adaptTo()

APACHE SLING & FRIENDS TECH MEETUP
10-12 SEPTEMBER 2018

When product meets real world data

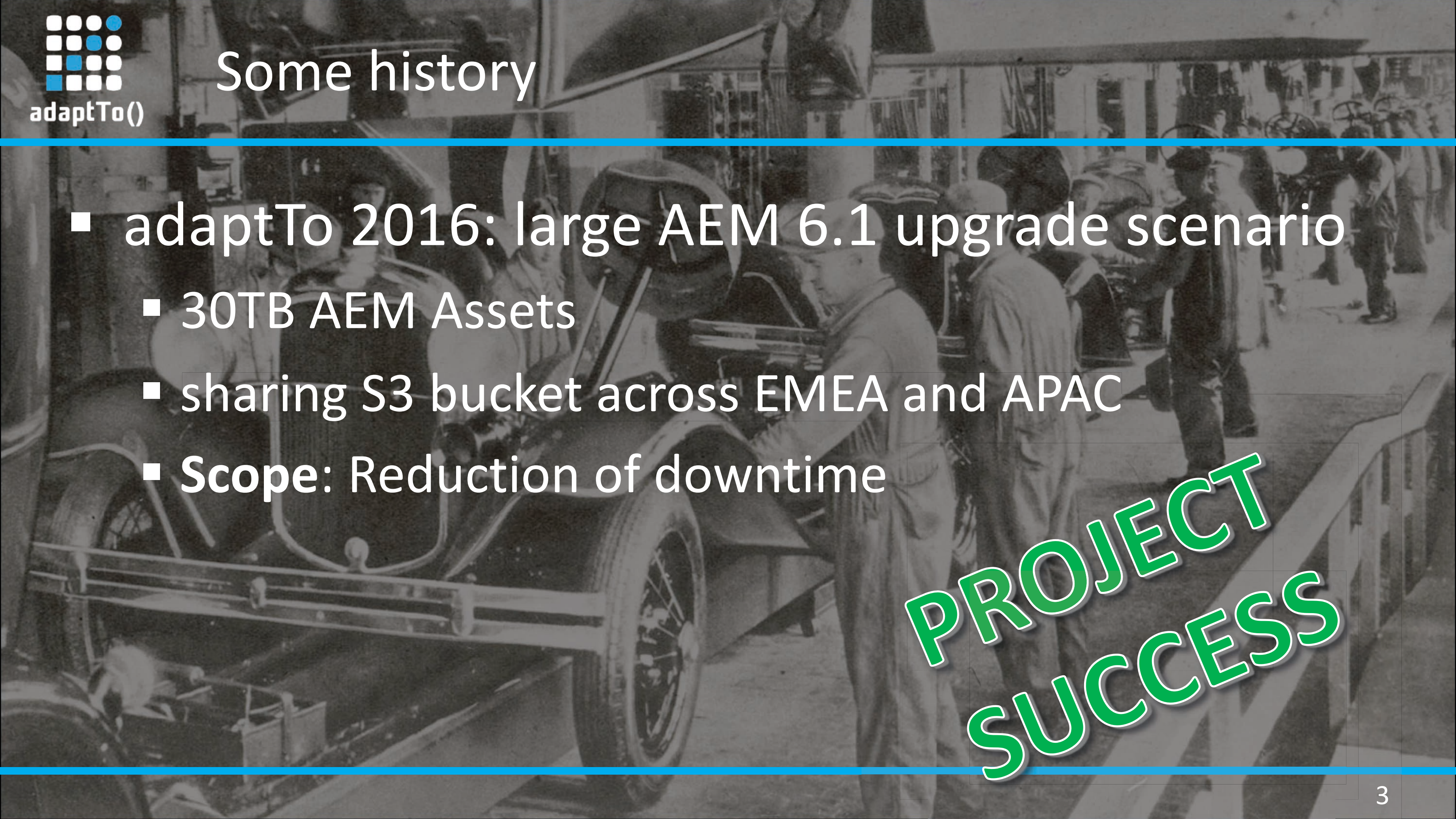
Tom Blackford, Dominik Suess - Adobe

adaptTo()

Ancient times

- pre AEM 6.2 -

Some history



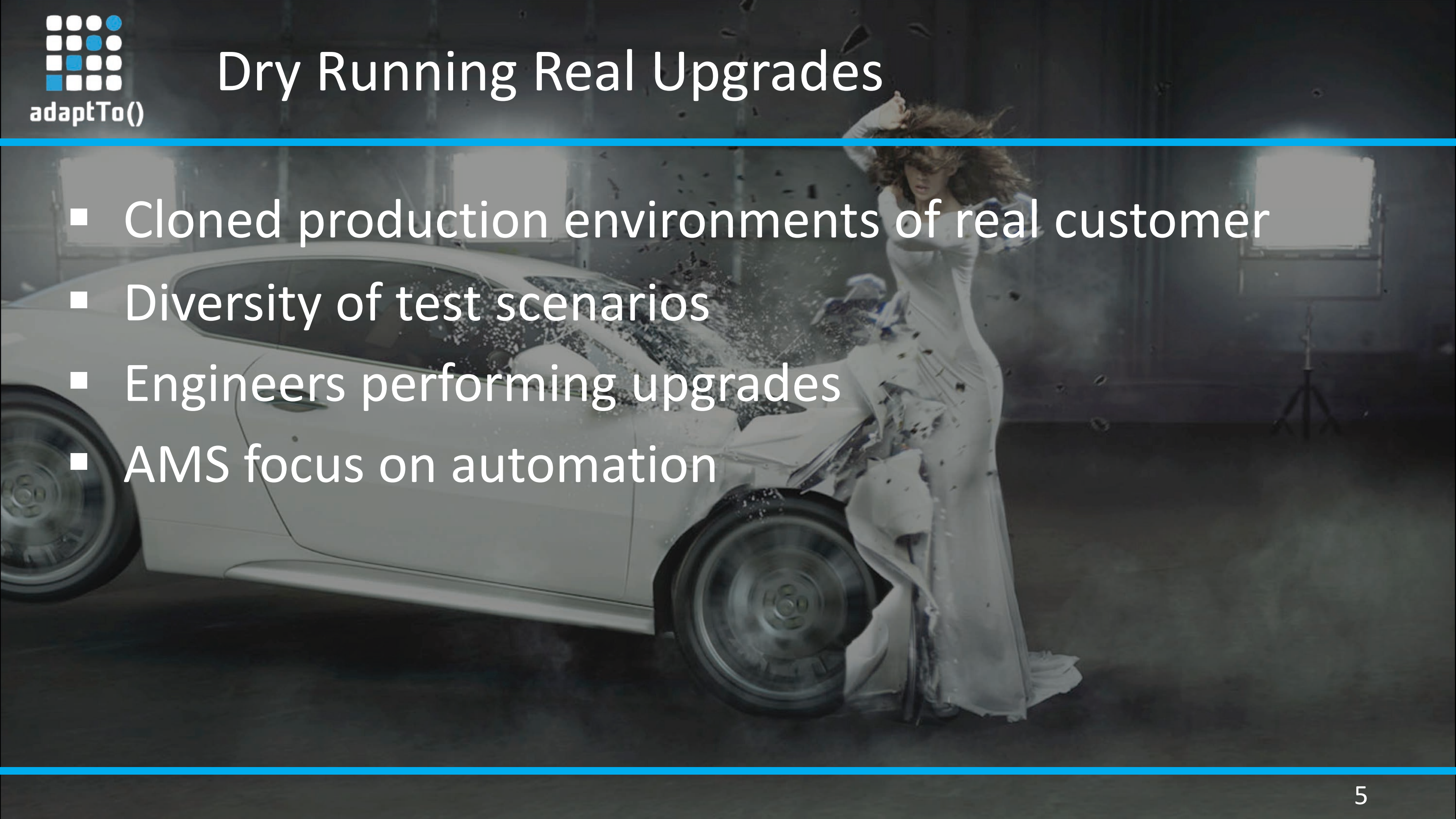
- adaptTo 2016: large AEM 6.1 upgrade scenario
  - 30TB AEM Assets
  - sharing S3 bucket across EMEA and APAC
  - **Scope**: Reduction of downtime

PROJECT SUCCESS

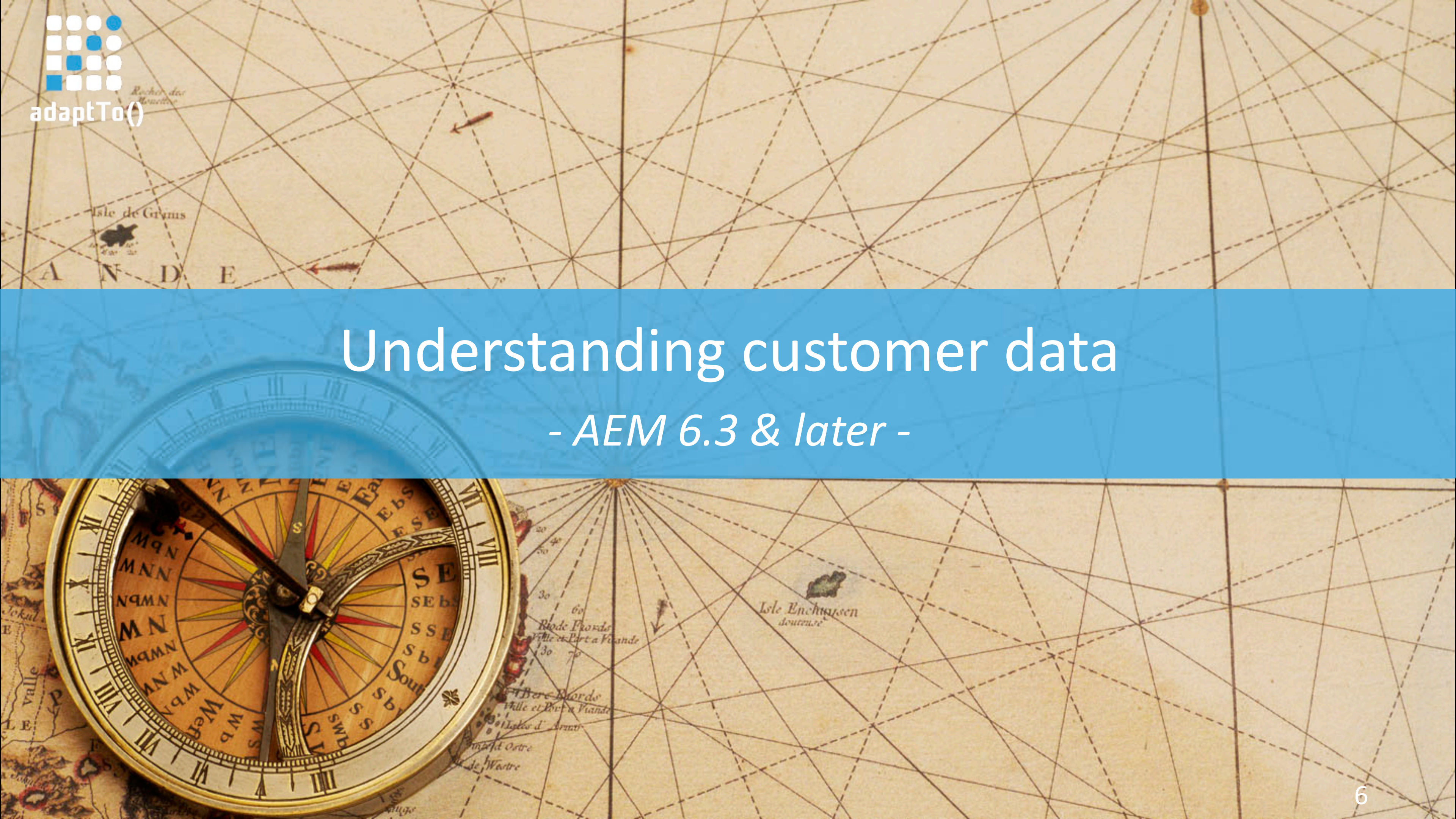
Dry running real upgrades

- AEM 6.2 & later -

Dry Running Real Upgrades



- Cloned production environments of real customer
- Diversity of test scenarios
- Engineers performing upgrades
- AMS focus on automation



Understanding customer data

- AEM 6.3 & later -

adaptTo()

Measuring Usage & Acting on it

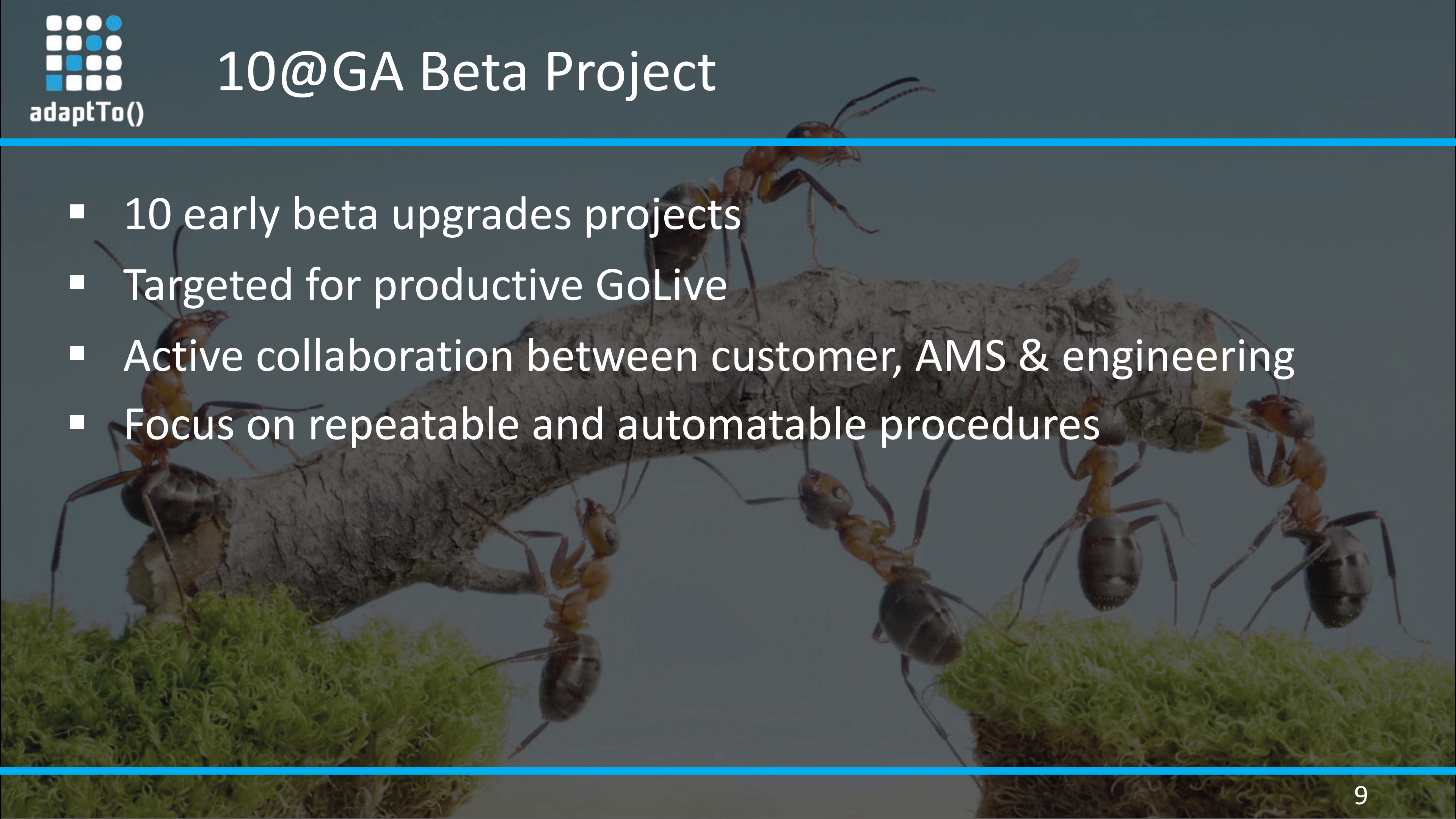
- Significant Sampleset
- Static analysis of code usage patterns
- Automated extraction & aggregated reporting
- Making data actionable:
  - Data-driven decission making
  - Pattern Detector

Feeling & living reality

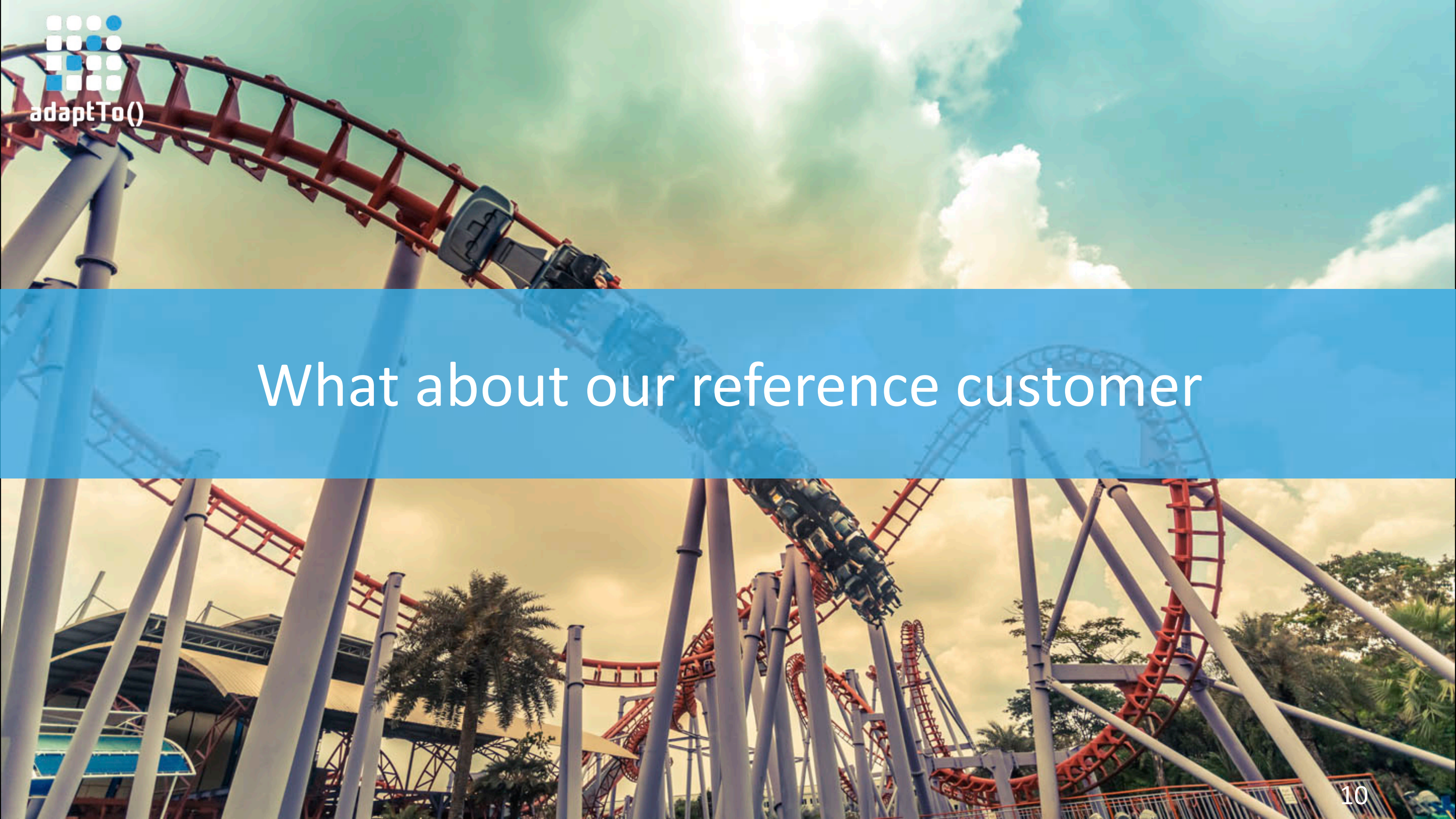
- AEM 6.4 & later -

adaptTo()

10@GA Beta Project



- 10 early beta upgrades projects
- Targeted for productive GoLive
- Active collaboration between customer, AMS & engineering
- Focus on repeatable and automatable procedures



What about our reference customer

Condé Nast International (10@GA)

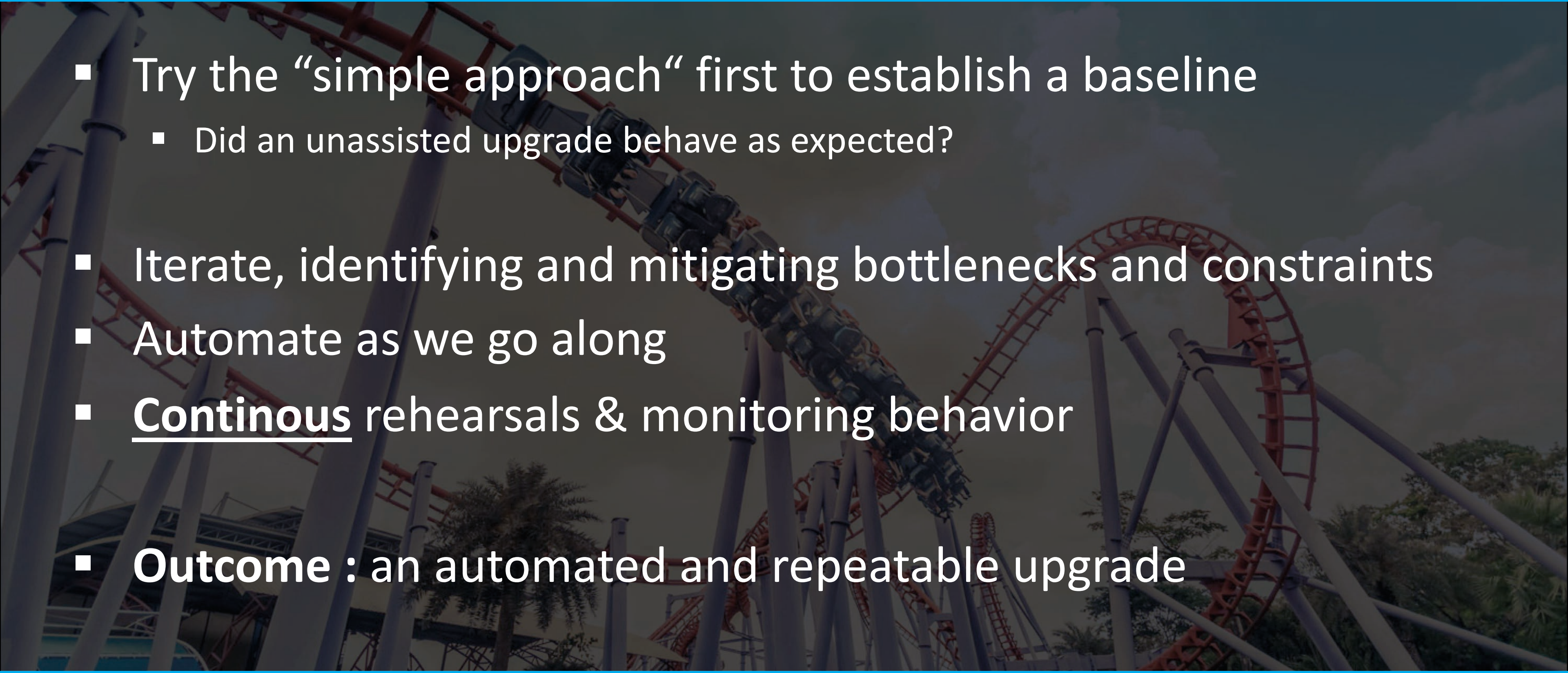
- Increased Challenge: > 2 years of additional content
  - Production repository grown to 100TB (~4 million assets)
  - Central to day-to-day content syndication and magazine production (200+ daily editors)

- Limited downtime possible
  - Window between EOD Friday in Americas to 9am Monday in CNI's APAC markets

- Automation necessary
  - In-place upgrade, but fully optimized process involved a complex sequence of steps, some taking hours
  - Steps needed to be run 'back-to-back' with automatic validation of the previous step

- Early AEM Engineering involvement allowed optimization options to roll back into product

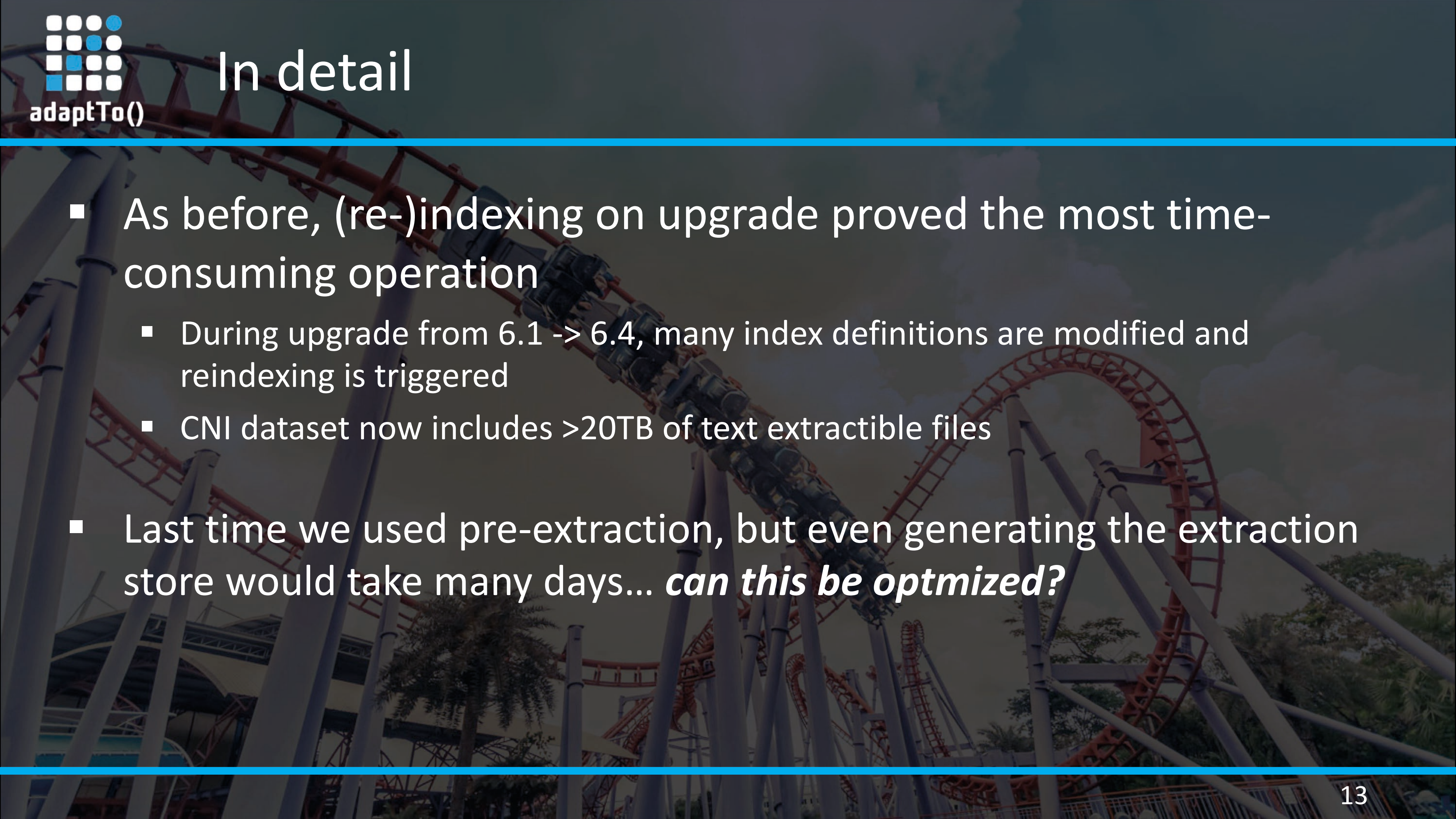
Recipe for success



- Try the “simple approach“ first to establish a baseline
  - Did an unassisted upgrade behave as expected?

- Iterate, identifying and mitigating bottlenecks and constraints
- Automate as we go along
- **Continous** rehearsals & monitoring behavior

- **Outcome :** an automated and repeatable upgrade

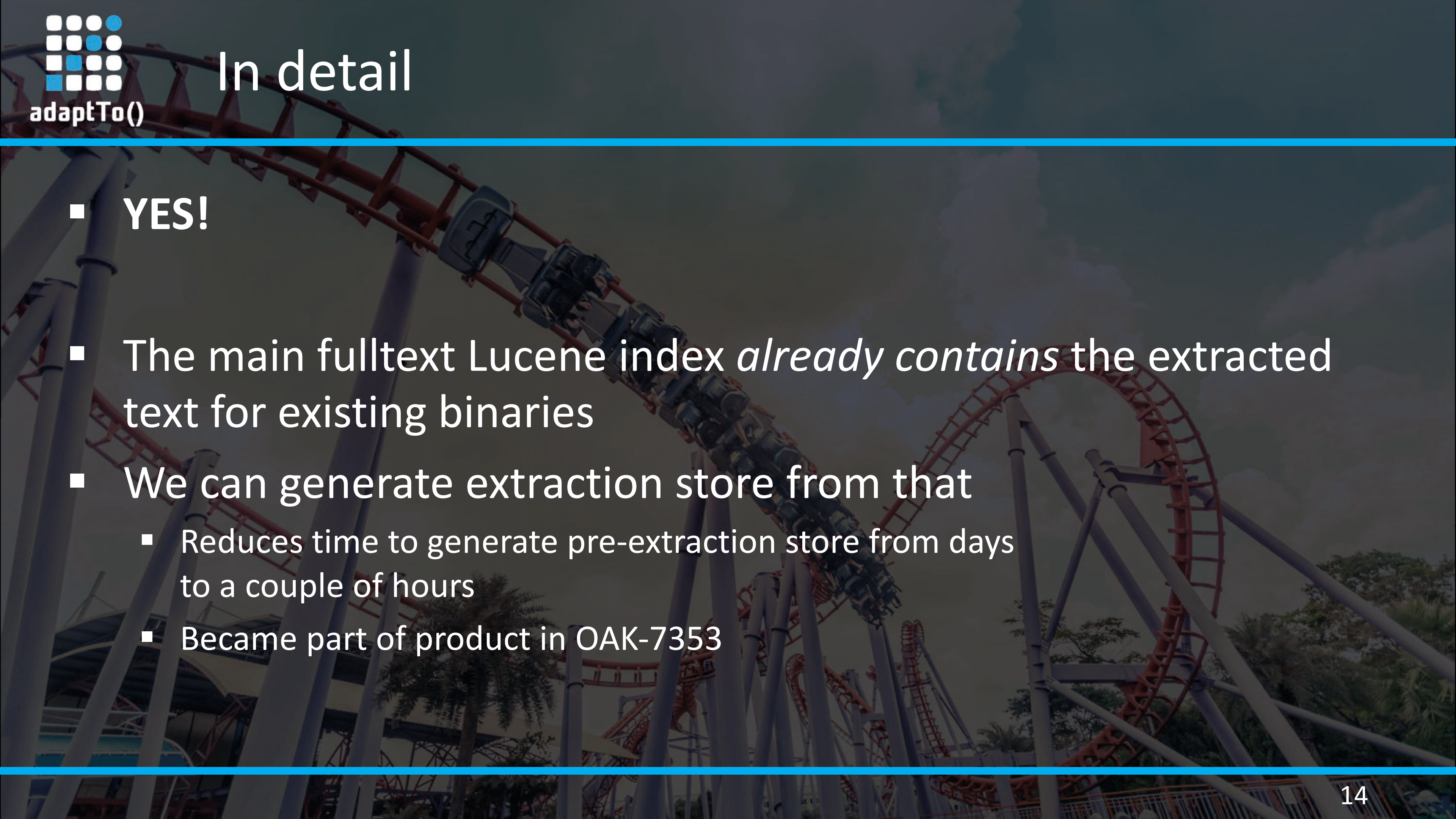


In detail

- As before, (re-)indexing on upgrade proved the most time-consuming operation
  - During upgrade from 6.1 -> 6.4, many index definitions are modified and reindexing is triggered
  - CNI dataset now includes >20TB of text extractible files

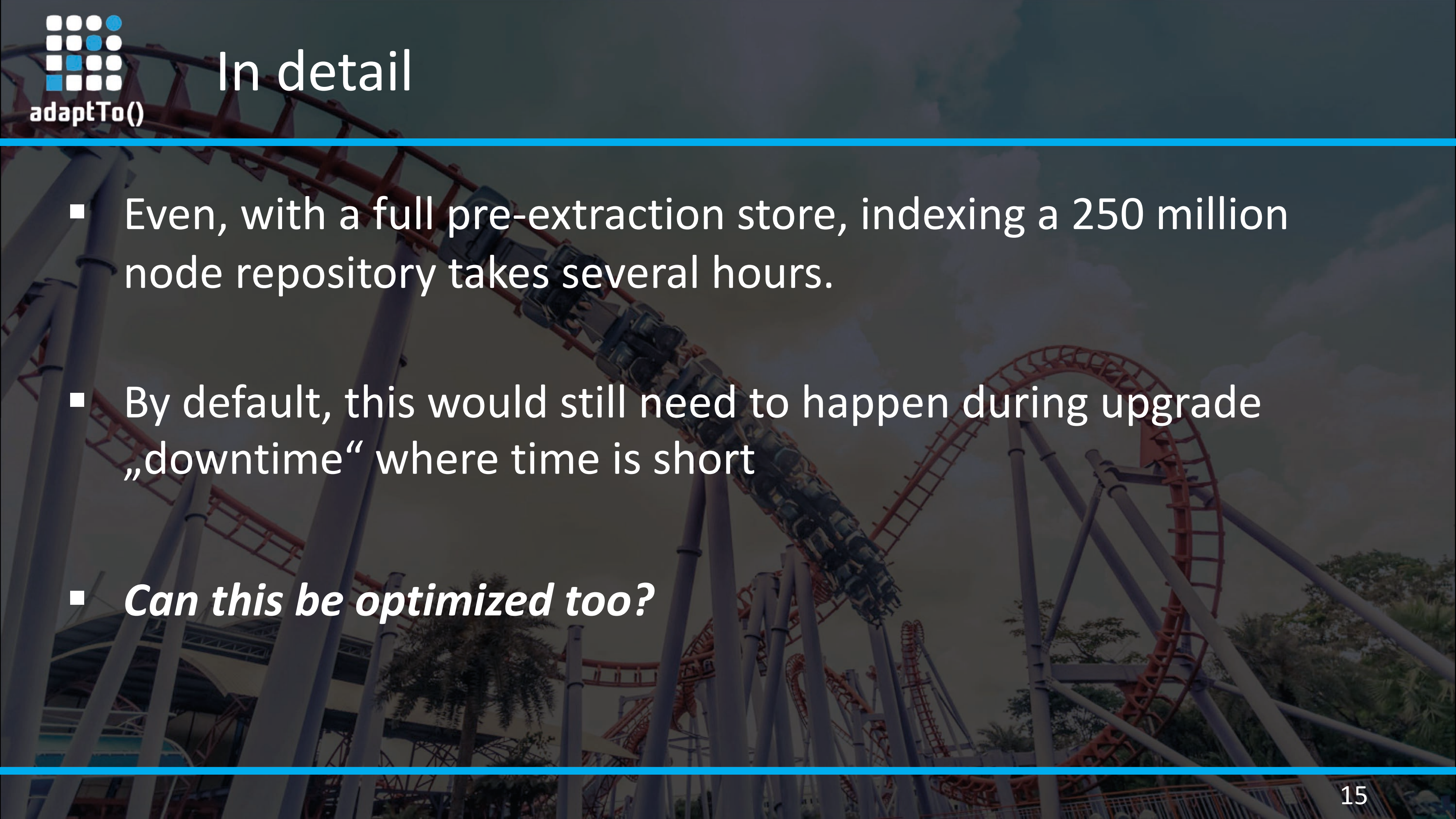
- Last time we used pre-extraction, but even generating the extraction store would take many days… ***can this be optmized?***

In detail



- **YES!**
- The main fulltext Lucene index *already contains* the extracted text for existing binaries
- We can generate extraction store from that
  - Reduces time to generate pre-extraction store from days to a couple of hours
  - Became part of product in OAK-7353

In detail



- Even, with a full pre-extraction store, indexing a 250 million node repository takes several hours.
- By default, this would still need to happen during upgrade „downtime“ where time is short
- ***Can this be optimized too?***

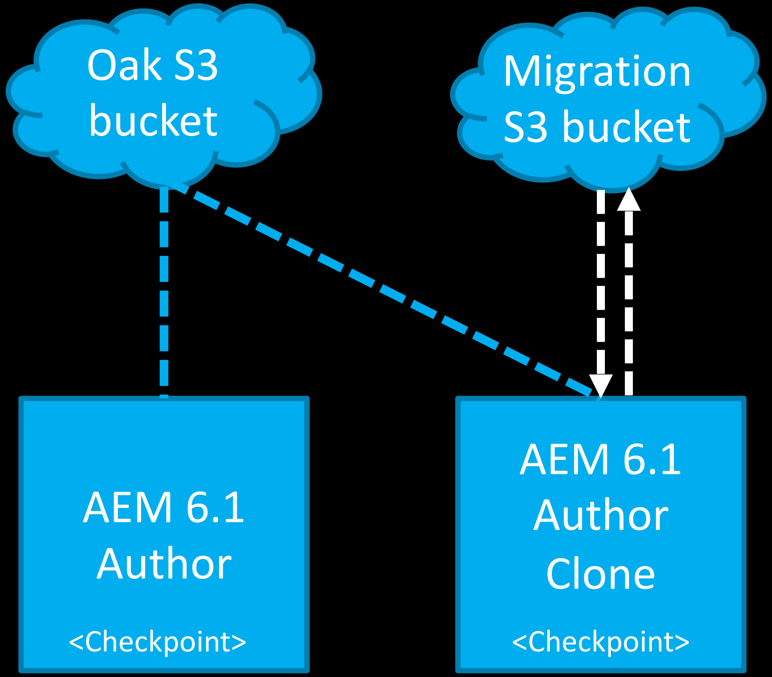
In detail

- **YES!**

- Leverage ‚out of band‘ indexing capabilities of Oak 1.8
  - Perform indexing customer data with new index definitions outside the upgrade window.
  - Indexing run (up to a specific checkpoint) using read-only nodestore connection (typically we would run on a clone)
  - Generated indexes (and updated index definitions) imported
  - Incremental ‚catchup‘ between checkpoint and ‚now‘

Step-by-Step

● DAM available to end users

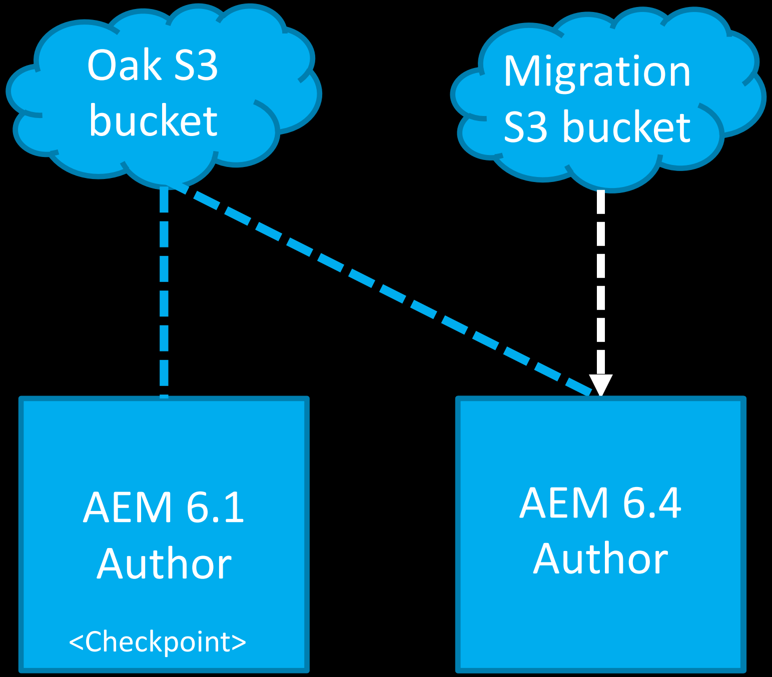


Phase 1

1. Create checkpoint on AEM 6.1 Author
2. Clone author
3. Download existing store & tools from S3
4. Update pre-extraction store from Lucene index
5. Upload store to S3
6. Execute out of band indexing run (using AEM 6.4 index definitions)
7. Upload generated index data to S3

Step-by-Step

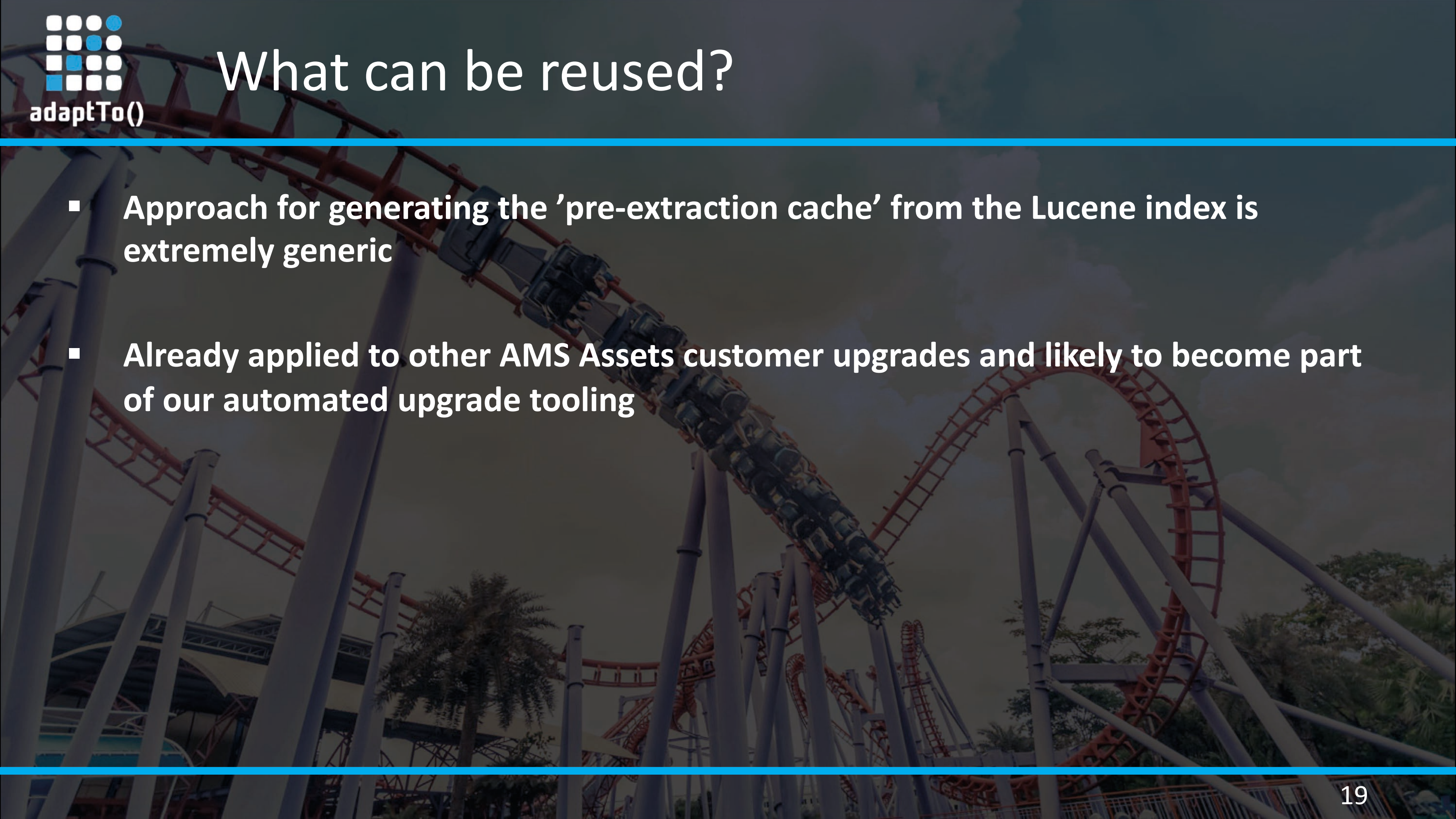
● DAM unavailable



Phase 2

1. Clone author
2. Download store and generated index
3. Place index in AEM repository folder
4. Migrate repository (part of 6.1 -> 6.4 upgrade)
5. Perform upgrade - indexes merged in
6. Smoke test and cutover

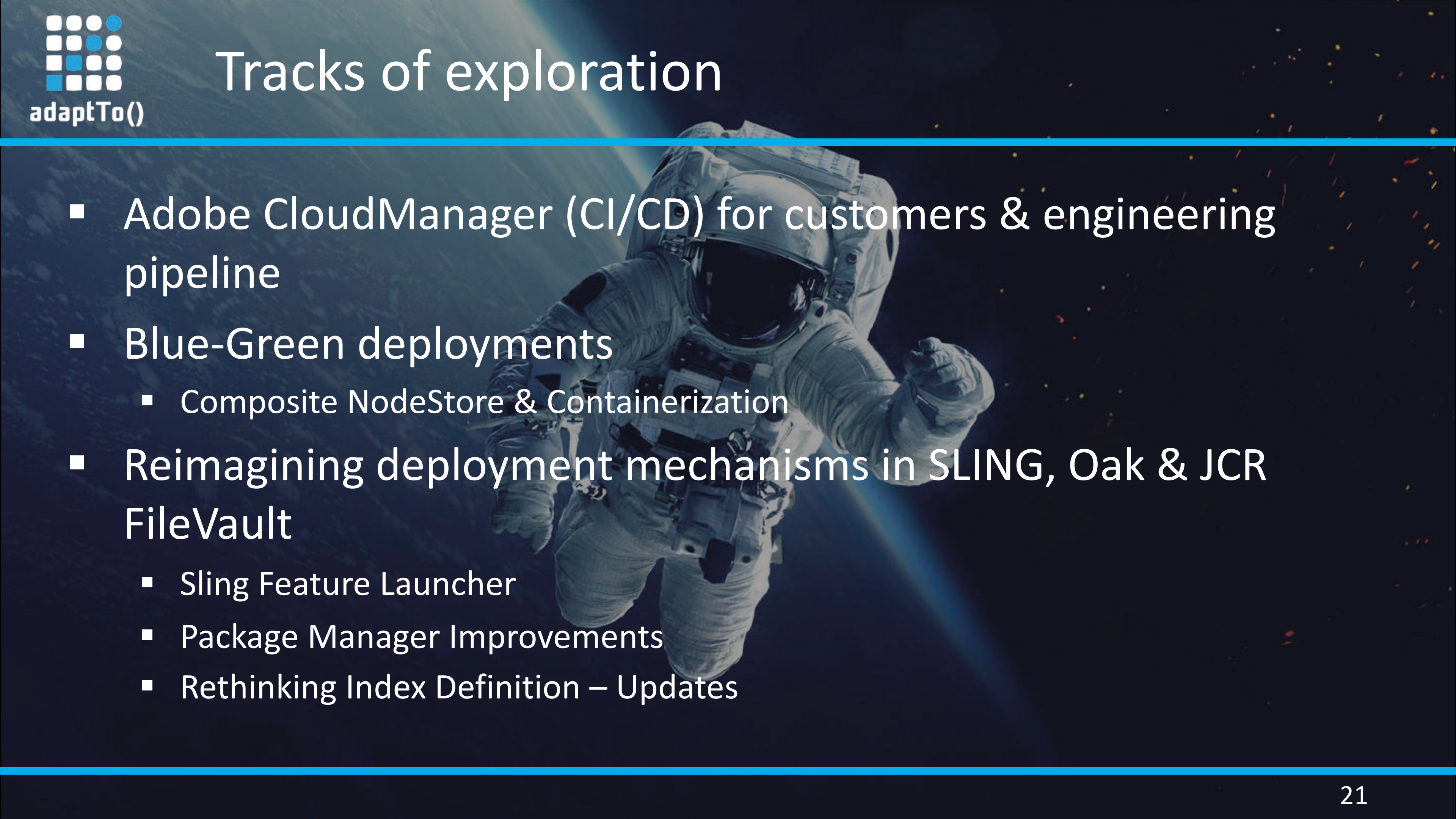
What can be reused?



- **Approach for generating the ’pre-extraction cache’ from the Lucene index is extremely generic**
- **Already applied to other AMS Assets customer upgrades and likely to become part of our automated upgrade tooling**

Exploring what might be coming next…

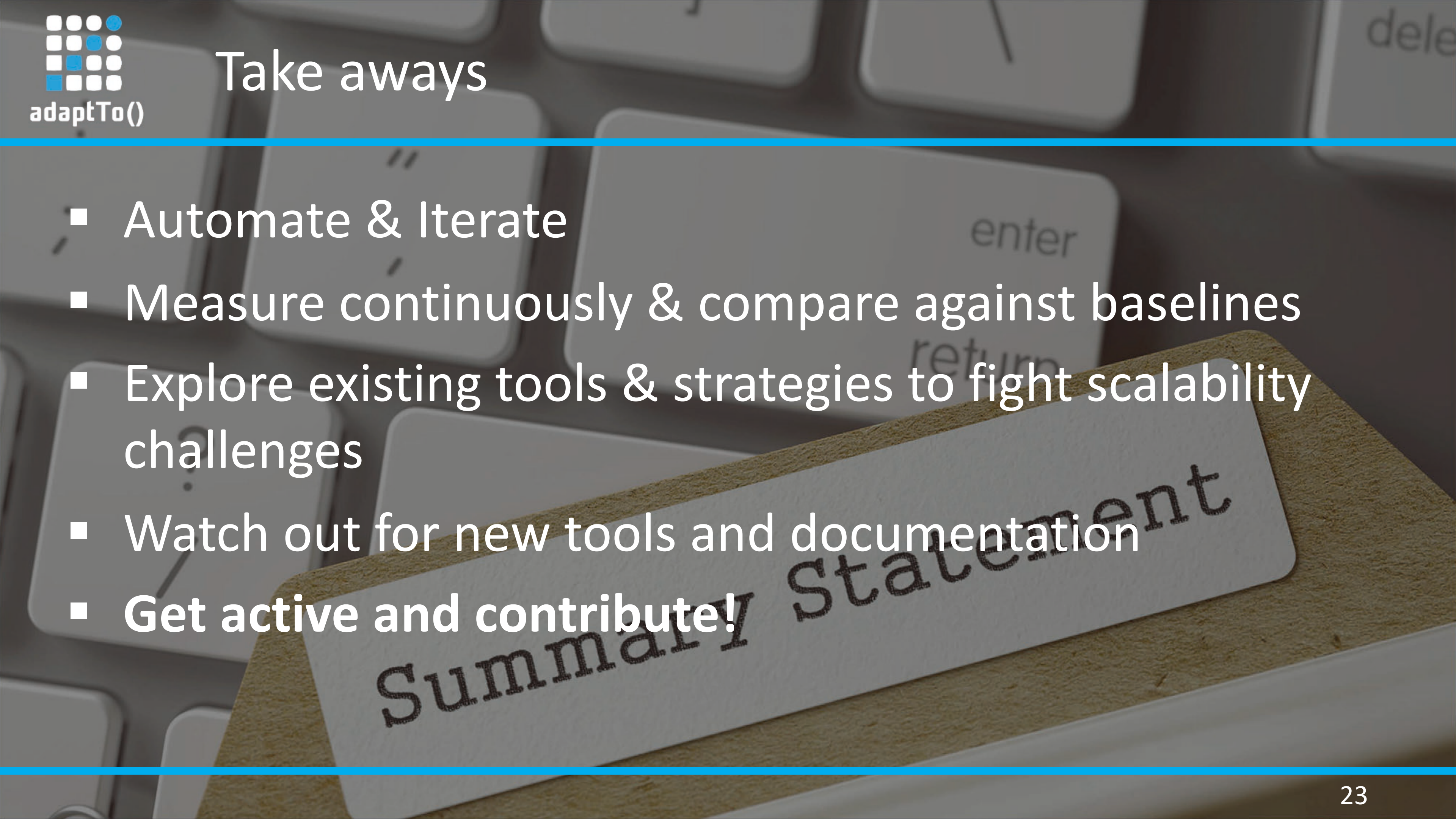
Tracks of exploration



- Adobe CloudManager (CI/CD) for customers & engineering pipeline
- Blue-Green deployments
  - Composite NodeStore & Containerization
- Reimagining deployment mechanisms in SLING, Oak & JCR FileVault
  - Sling Feature Launcher
  - Package Manager Improvements
  - Rethinking Index Definition – Updates

Recap

Take aways



- Automate & Iterate
- Measure continuously & compare against baselines
- Explore existing tools & strategies to fight scalability challenges
- Watch out for new tools and documentation
- **Get active and contribute!**