling update of the app new version by a push into the “main” branch of its GitHub repository
- ▶ **[Attractive]** HA of the HAProxy load balancer itself (for example, “keepalived” service vs. “F5 BIG-IP”)

Epic: Remote health-checker

- ▶ **[Must have]** Health-checker in data center: A container in an existing utility Red Hat OpenShift cluster in the second company’s data center
- ▶ **[Indifferent]** Health-checker in public cloud: A “periodic” AWS Lambda in public cloud AWS (questions: any compliance, security considerations?)
- ▶ **[Attractive]** Timely deployment of health-checkers: using something like “url_check” EDA event source to deploy them only after app deployment
- ▶ **[Attractive]** Silencing of health-checkers (turning off their potential notifications in special cases)

3.4.3.2.2. Sample: Value Slices

Here is how the “sample project” team and relevant stakeholders have sliced out the high value to form the incremental release strategy. They want to maintain focus on delivering valuable outcomes.



4. Deliver automation efficiently

We believe the following are the key aspects of a valuable engineering solution, which should be balanced at all times:

- ▶ Value of the solution to customers from business perspective
- ▶ Health of the solution in production environment
- ▶ Requirements for the solution
- ▶ Design of the solution
- ▶ Development and delivery of the solution
- ▶ Automation level
- ▶ Security and compliance
- ▶ Cost of the solution

Such disciplines as Site Reliability Engineering (SRE) define a set of principles and practices which help manage IT solutions through automation, which is more scalable and sustainable than manual repetitive operations. SRE is especially useful to apply for maintaining the desired level of balance between the following aspects of IT solutions:

- ▶ Health of the solution in production environment
- ▶ Development and delivery of the solution

For example, SRE recommends how to manage risk of system reliability. Such a risk can emerge because of planned changes to the IT solution itself or due to some changes in the environment where it runs. SRE introduces a special “error budget” metric which removes the politics from negotiations between the operations and developers when deciding how much risk to allow. See also: Embracing risk.³⁸

Speaking of risks, in situations when the original project plan may change all of a sudden or the project is happening in a highly innovative or dynamic environment, the risk of imbalance of the following is quite high:

- ▶ Value of the solution to customers from business perspective
- ▶ Requirements for the solution
- ▶ Design of the solution
- ▶ Development and delivery of the solution

In such cases, unfortunately, the following approach to delivery may not work well in the long run: “Let’s just build everything according to the original plan.”

For the project team, it just makes sense to be more adaptable and agile for the success of such projects. In our experience, about 80% of projects are like that. Hence, in the next section we introduce a structured approach to “being agile” after a quick refresher in terms of the context of this guide.

4.1. Introducing the “Delivery Loop” and agile practices

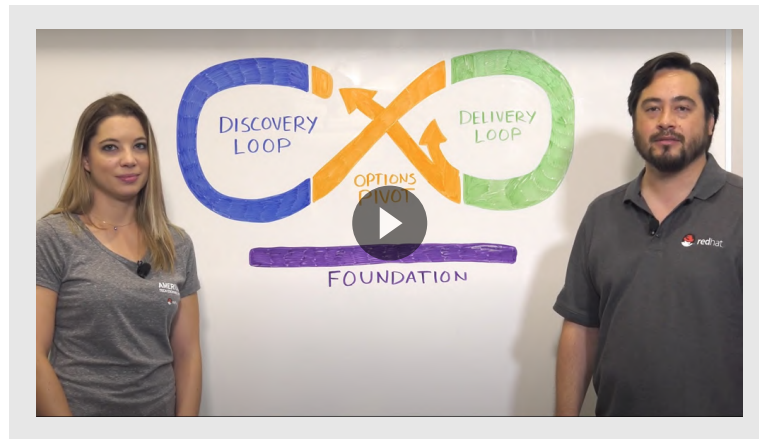
In the previous large section 3, we covered the following practices:

- ▶ The “Discovery Loop”
 - ▶ Motivation Mapping
 - ▶ Big Picture Event Storming
 - ▶ Domain Storytelling
 - ▶ MBPM
 - ▶ User Story Mapping
 - ▶ Target Outcomes

³⁸ Embracing risk <https://sre.google/sre-book/embracing-risk/>

- ▶ The “Options Pivot” phase
 - ▶ Kano Model
 - ▶ Value Slicing
 - ▶ MVP

In this section, we focus on some practices related to the “Delivery Loop” explained in the second part of the same video mentioned earlier, the “Outcome-driven delivery with Open Practice Library” video:³⁹



As mentioned in the introduction to this large section 4, we often want the project team to “be agile” for the project success. Over time, agile practitioners have identified several common ceremonies/practices to effectively facilitate agile development and delivery of IT solutions. Here is how those practices map to the “Delivery Loop”:

- ▶ Backlog refinement (options practice)
- ▶ “Increment Planning” or “Sprint Planning” (delivery practice)
- ▶ Daily standup (foundational practice)
- ▶ Showcase or demo (delivery practice)
- ▶ Retrospectives (foundational practice)
- ▶ “Definition of Done” (foundational practice)

Please find more details about the practices in the following sections.

4.1.1. Backlog refinement (options practice)

The “backlog refinement” is a time-boxed activity during which the project team collaborates with the product owner/informed stakeholders to review items in the current increment (iteration) and any top priority items in the backlog. During backlog refinement, the team clarifies the details and acceptance criteria for each item to deliver. See also: Backlog refinement.⁴⁰

4.1.2. “Increment Planning” or “Sprint Planning” (delivery practice)

Increment planning is a time-boxed activity during which the project team collaborates with the product owner/informed stakeholders, committing to a set of defined work and setting a goal for the upcoming increment (iteration). See also: Iteration (Sprint) Planning.⁴¹

³⁹ Outcome-driven delivery with Open Practice Library <https://www.youtube.com/watch?v=N4mBIZq8MnQ>

⁴⁰ Backlog refinement in Open Practice Library <https://openpracticelibrary.com/practice/backlog-refinement/>

⁴¹ Iteration (Sprint) Planning in Open Practice Library <https://openpracticelibrary.com/practice/iteration-planning/>

4.1.3. Daily standup (foundational practice)

The daily standup is a short activity that provides time for the project team to synchronize their efforts towards the agreed increment (iteration) goals. Team members meet to share the status of ongoing work and highlight impediments to their progress. See also: Daily standup.⁴²

4.1.4. Showcase or demo (delivery practice)

The showcase is an activity where the project stakeholders and interested parties are given a demonstration of recent work performed during the increment (iteration) and feedback is collected. A showcase (demo) can also be performed at key milestones in the life cycle of a product/project or scheduled on demand. See also: Showcase.⁴³

4.1.5. Retrospectives (foundational practice)

At the end of an increment (iteration), the project team uses the Retrospective activity to reflect on, analyze, and adapt their ways of working together. Retrospectives help reveal facts, observations, and feelings that have an effect on project team performance. It is all used to propose, collect, and agree on ideas for the relevant project team improvements. See also: Retrospectives.⁴⁴

4.1.6. "Definition of Done" (foundational practice)

The "Definition of Done" practice aligns understanding and shared expectations across the project team and the informed stakeholders. For example, the "Definition of Done" can be agreed as follows:

- ▶ User story is tested against acceptance criteria
- ▶ User story delivered without observed errors
- ▶ User story demoed to the stakeholders
- ▶ User story approved by the product owner
- ▶ Documentation related to the user story has been updated

See also: "Definition of Done"⁴⁵

4.2. Sample: Let's deliver according to the plan of our first iteration

In order to further refine and plan activities in continuation of the outcomes of the "Options Pivot" phase, the "sample project" team and the product owner decided to apply the following two practices, one after another:

- ▶ Backlog refinement (options practice)
- ▶ "Increment Planning" or "Sprint Planning" (delivery practice)

For example, they noticed, there are the following obstacles for the "sample project" (as mentioned in section 3.1.3. "Sample project for Sample Org: Motivation Mapping..."):

Onboarding for engineers who should implement the HA for the app, improvements to the patching process will take two-four weeks because of security reasons. Also, the compute capacity for the development (non-production) environment in the regional data centers, which are available for the project team, is scarce right now. Extra capacity can be provisioned in three-six weeks.

⁴² Daily standup in Open Practice Library <https://openpracticelibrary.com/practice/daily-standup/>

⁴³ Showcase in Open Practice Library <https://openpracticelibrary.com/practice/showcase/>

⁴⁴ Retrospectives in Open Practice Library <https://openpracticelibrary.com/practice/retrospectives/>

⁴⁵ Definition of Done in Open Practice Library <https://openpracticelibrary.com/practice/definition-of-done/>

Hence, the “sample project” team and the product owner decided to agree on the following extra action item:

Find somewhere to provision environments that will support deployment of the MVP without impacting current systems in the data centers. The engineering team should be able to start immediately.

To summarize, they have refined several decisions and agreed on the action plan for the first delivery iteration. Please find them in the next two sections.

4.2.1. Sample: Decisions for iteration

The MVP agreed as follows:

- ▶ Two Linux boxes for the app behind an HAProxy load balancer
 - ▶ Such organization of compute resources gives the increased compute capacity
 - ▶ It should support automated rolling updates of new app versions
 - ▶ It should support automated patching and security remediation of the two Linux boxes (during rolling updates)
 - ▶ In the scope of MVP, we will support GitOps-style initiating of rolling deployment/update of the new version of the app behind the HAProxy. In particular:
 - ▶ We will implement the standard “ansible.eda.webhook” event source listening for “push” events from GitHub for the app repository via EDA. Once such an event is received, EDA should initiate the automated rolling update to the new app versions.
 - ▶ No rollback support will be implemented
- ▶ Execute automated functional tests before as well as after the app upgrade
- ▶ The user-like health-checker for the app:
 - ▶ It is a container in an existing utility Red Hat OpenShift cluster in the second company’s data center.
The health-checker application will be implemented in JavaScript/Typescript.
 - ▶ It should be timely deployed, but, for now, without using the “url_check” event source provided by EDA.

4.2.2. Sample: Action plan for iteration

- ▶ Investigate HAProxy load balancer configuration
- ▶ Define the functional tests for the app. Implement them in Ansible.
- ▶ Design and implement the Ansible Playbook which performs:
 - ▶ Automated rolling updates of new app versions
 - ▶ Automated patching and security remediation of the two Linux boxes (during rolling updates)
 - ▶ Execute automated functional tests before as well as after the app upgrade
- ▶ Configuration of EDA for GitOps-style rolling deployment of the app
 - ▶ Implementation and deployment of the rulebook.
The standard “ansible.eda.webhook” event source will be listening for “push” events from GitHub for the app repository via EDA. Once such an event is received, EDA should initiate the automated rolling update of the app using the above-mentioned Ansible Playbook.
- ▶ Find somewhere to provision environments that will support deployment of the MVP without impacting current systems in the data centers. The engineering team should be able to start immediately.

Note: For the needs of this guide, we will be deploying the Linux boxes for the MVP in AWS public cloud. It can be actually also a “real life decision” by the “sample project” team potentially, provided it satisfies all the real life requirements and regulations of course.

- ▶ AWS configuration to deploy the app instances and the HAProxy load balancer instance
- ▶ Implement Terraform code to deploy the data center-like three EC2 Linux instances

- ▶ For the health-checker application:
 - ▶ Design and implement the health-checker application (in JavaScript/Typescript)
 - ▶ Investigate API for integration with Red Hat OpenShift

Note: the “sample project” team is planning to deploy into:
“an existing utility Red Hat OpenShift cluster in the second company’s data center”
However, the project is for an imaginary “Sample Org”, but we still want to show how to deploy.
Let’s just pretend the following is in the second company’s data center:
A standard “Developer Sandbox” Red Hat OpenShift environment
See also: Start exploring in the Developer Sandbox for free.⁴⁶

 - ▶ Configuration of the utility Red Hat OpenShift cluster in the second company’s data center for the health-checker deployment

Note: in this guide, it will be actually:
Configuration of the standard “Developer Sandbox” Red Hat OpenShift environment

4.2.3. Sample: During the iteration

Note: A possible MVP implementation can be found in the following personal repository published under MIT license : <https://github.com/mikhailknyazev/automation-samples>

During the iteration, the “sample project” team followed up the action plan as follows.

- ▶ **Investigate HAProxy load balancer configuration**

Useful:

- ▶ Ansible Role - <https://github.com/geerlingguy/ansible-role-haproxy>
- ▶ Code repository for “Ansible for Real-Life Automation”, published by Packt <https://github.com/PacktPublishing/Ansible-for-Real-life-Automation>
- ▶ HAProxy community edition <https://www.haproxy.org>

```
---
- name: Deploy Load Balancer using HAProxy
  hosts: loadbalancer
  become: yes
  vars:
    haproxy_frontend_name: 'hafrontend'
    haproxy_backend_name: 'habackend'
    haproxy_backend_servers:
      - name: node1
        address: node1:80
      - name: node2
        address: node2:80
  tasks:
    - name: Install haproxy
      include_role:
        name: geerlingguy.haproxy
```

⁴⁶ Start exploring in the Developer Sandbox for free <https://developers.redhat.com/developer-sandbox>

See also: <https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/haproxy-plus-health-checker.yaml>

- ▶ **Define the functional tests for the app. Implement them in Ansible.**

```
- name: Verify deployment
  hosts: "web"
  become: no
  tasks:
    - name: Verify application health
      delegate_to: localhost
      ansible.builtin.uri:
        url: http://{{ inventory_hostname }}
        status_code: 200
        return_content: true
        register: response
    - name: Check if 'Serving from...' is in the response
      delegate_to: localhost
      ansible.builtin.assert:
        that: "'Serving from {{ inventory_hostname }}' in response.content"
        fail_msg: "The phrase 'Serving from {{ inventory_hostname }}' was not found in the response"
```

See also: <https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/app-deploy.yaml>

- ▶ **Design and implement the Ansible Playbook which performs:**
 - ▶ **Automated rolling updates of new app versions**
 - ▶ **Automated patching and security remediation of the two Linux boxes (during rolling updates)**
 - ▶ **Execute automated functional tests before as well as after the app upgrade**

```
---
- name: Rolling Update
  hosts: "web"
  become: yes
  serial: 1
  vars:
    haproxy_backend_name: 'habackend'
    application_repo: 'https://github.com/mikhailknyazev/automation-samples'
    application_branch: main
    subfolder_path: sample-app
    application_path: /var/www/html

  tasks:
    - name: Preparing rolling deployment of sample App
      ansible.builtin.debug:
        msg: >
          Branch: {{ application_branch }}
          Subfolder: {{ subfolder_path }}
          Repo: {{ application_repo }}
```

See also: <https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/rolling-update.yaml>

► **Configuration of EDA for GitOps-style rolling deployment of the app**

```
yum update -y
yum install -y vim git wget java-17-amazon-corretto-devel sshpass

yum -y groupinstall "Development Tools"
yum -y install openssl-devel bzip2-devel libffi-devel

wget https://www.python.org/ftp/python/3.9.17/Python-3.9.17.tgz
tar xvf Python-3.9.17.tgz
cd Python-*/
./configure --enable-optimizations
make altinstall

cd ~
python3.9 --version
pip3.9 --version

/usr/local/bin/python3.9 -m pip install --upgrade pip
pip3.9 --version
pip3.9 install ansible-rulebook ansible ansible-runner wheel openshift

# Note: we are opening port 5000 for incoming webhook calls on port 5000
yum install -y firewalld
systemctl enable --now firewalld
firewall-cmd --add-port=5000/tcp --permanent
firewall-cmd --reload
firewall-cmd --list-ports

sudo -u devops /usr/local/bin/ansible-galaxy collection install community.general ansible.eda
```

See also: <https://github.com/mikhailknyazev/automation-samples/blob/main/sample-infra/user-data-ansible-engine-with-eda.sh>

► **Implementation and deployment of the rulebook.**

The standard “ansible.eda.webhook” event source will be listening for “push” events from GitHub for the app repository via EDA. Once such an event is received, EDA should initiate the automated rolling update of the app using the above-mentioned Ansible Playbook.

```
---
- name: Listen for events on a webhook
  hosts: all

  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5000
        filters:
          - ansible.eda.dashes_to_underscores:

  rules:
    - name: Handle Webhook event
      condition: >
        event.payload is defined and event.meta.headers.X_GitHub_Event == "push"
        and event.payload.ref == vars.application_branch_in_webhook_event
        and event.payload.repository.url == vars.application_repo
      # condition: event.payload is defined

    action:
      run_playbook:
        name: rolling-update.yaml
        extra_vars:
          application_repo: "{{ application_repo }}"
          application_branch: "{{ application_branch }}"
          subfolder_path: "{{ subfolder_path }}"
```

See also: <https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/webhook-for-rolling-update.yaml>

- ▶ **Find somewhere to provision environments that will support deployment of the MVP without impacting current systems in the data centers. The engineering team should be able to start immediately.**
 - ▶ **AWS configuration to deploy the app instances and the HAProxy load balancer instance**
 - ▶ **Implement Terraform code to deploy the datacentre-like three EC2 Linux instances**

Useful: "Use Terraform to Create a FREE Ansible Lab in AWS" <https://www.techbeatly.com/use-terraform-to-create-a-free-ansible-lab-in-aws/>

```

locals {
  connection = {
    type      = "ssh"
    user      = "ec2-user"
    private_key = file(pathexpand(var.ssh_pair_private_key))
  }
}

resource "aws_instance" "ansible-engine" {
  ami           = var.aws_ami_id
  instance_type = "t2.micro"
  key_name      = aws_key_pair.ec2Loginkey.key_name
  security_groups = ["sample-mvp-sg", "sample-mvp-webhooks-sg"]
  user_data     = file("user-data-ansible-engine-with-eda.sh")

  // Copy the "get-automation-sample-playbooks.sh" script to the remote machine
  provisioner "file" {
    source      = "files-for-upload/get-automation-sample-playbooks.sh"
    destination = "/home/ec2-user/get-automation-sample-playbooks.sh"
    connection {
      type      = local.connection["type"]
      user      = local.connection["user"]
      private_key = local.connection["private_key"]
      host      = self.public_ip
    }
  }
}

```

See also: <https://github.com/mikhailknyazev/automation-samples/tree/main/sample-infra>

- ▶ **For the health-checker application:**
 - ▶ **Design and implement the health-checker application (in JavaScript/Typescript)**
 - ▶ **Investigate API for integration with Red Hat OpenShift**
 - ▶ **Configuration of the standard "Developer Sandbox" Red Hat OpenShift environment**

Useful:

- ▶ Part 2: Deploying full-stack JavaScript applications to the Developer Sandbox for Red Hat OpenShift <https://developers.redhat.com/developer-sandbox/activities/deploying-full-stack-javascript-applications-to-the-sandbox/part2>
- ▶ Chapter "Managing Kubernetes Using Ansible" in "Ansible for Real-Life Automation" by Gineesh Madapparambath <https://amzn.asia/d/fIHSDjh>
- ▶ "Ansible for Kubernetes by Example: Automate Your Kubernetes Cluster with Ansible" by Luca Berton <https://a.co/d/7tJWteG>

```

- name: Display list of all Pods from the current Namespace (Project)
  kubernetes.core.k8s_info:
    kubeconfig: "{{ openshift.kubeconfig_file }}"
    kind: Pod
    namespace: "{{ openshift.namespace_name }}"
  register: pod_list
- name: Print pod_list
  debug:
    var: pod_list

- name: Deploy the Health-Checker with OpenShift feature "Source-to-Image" (S2I)
  ansible.builtin.command: >
    oc --kubeconfig={{ openshift.kubeconfig_file }} new-app
    https://github.com/mikhailknyazev/automation-samples --context-dir=sample-health-checker --name={{ health_checker_label }}
    -e HAPROXY_PUBLIC_IP={{ haproxy_public_ip }}

```


See also: <https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/health-checker.yaml>

```
async function fetchData() : Promise<void> {
  const haproxy_public_ip = document.getElementById('haproxy_public_ip').value;
  const response : Response = await fetch(`http://${haproxy_public_ip}`, {
    method: 'GET',
    headers: {
      'Cache-Control': 'no-cache'
    }
  });
  const data : string = await response.text();

  let lines : string[] = data.split('\n');
  let resultIndex : number = lines.findIndex(e : string => e.includes("Serving from"));

  let str;
  if (resultIndex === -1) {
    str = "No matches found";
  } else {
    str = lines.slice(resultIndex).join('\n').replace(/<[^>]+>/g, ' ');
  }

  const listItem : HTMLLIElement = document.createElement('li');
  listItem.textContent = str;
  document.getElementById('resultList').appendChild(listItem);
}

// Call fetchData every 5 seconds
setInterval(fetchData, 5000);
```

See also: <https://github.com/mikhailknyazev/automation-samples/blob/main/sample-health-checker/public/script.js>

4.2.4. Sample: The showcase (demo) for iteration

Note: Some screenshots can be found in the personal repository published under MIT license: <https://github.com/mikhailknyazev/automation-samples/blob/main/README.md>

At the showcase (demo) at the end of the first delivery iteration, the “sample project” stakeholders and interested parties were given a demonstration of recent work performed during the increment (iteration) and gave their feedback. In general, they were impressed with the results pertaining to the goal of the MVP: “Improvements to the app deployment, patching, observability, and performance.”

In particular:

- ▶ The stakeholders estimated that some of the demonstrated automation techniques can be reused organization-wide and lead to reduction of manual IT effort by 50-80% over time
- ▶ The app deployment is now fully automated. So that, the infrastructure team can realistically plan shifting their ways of working toward the efficient “Site Reliability Engineer” model.
- ▶ The stakeholders appreciated that thanks to the architectural changes pertaining to load balancing, compute capacity can now be added to the app on demand easier than before
- ▶ The demonstrated “GitOps with Ansible” technique should lead to increased app development productivity and improved traceability of changes for mission-critical Linux workloads
- ▶ The “health-checker” remote component improves observability for the app, eventually improving stability of the service delivered to the users. The stakeholders were impressed how simple it was to start using the Source-to-Image (S2I) feature of Red Hat OpenShift to meet the project goals.

Speaking in terms of what could have been done better for the MVP, the stakeholders mentioned they would have benefited from a more developed patching and security remediation demonstration. However, they appreciate that improvements to it are now much easier than before, thanks to the demonstrated updating of Linux packages during the app rolling update.

5. What Red Hat offers to support customers

DISCLAIMER: This section describes solely personal, individual opinions of this guide's authors. The information in it is given "as is," without any warranty. It can be outdated at the time when you read this. The authors are not liable for anything related to this information. The authors *guess* it can be helpful to support use of the other relevant Red Hat resources.

This section describes some of Red Hat's commercial product and service offerings related to the context of this guide.

5.1. Better confidence in your automation or container adoption journey with Red Hat Open Innovation Labs

Red Hat Open Innovation Labs can assist organizations embark on an automation or container adoption journey! Customer's people will learn to build great products the open source way – Red Hat's immersive residency pairs their engineers with open source experts, transforming customer's ideas into business outcomes.

In a 4-12 week Open Innovation Labs residency, they'll learn to connect their team's ideas with the best that open source communities have to offer. Red Hat experts bring deep experience with Red Hat technologies, open source communities, and the key transformative practices needed to unlock customer's teams' potential.⁴⁷

The World Health Organization (WHO) embarked on an eight-week Red Hat Open Innovation Labs residency, held virtually, to create a DevOps platform. Here is a short video summarizing it.⁴⁸ It is also worth checking the following engaging presentation to have an idea what professionals drive the engagements: "Turning stories into software"⁴⁹ by Donna Benjamin.

5.2. Lots of moving parts at scale? Consider using Red Hat Ansible Automation Platform

Red Hat Ansible Automation Platform⁵⁰ is a product that helps you manage multi-tier architectures, heterogeneous IT environments, and user permissions. It is designed for organization-level IT management and provides a convenient UI dashboard which shows summaries of all the managed hosts, has easy navigation to the automation functionality of all sorts. It covers flexible governance and security needs of organizations. AAP is highly "embeddable" into existing standard approaches, for example, using Active Directory for authentication and authorization. It gives the IT teams best of class observability for Ansible Playbooks execution.

For example, with Ansible Automation Platform, you can:

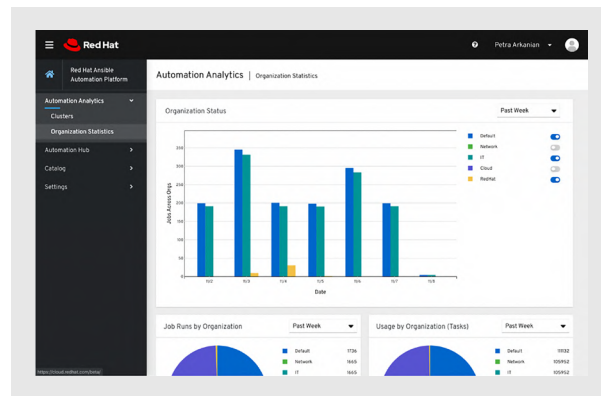
- ▶ Secure your automation through role-based access control and change request management
- ▶ Eliminate siloed teams and processes by consolidating on a single, flexible platform
- ▶ Scale your automation anywhere – on premise, in the cloud, or at the network edge

⁴⁷ Red Hat Open Innovation Labs <https://www.redhat.com/en/services/consulting/open-innovation-labs>

⁴⁸ WHO built this: Creating an innovative learning platform together <https://www.youtube.com/watch?v=f69I6Vo9rGk>

⁴⁹ Turning stories into software <https://www.youtube.com/watch?v=J3VaaPYshek>

⁵⁰ Red Hat Ansible Automation Platform <https://www.redhat.com/en/technologies/management/ansible>



AAP's Workflows allow you to chain any number of playbooks, regardless of the usage of different inventories, utilize various credentials, or run under different users. See also: Chapter 23. Workflows.⁵¹

AAP exposes a secure REST API and corresponding CLI tools. This all makes integration with say such ITSM tools as ServiceNow and with CI systems like Jenkins an easy task. For example, you can launch an automation job (managed playbook) via the API or CLI.

AAP adds secure storage of all your credentials for machines and cloud systems, and a powerful role-based access control engine that allows you to easily set policies on who can run what automation in what environments, ensuring that only the proper people have the ability to access machines and apply configuration. AAP itself is a distributed system, with a lot of engineering thought put into making it a highly available platform.

AAP is being actively developed, for example, according with "What's new in Ansible Automation Platform 2.4,"⁵² the following are the new/revised additions to AAP:

EDA; collection repository management; validated content integration; Ansible Builder 3.0; platform install support for ARM; The Ansible Lightspeed with IBM Watson Code Assistant technical preview is now available.

6. Conclusion

Thank you for your interest in efficient Automation with Ansible!

In the beginning of this guide based on a case study, we described why Ansible is a good friend of DevOps tools and can serve as universal "glue" for IT artifacts. We presented some organization-level automation ideas as well as how automation can help with everyday repetitive tasks.

In the main parts of the guide, we explained and demonstrated how practices of "Discovery Loop," "Options Pivot," and "Delivery Loop" can help achieve success with automation projects. Finally, we described some relevant offerings by Red Hat.

We hope it is all useful for you!

⁵¹Chapter "23. Workflows" <https://docs.ansible.com/automation-controller/latest/html/userguide/workflows.html>

⁵²What's new in Ansible Automation Platform 2.4 <https://www.ansible.com/blog/whats-new-in-ansible-automation-platform-2.4>

7. About the authors



Michael Knyazev
Red Hat

Michael Knyazev is an experienced Hands-on Architect / DevOps Consultant specialising in OpenShift, Ansible, AWS, GCP, Kafka.

Michael has played central roles in delivering valuable outcomes for major Red Hat customers in APAC region. Prior to Red Hat, he co-designed and built several world-class solutions. Some of them are the Kubernetes-based Platforms for TPG Telecom and for Insurance Australia Group; the Australian National Coronavirus Helpline cloud solution, the Morningstar Tick Data and Beam Wallet FinTech platforms.

Michael enjoys contributing to the Community! He served as the primary contributor to Red Hat's "Efficient automation with Ansible" e-book, built a popular Udemy course "Configuring Kubernetes for Reliability", achieved scientific results with his Ph.D. thesis about automated development of IT architectures he defended back in 2006.

A keen learner, Michael also holds an MBA as well as more than 15 professional certifications. He spoke at the global chaos engineering conferences in recent years and JavaOne in the past.

He lives with his family in Sydney, Australia.



Gineesh Madapparambath
Red Hat

Gineesh Madapparambath is a Platform & DevOps Consultant at Red Hat Singapore. His extensive career has seen him in roles ranging from Systems Engineer to Automation Specialist and content author, with a primary focus on Ansible automation, Containerization (OpenShift and Kubernetes), and Infrastructure as Code (Terraform). Gineesh is the proud author of the book "Ansible for Real-Life Automation." He has a rich history of designing, developing, and deploying automation solutions, employing Ansible and Ansible Automation Platform (formerly Ansible Tower). These solutions encompass various tasks such

as building bare metal/virtual servers, patching, license management, Network Operations, and custom monitoring. Furthermore, he has orchestrated the design and deployment of servers in data centres worldwide, gaining valuable cross-cultural experience across classic, private cloud (OpenStack, VMWare), virtual, and public cloud environments (AWS, Azure, Google Cloud). Throughout his career, Gineesh has held multiple roles, including Systems Engineer, Automation Specialist, Infrastructure Designer, and content author.



Nicholas Wong
Red Hat

Nicholas Wong is a force for good, the "betterment of society" is his life's mission. He takes his responsibilities seriously, working with individuals, teams and enterprises, as a guide, a coach and a mentor. He thrives on learning critical skills that increase his ability to meaningfully connect with others and in so doing, creating beautiful relationships everywhere.

A senior delivery consultant, value stream expert, master coach and certified instructor in several disciplines, his experience comes from the ground up, through the trenches as a quality assessment consultant, system administrator and technical delivery lead.

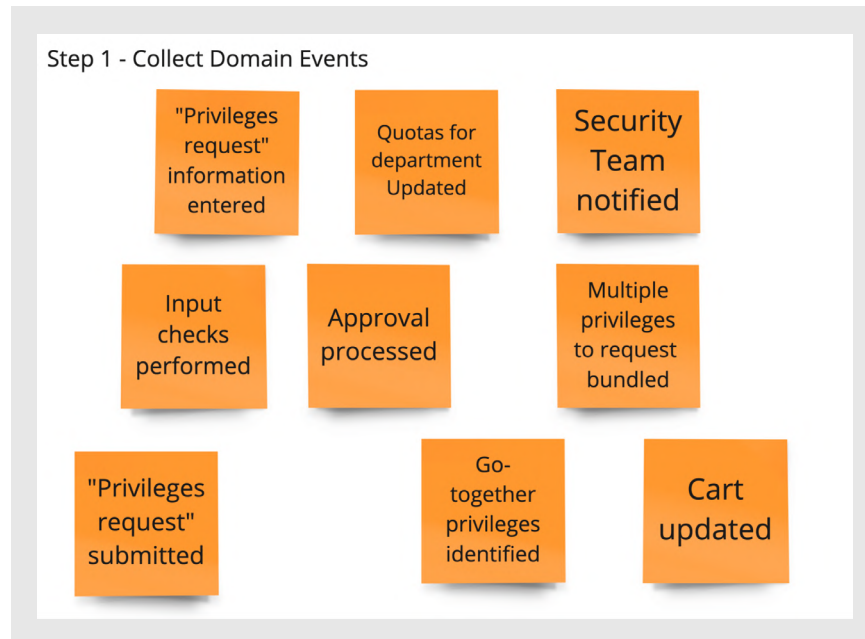
Nicholas spends much of his time helping others to realize their best selves, face new/existing challenges and achieve their goals. He does this with great awareness of what the learner needs and customizes his offerings accordingly. Outside of work, he can often be found giving one to one coaching for executives and high performers.

Nicholas is also active in the DevOps, agile coaching and corporate/personal coaching communities. He strongly believes that knowledge and expertise should be available to all to benefit from, and enjoys working with leaders from all walks to lead highly engaged, motivated and autonomous teams into the future.

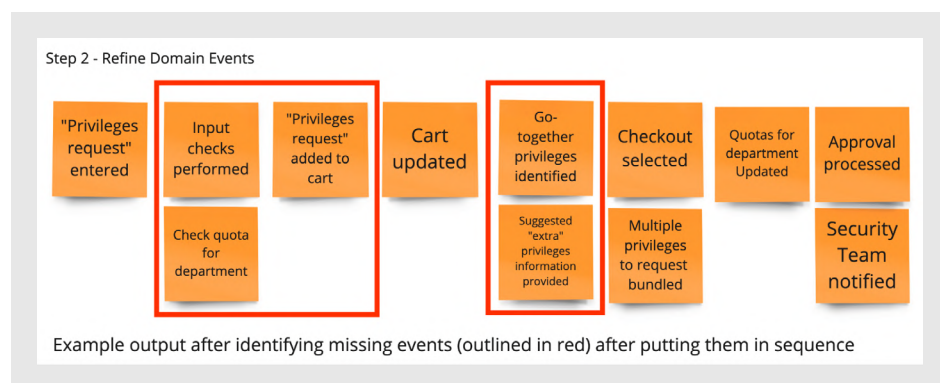
"Improving lives with people and outcomes" ~ Nicholas Wong

Appendix A. Event Storming for "App in use – requesting access privileges"

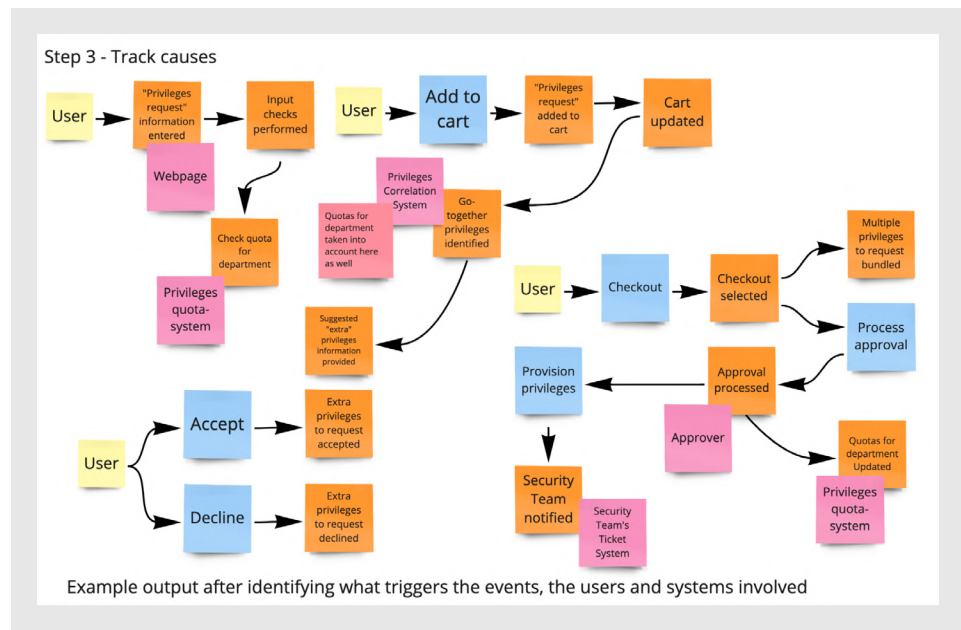
During the "Step 1 - Collect domain events (Big Picture)," each participant uses only orange post-its.



During the "Step 2 - Refine domain events (Big Picture)," go through the domain event post-its with the participants. Ask the participants to explain what each event means. Check for syntactical correctness. Also discuss again whether the events are in the right order in terms of time. Unify occurring synonyms (different terms for the same thing) and sharpen differences if the same term was used to describe different things.

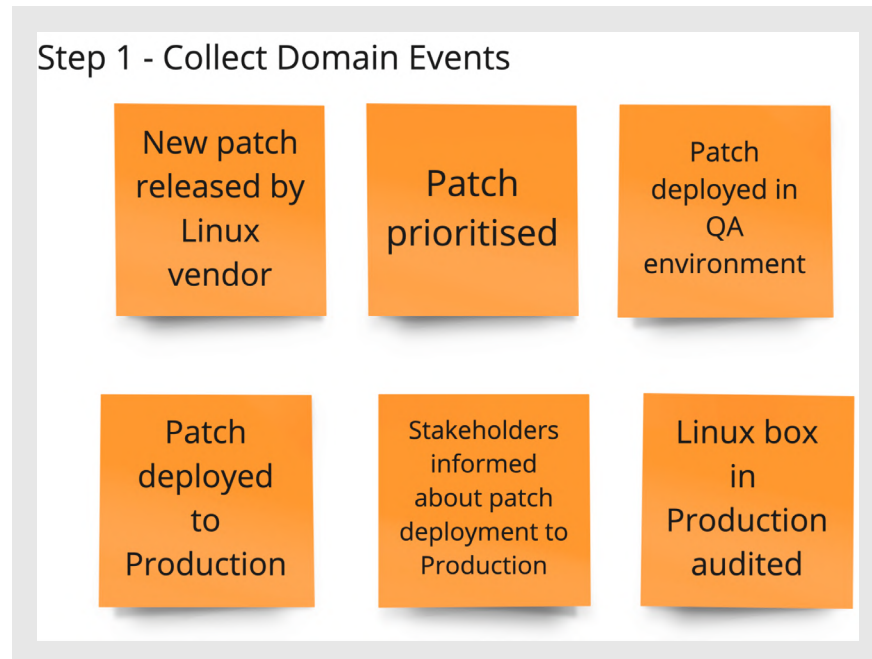


During the "Step 3 - Track causes (Process Modeling)," get into the cause analysis. Where do the domain events come from?

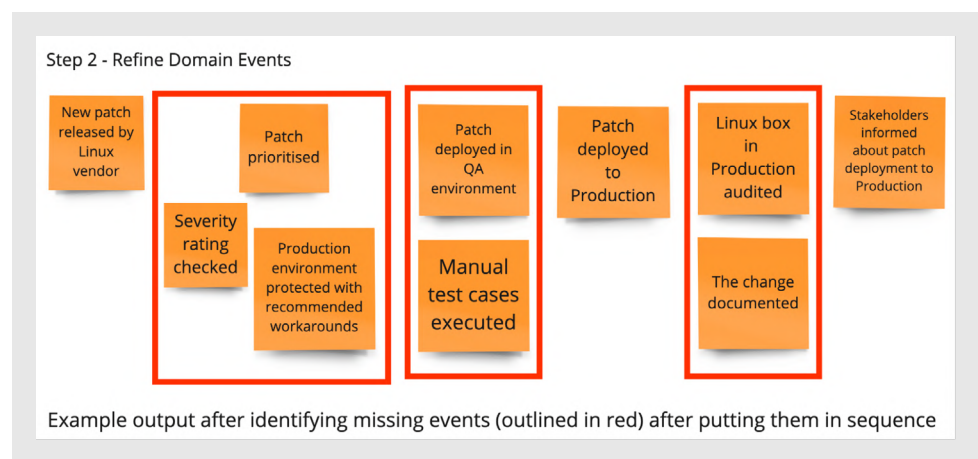


Appendix B. Event Storming for “Patching the Linux box of app”

During the “Step 1 - Collect domain events (Big Picture),” each participant uses only orange post-its.



During the “Step 2 - Refine domain events (Big Picture),” go through the domain event post-its with the participants. Ask the participants to explain what each event means. Check for syntactical correctness. Also discuss again whether the events are in the right order in terms of time. Unify occurring synonyms (different terms for the same thing) and sharpen differences if the same term was used to describe different things.



During the “Step 3 - Track causes (Process Modeling),” get into the cause analysis. Where do the domain events come from?

