E-BITUG
European NonStop Symposium
EDINBURGH 2023

Hewlett Packard Enterprise

E-BITUG 2023
The Art of the Possible II

Franz König
Director HPE WW Business Development & Advanced Technology Center (ATC)

May 2023

| 1

1

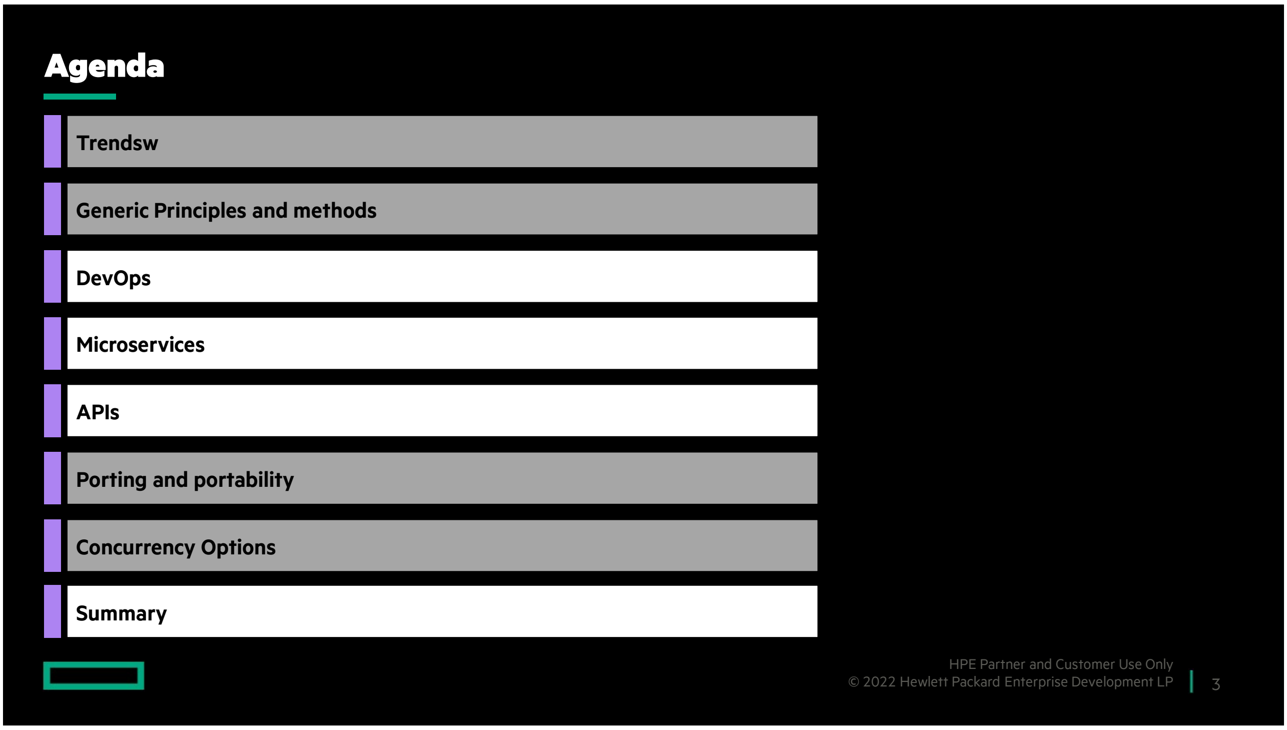
FORWARD-LOOKING STATEMENTS

This is a rolling (up to three year) Roadmap and is subject to change without notice

This document contains forward looking statements regarding future operations, product development, product capabilities and availability dates. This information is subject to substantial uncertainties and is subject to change at any time without prior notification. Statements contained in this document concerning these matters only reflect Hewlett Packard Enterprise's predictions and / or expectations as of the date of this document and actual results and future plans of Hewlett Packard Enterprise may differ significantly as a result of, among other things, changes in product strategy resulting from technological, internal corporate, market and other changes. This is not a commitment to deliver any material, code or functionality and should not be relied upon in making purchasing decisions.

| 2

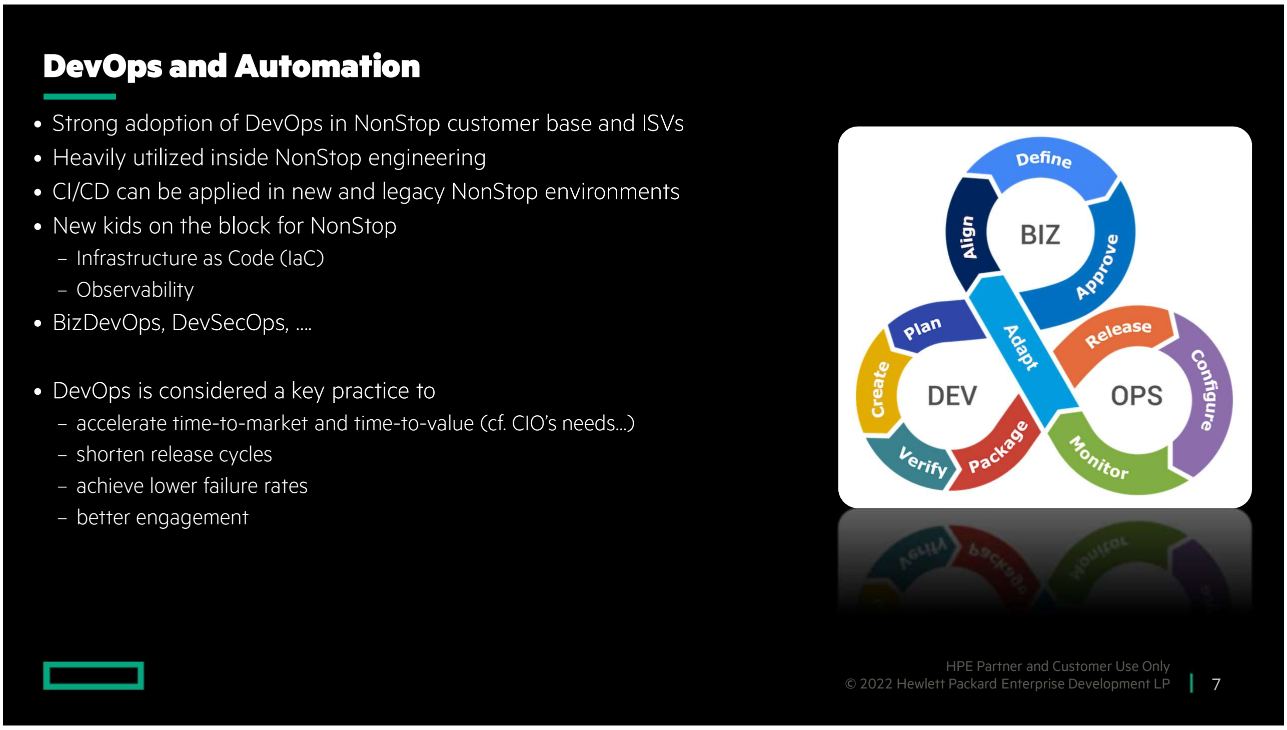
2



Agenda

- Trendsw
- Generic Principles and methods
- DevOps
- Microservices
- APIs
- Porting and portability
- Concurrency Options
- Summary

DevOps

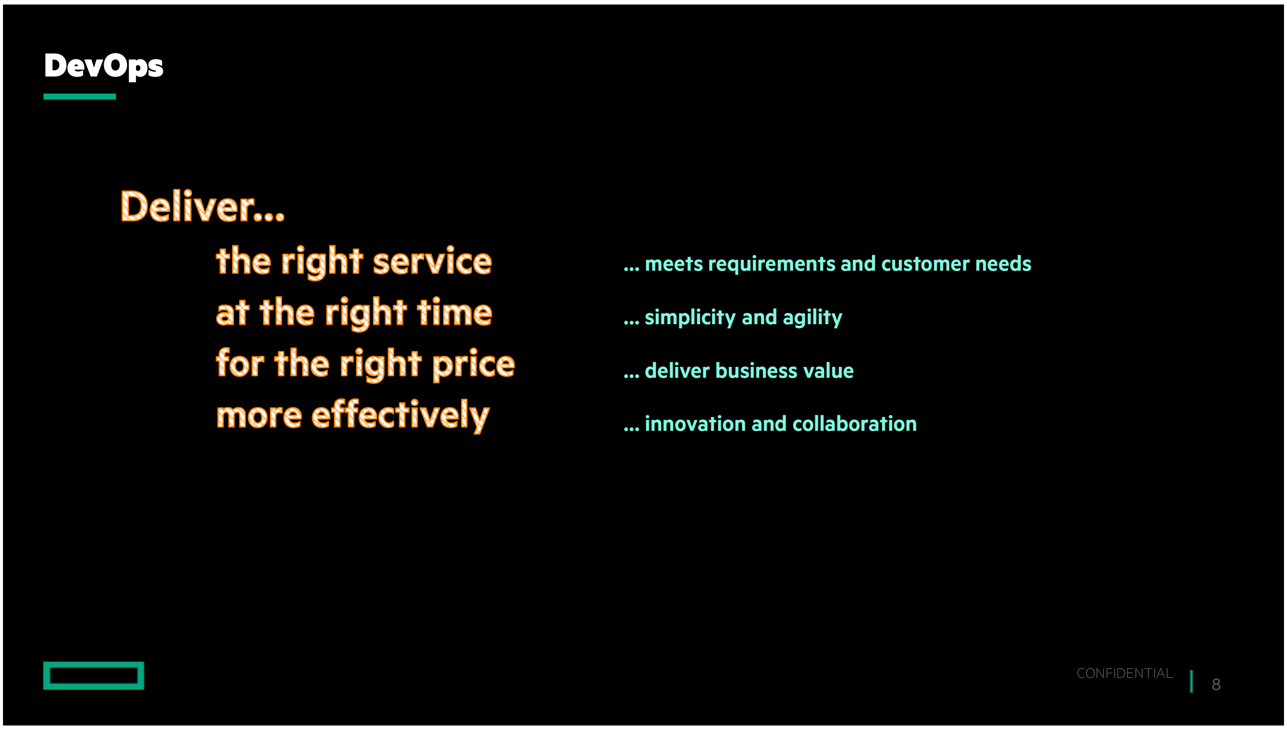


DevOps and Automation

- Strong adoption of DevOps in NonStop customer base and ISVs
- Heavily utilized inside NonStop engineering
- CI/CD can be applied in new and legacy NonStop environments
- New kids on the block for NonStop
  - Infrastructure as Code (IaC)
  - Observability
- BizDevOps, DevSecOps, ...

- DevOps is considered a key practice to
  - accelerate time-to-market and time-to-value (cf. CIO's needs...)
  - shorten release cycles
  - achieve lower failure rates
  - better engagement

HPE Partner and Customer Use Only



DevOps

Deliver...

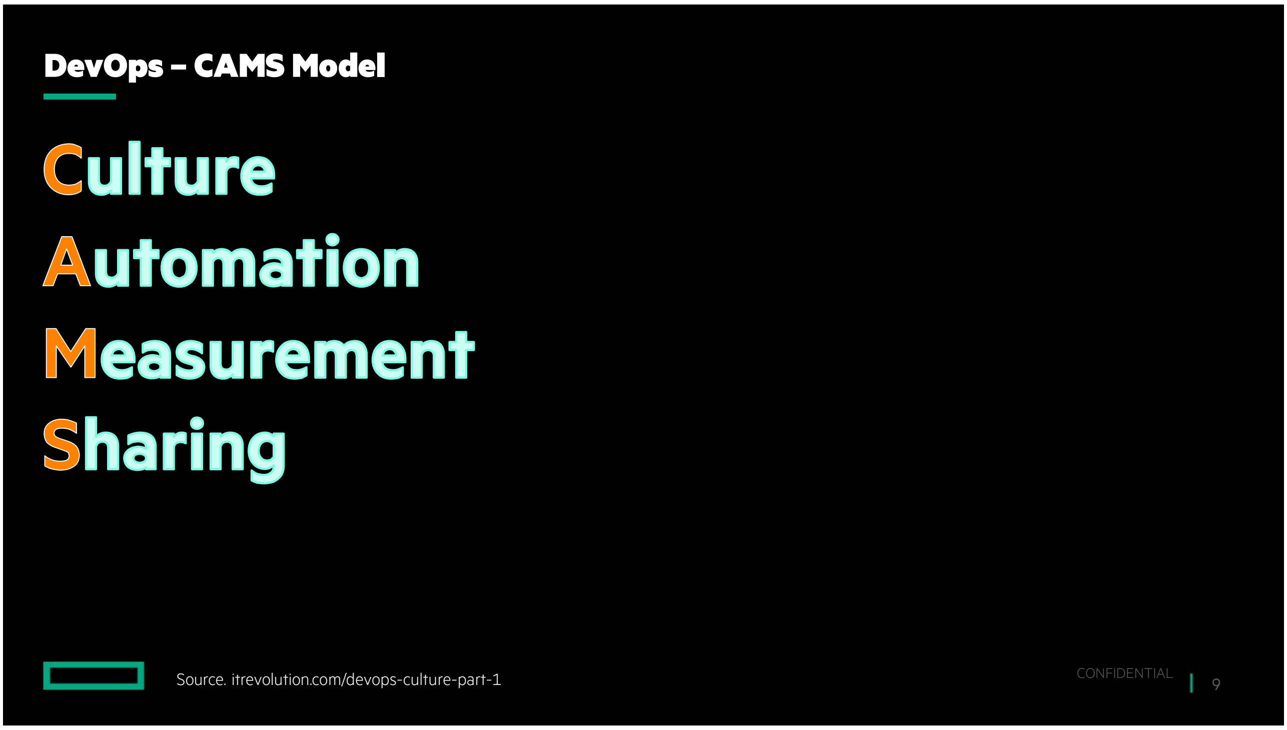
the right service ... meets requirements and customer needs

at the right time ... simplicity and agility

for the right price ... deliver business value

more effectively ... innovation and collaboration

CONFIDENTIAL



DevOps – CAMS Model

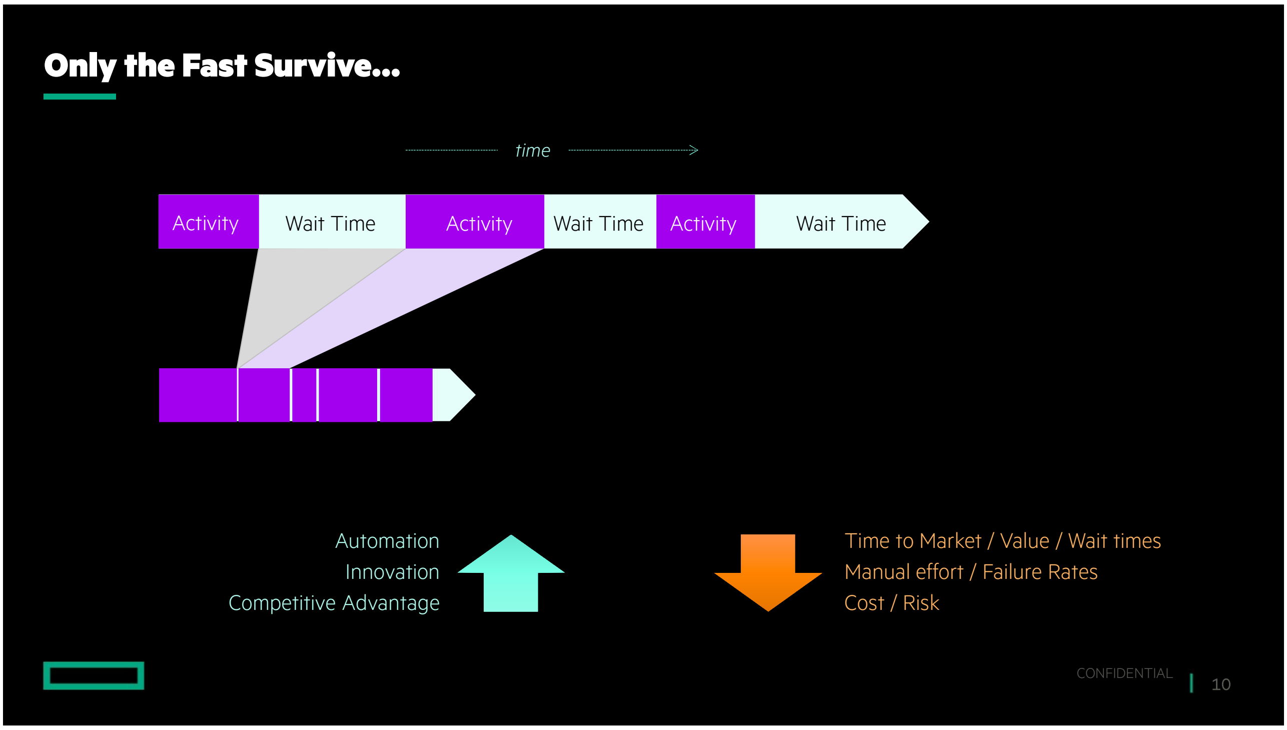
Culture

Automation

Measurement

Sharing

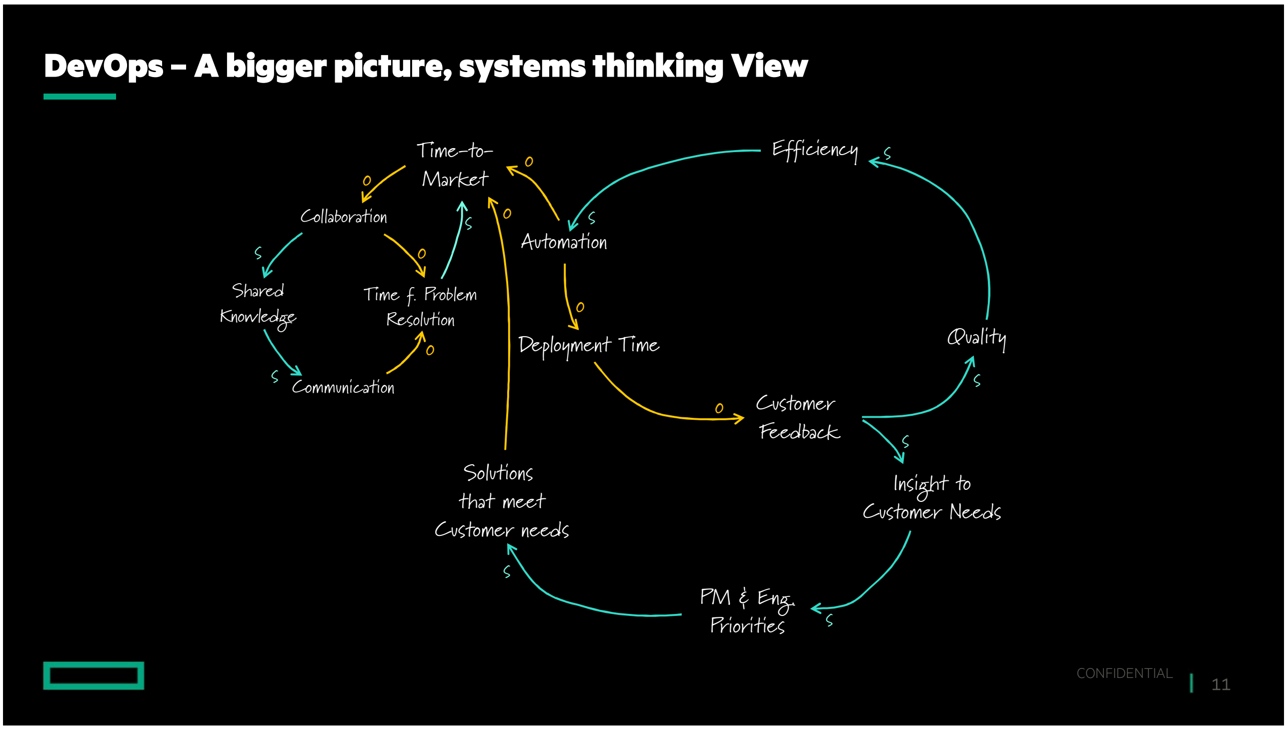
Source. itrevolution.com/devops-culture-part-1



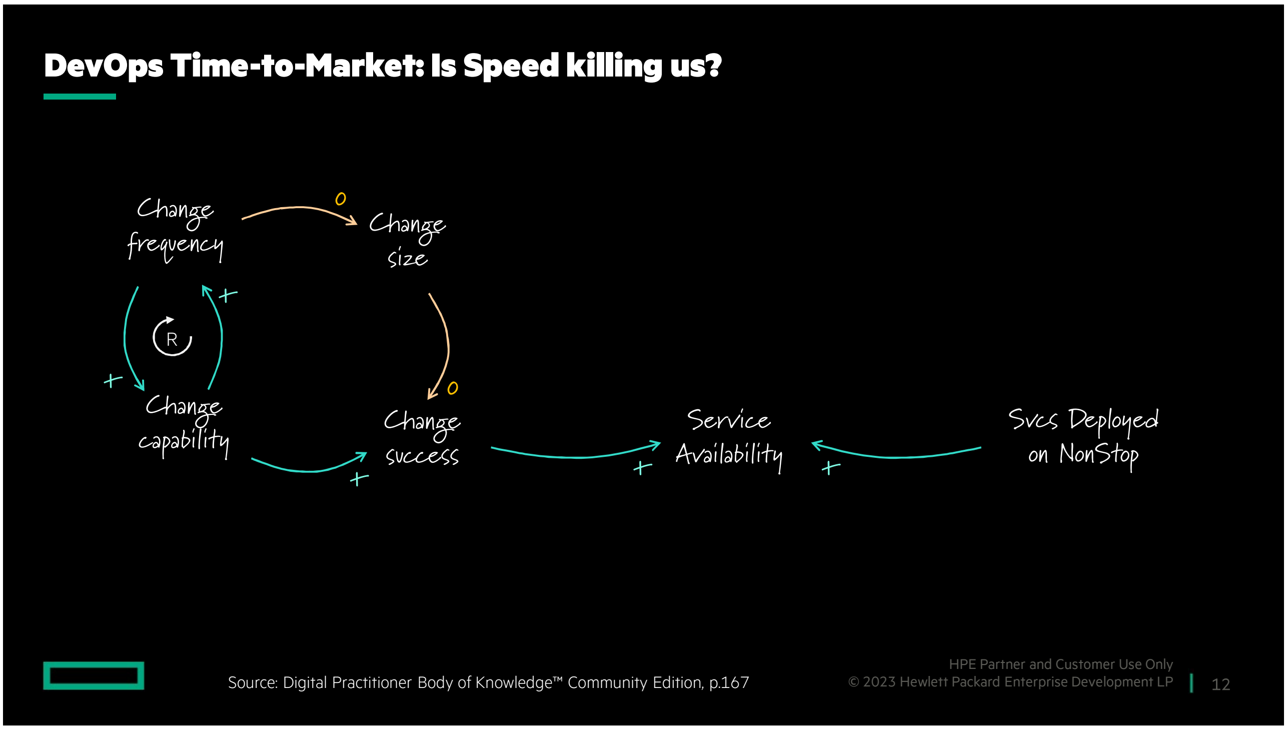
Only the Fast Survive…

Automation
Innovation
Competitive Advantage

Time to Market / Value / Wait times
Manual effort / Failure Rates
Cost / Risk



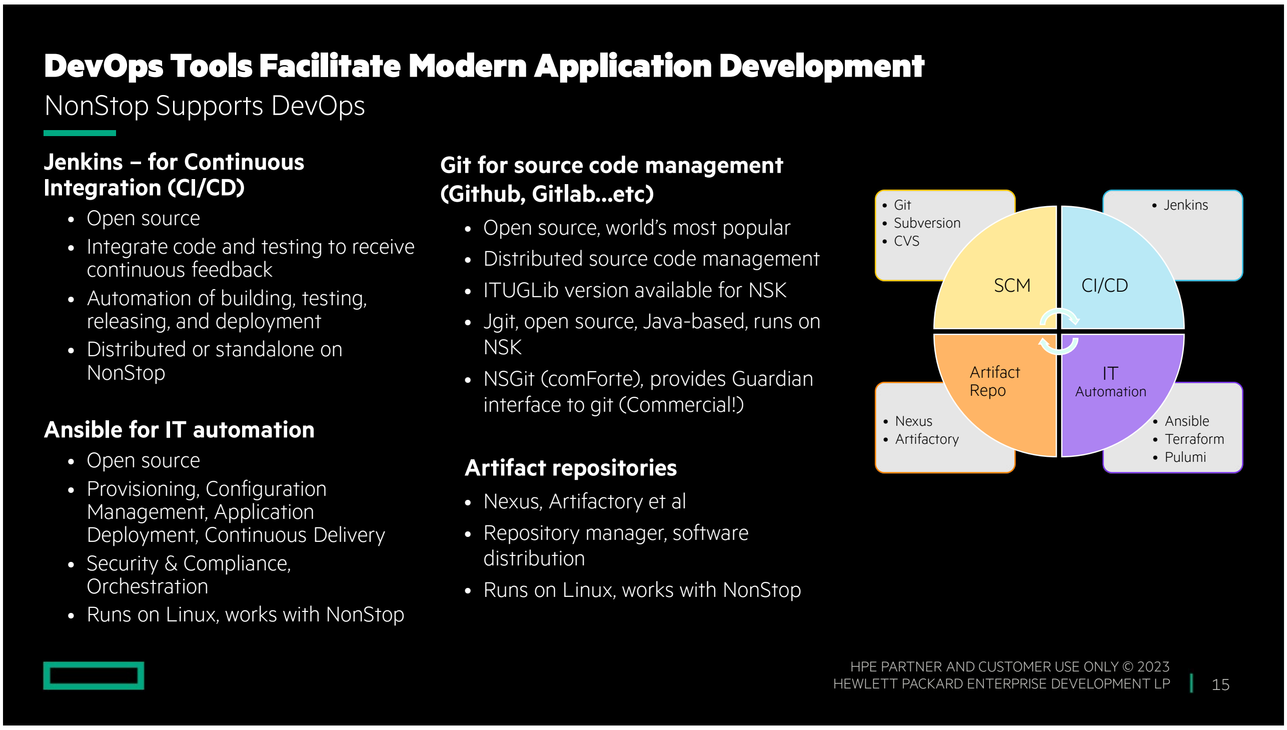
DevOps – A bigger picture, systems thinking View



DevOps Time-to-Market: Is Speed killing us?

Source: Digital Practitioner Body of Knowledge™ Community Edition, p.167

HPE Partner and Customer Use Only



DevOps Tools Facilitate Modern Application Development

NonStop Supports DevOps

Jenkins – for Continuous Integration (CI/CD)

- Open source
- Integrate code and testing to receive continuous feedback
- Automation of building, testing, releasing, and deployment
- Distributed or standalone on NonStop

Ansible for IT automation

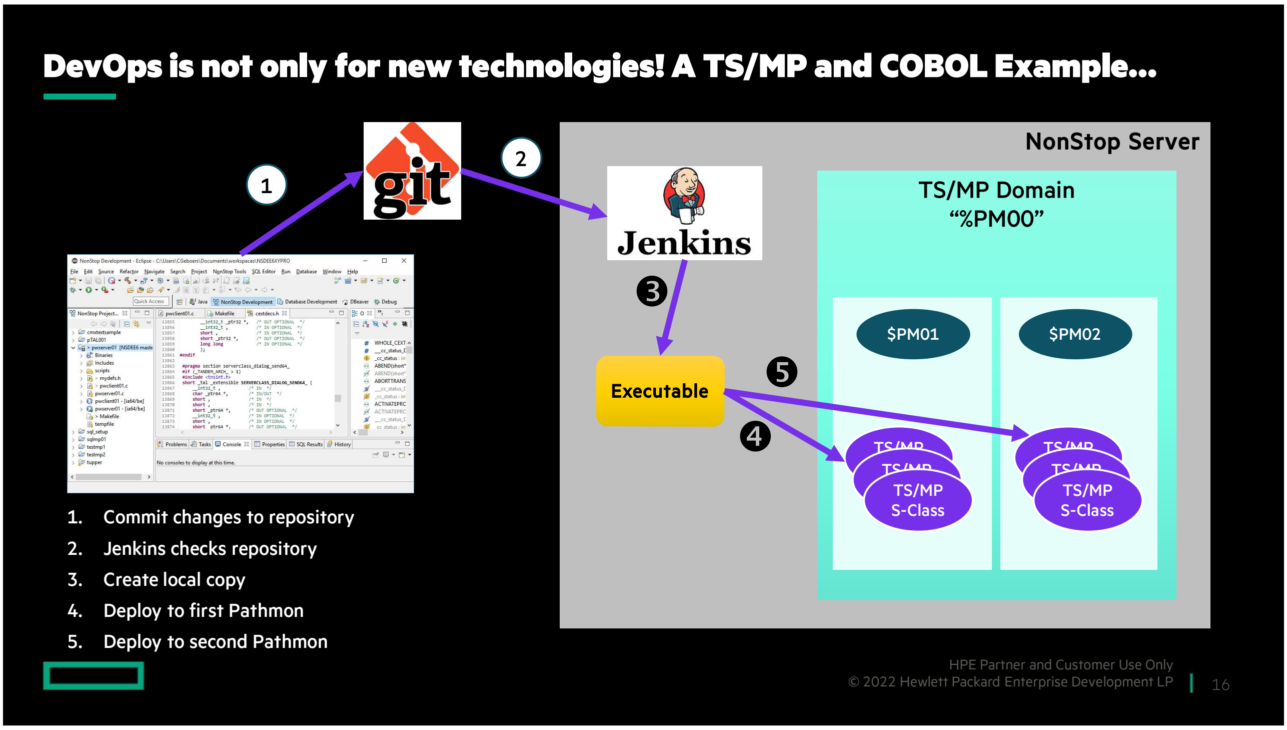
- Open source
- Provisioning, Configuration Management, Application Deployment, Continuous Delivery
- Security & Compliance, Orchestration
- Runs on Linux, works with NonStop

Git for source code management (Github, Gitlab…etc)

- Open source, world's most popular
- Distributed source code management
- ITUGLib version available for NSK
- Jgit, open source, Java-based, runs on NSK
- NSGit (comForte), provides Guardian interface to git (Commercial!)

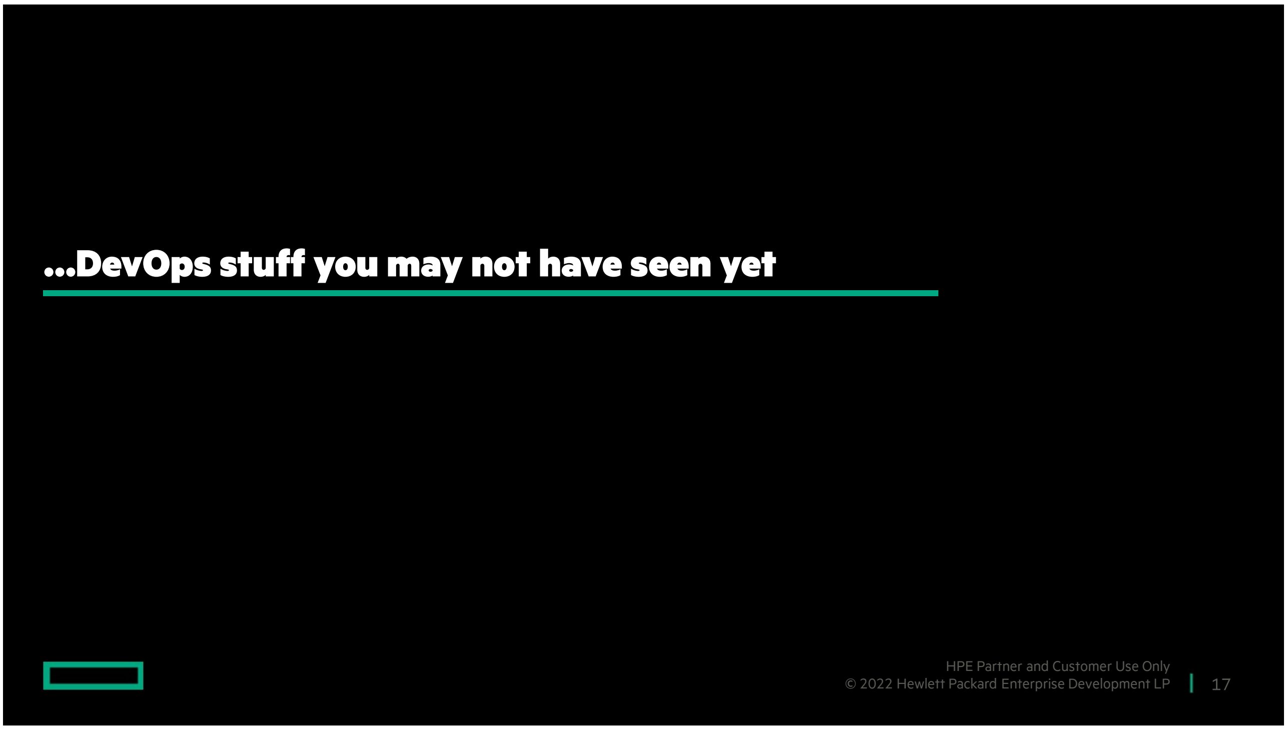
Artifact repositories

- Nexus, Artifactory et al
- Repository manager, software distribution
- Runs on Linux, works with NonStop

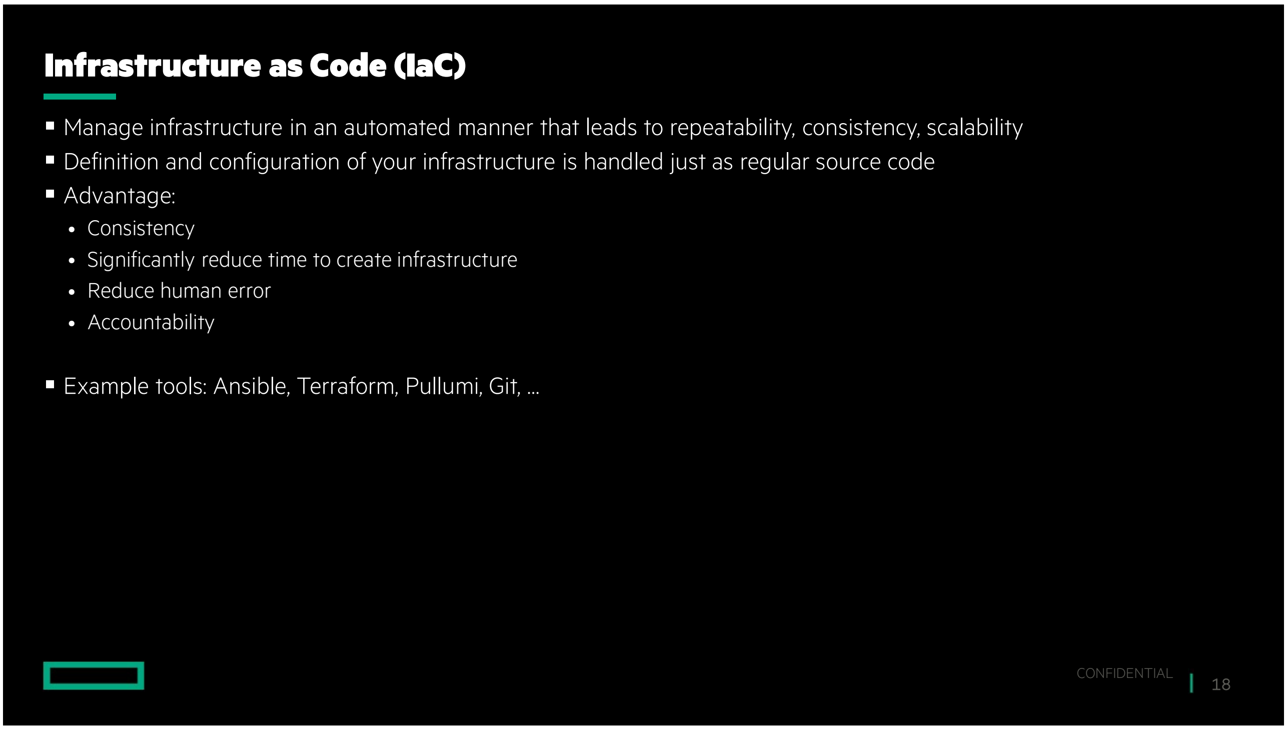


DevOps is not only for new technologies! A TS/MP and COBOL Example…

1. Commit changes to repository
2. Jenkins checks repository
3. Create local copy
4. Deploy to first Pathmon
5. Deploy to second Pathmon



…DevOps stuff you may not have seen yet



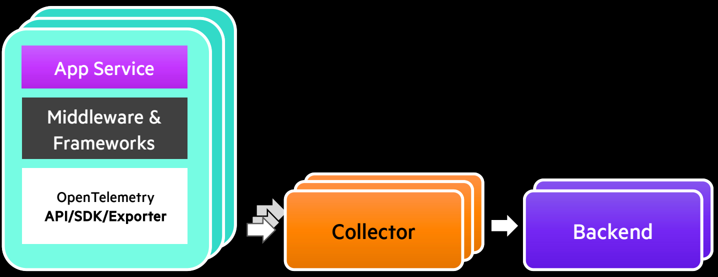
Infrastructure as Code (IaC)

- Manage infrastructure in an automated manner that leads to repeatability, consistency, scalability
- Definition and configuration of your infrastructure is handled just as regular source code
- Advantage:
  - Consistency
  - Significantly reduce time to create infrastructure
  - Reduce human error
  - Accountability

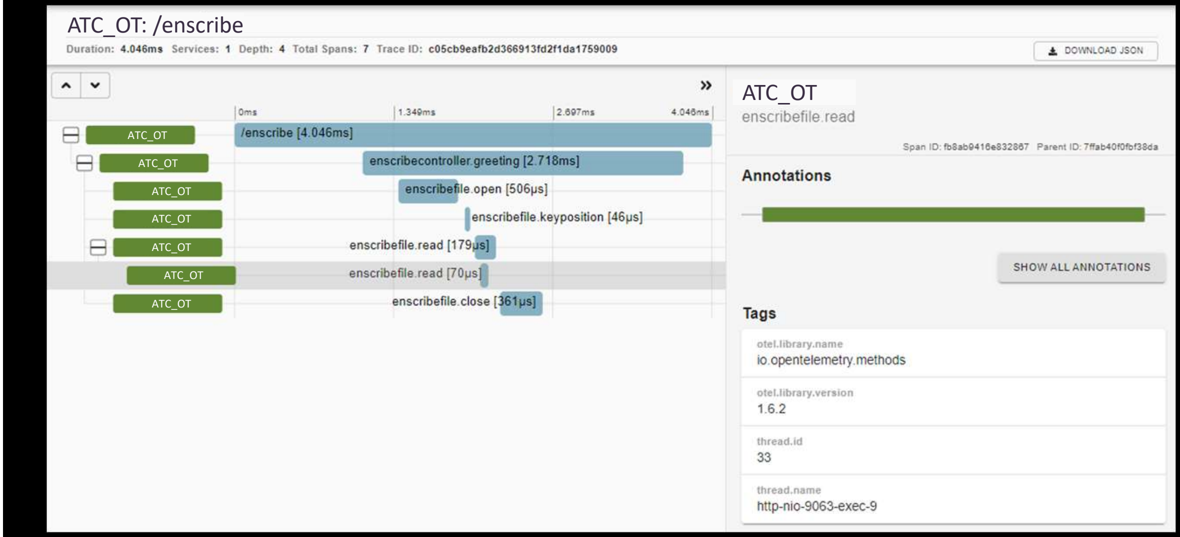
- Example tools: Ansible, Terraform, Pullumi, Git, …

Observability – a powerful Devops topic

- Today's business use cases oftentimes consist of many distributed services spread over a heterogeneous IT environment
- Problem and pain point: smaller grained, highly distributed services with lots of interactions can make it very hard to maintain a good overview of the system's status – and extremely difficult to spot issues in a timely manner
- Focus on allowing to reconstruct a systems state through external observation
- Emerging from Application Platform Monitoring
- Three pillars of distributed services:
  - Tracing (Concept of Spans)
  - Metrics (e.g. Rate, Errors, Duration,…)
  - Logging
- Lots of Open Source:
  - OpenTelemetry!
  - …and many more



Example: OpenTelemetry Measuring JEnscribe – Visualized in ZIPKIN


ATC_OT: /enscribe

Duration: **4.046ms** Services: **1** Depth: **4** Total Spans: **7** Trace ID: **c05cb9eafb2d366913fd2f1da1759009**

/enscribe [4.046ms]

enscribecontroller.greeting [2.718ms]

enscribefile.open [506µs]

enscribefile.keyposition [46µs]

enscribefile.read [179µs]

enscribefile.read [70µs]

enscribefile.close [361µs]

ATC_OT

enscribefile.read

Span ID: fb8ab9416e832867 Parent ID: 7ffab40f0fbf38da

Annotations

Tags

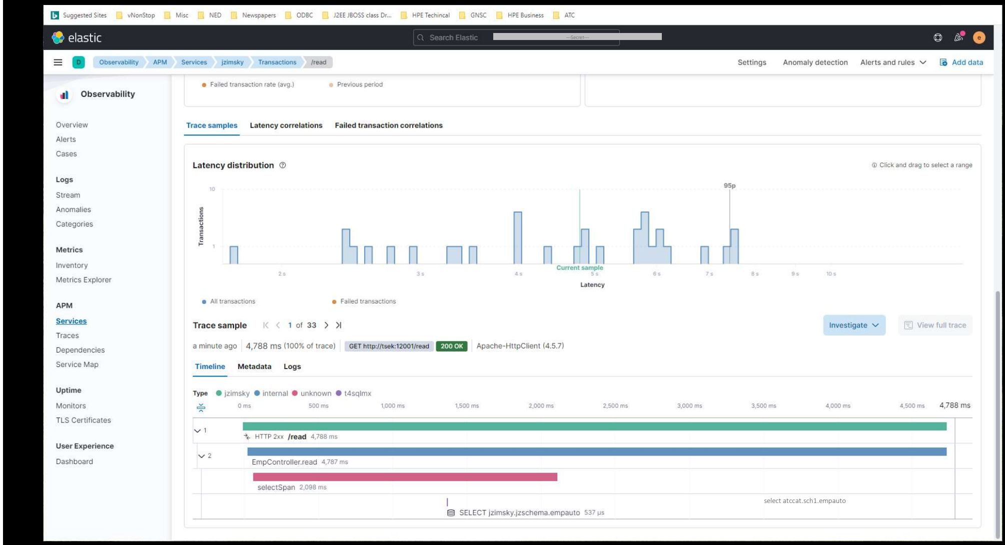
otel.library.name
io.opentelemetry.methods

otel.library.version
1.6.2

thread.id
33

thread.name
http-nio-9063-exec-9

Example 2: OpenTelemetry with ELK Stack Backend



Trace sample 1 of 33

a minute ago | 4,788 ms (100% of trace) | GET http://tsek:12001/read | 200 OK | Apache-HttpClient (4.5.7)

Type ● jzimsky ● internal ● unknown ● t4sqlmx

1 HTTP 2xx /read 4,788 ms

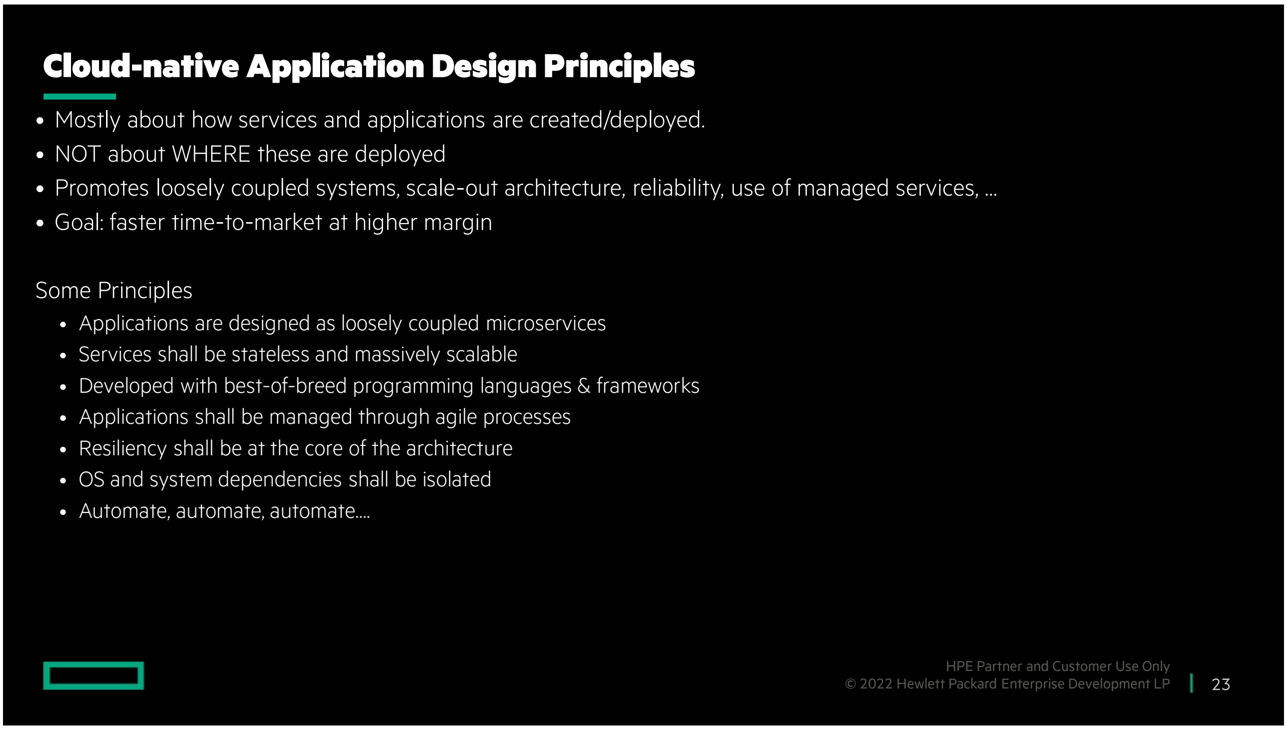
2 EmpController.read 4,787 ms

selectSpan 2,098 ms

SELECT jzimsky.jzschema.empauto 537 μs

select atccat.sch1.empauto

MICROSERVICES



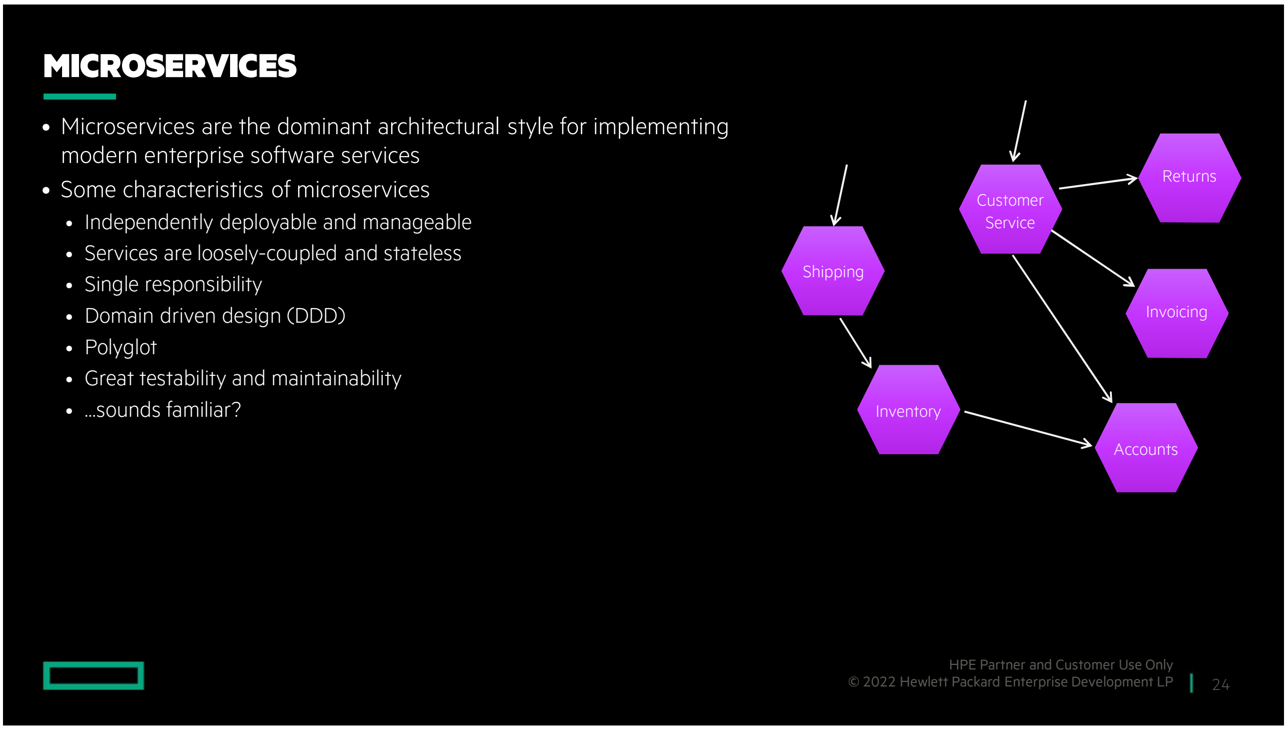
Cloud-native Application Design Principles

- Mostly about how services and applications are created/deployed.
- NOT about WHERE these are deployed
- Promotes loosely coupled systems, scale-out architecture, reliability, use of managed services, …
- Goal: faster time-to-market at higher margin

Some Principles

- Applications are designed as loosely coupled microservices
- Services shall be stateless and massively scalable
- Developed with best-of-breed programming languages & frameworks
- Applications shall be managed through agile processes
- Resiliency shall be at the core of the architecture
- OS and system dependencies shall be isolated
- Automate, automate, automate….

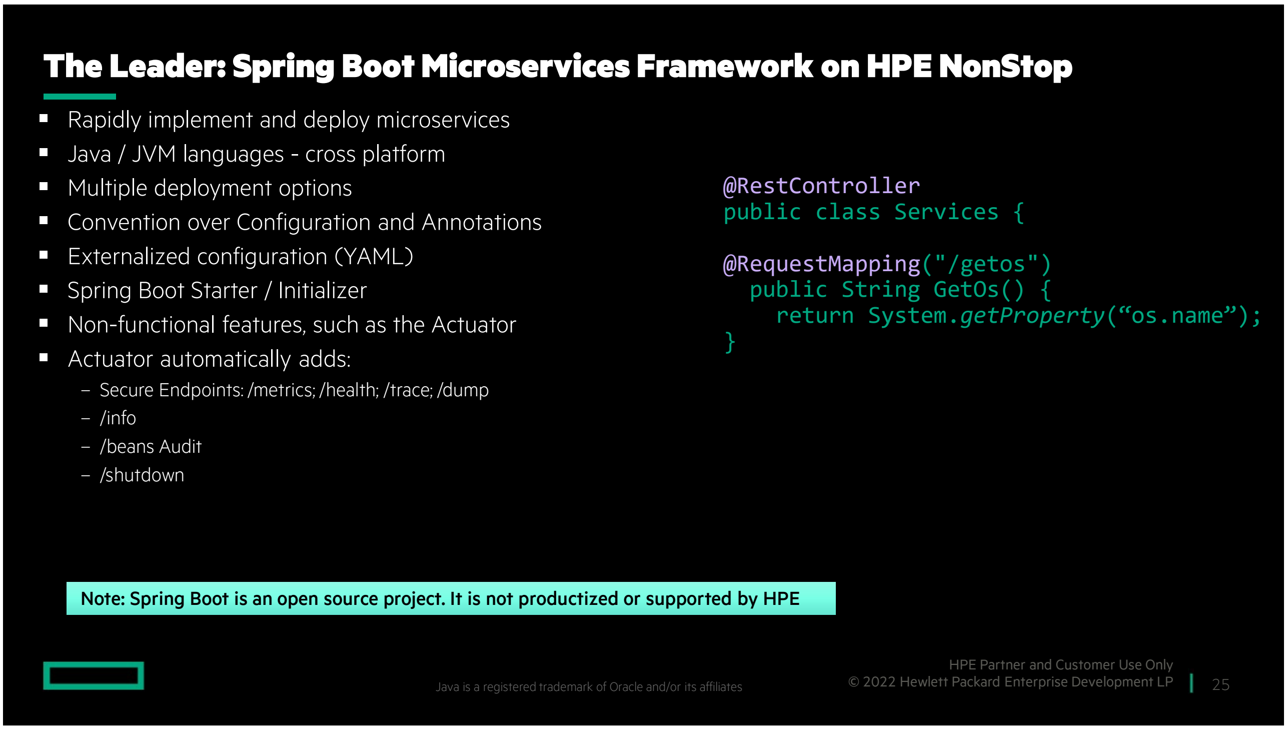
HPE Partner and Customer Use Only



MICROSERVICES

- Microservices are the dominant architectural style for implementing modern enterprise software services
- Some characteristics of microservices
  - Independently deployable and manageable
  - Services are loosely-coupled and stateless
  - Single responsibility
  - Domain driven design (DDD)
  - Polyglot
  - Great testability and maintainability
  - …sounds familiar?

HPE Partner and Customer Use Only



The Leader: Spring Boot Microservices Framework on HPE NonStop

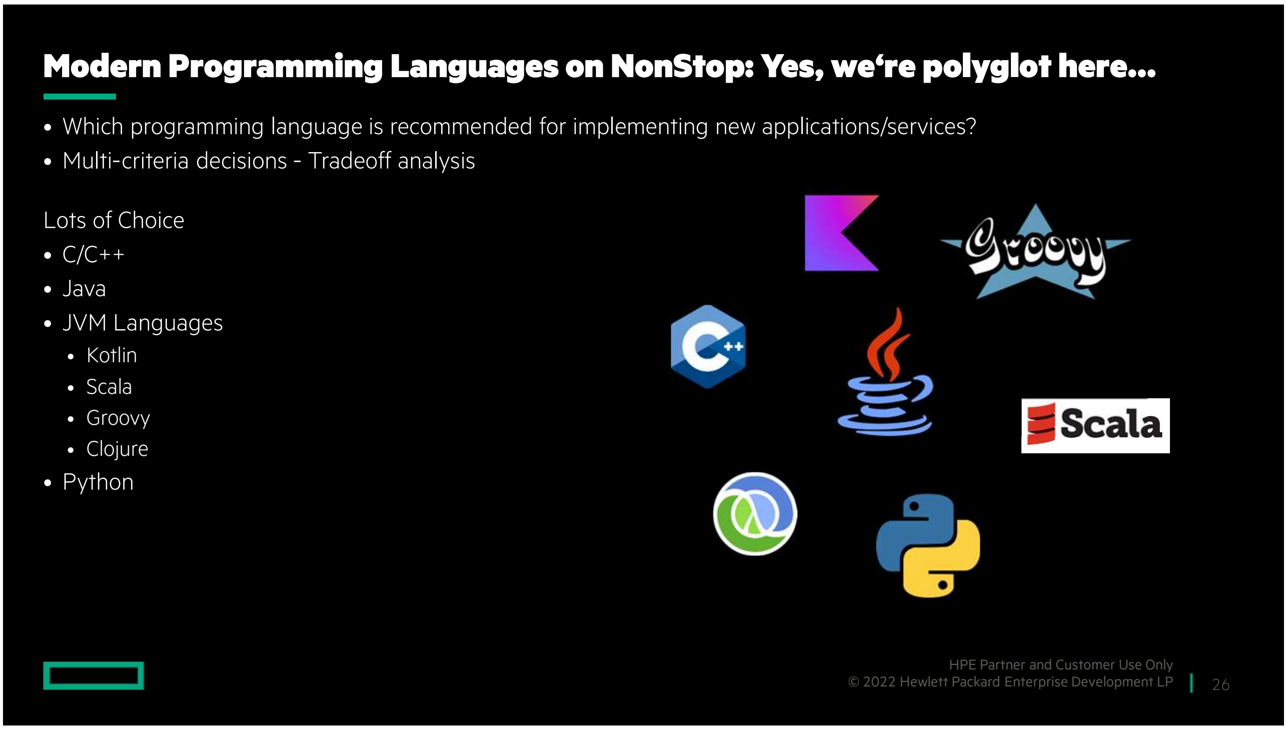
- Rapidly implement and deploy microservices
- Java / JVM languages - cross platform
- Multiple deployment options
- Convention over Configuration and Annotations
- Externalized configuration (YAML)
- Spring Boot Starter / Initializer
- Non-functional features, such as the Actuator
- Actuator automatically adds:
  - Secure Endpoints: /metrics; /health; /trace; /dump
  - /info
  - /beans Audit
  - /shutdown

```
@RestController
public class Services {

@RequestMapping("/getos")
  public String GetOs() {
    return System.getProperty(“os.name”);
}
```

Note: Spring Boot is an open source project. It is not productized or supported by HPE

Java is a registered trademark of Oracle and/or its affiliates

HPE Partner and Customer Use Only



Modern Programming Languages on NonStop: Yes, we‘re polyglot here…

- Which programming language is recommended for implementing new applications/services?
- Multi-criteria decisions - Tradeoff analysis

Lots of Choice

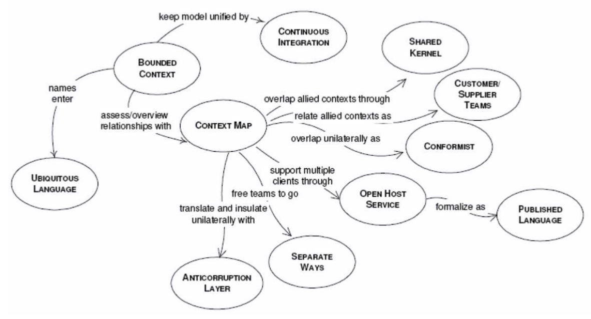
- C/C++
- Java
- JVM Languages
  - Kotlin
  - Scala
  - Groovy
  - Clojure
- Python

Groovy
Scala

HPE Partner and Customer Use Only

DOMAIN DRIVEN DESIGN (DDD) [1)]

- An approach to model and design complex software that shall match the actual business domain
- Introduced by Eric Evans in his seminal book[1)] in 2003
- Business / IT gap causes major difficulties
- Ubiquitous language – a common shared language defined by domain experts to describing artifacts
- Key concepts
  - Domain Model: Entities, Value Objects, Service Objects, Domain Events, Repositories, …
  - Architecture techniques: Evolving Structures/Order, Responsibilities, Extensibility, …
  - Design techniques: Intention revealing interfaces, standalone classes, anticorruption layer, …



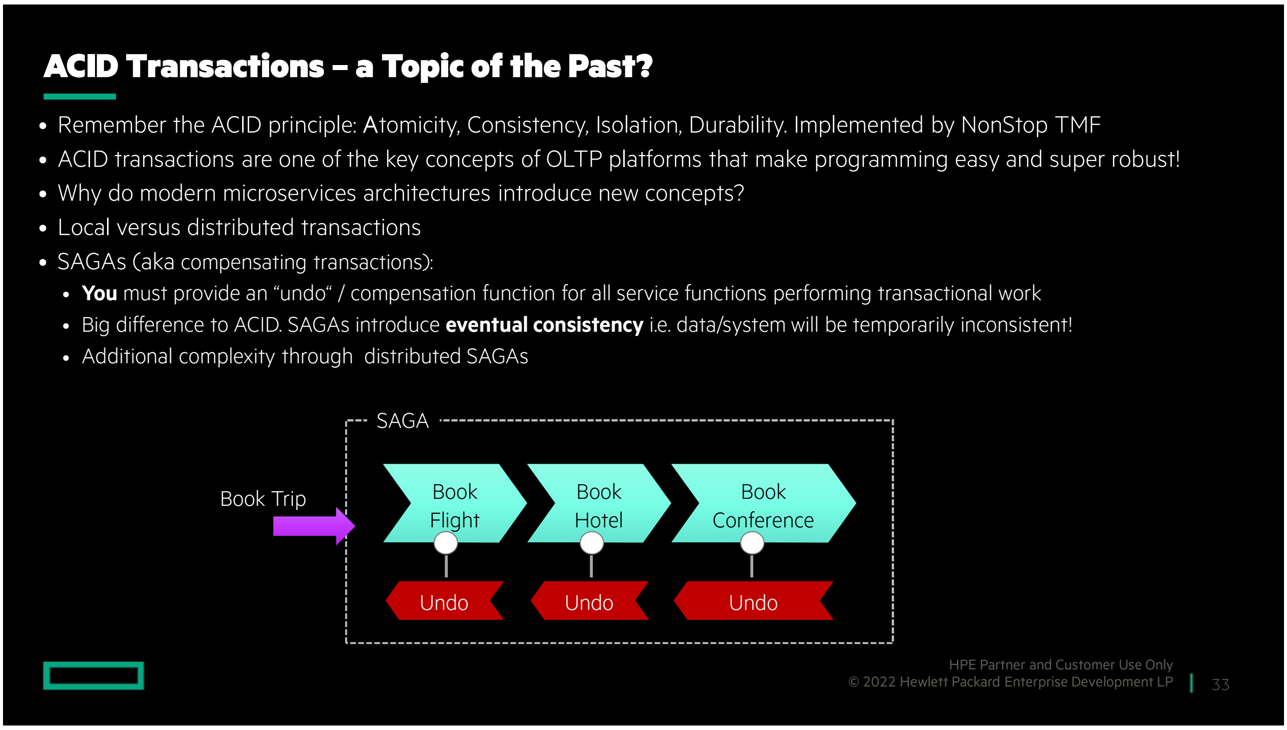
Source: Abel Avram: Domain-Driven Design Quickly p.28, 2006, 978-1-4116-0925-9
Free Online Version of Book on InfoQ.com

DDD is frequently applied in well engineered, large microservices architectures

1) Eric Evans: Domain-Driven Design. Tackling Complexity in the Heart of Software. Addison-Wesley, 2003, ISBN 978-0-321-12521-7

Microservices – Some Pros and Cons

Pros	Cons
On many platforms (other than NonStop) monoliths were the dominant architectural style. Microservices are an attempt to build more modular architectures that allow services to be maintained independently	Many developers implement microservices as mini-monoliths, because that's what they are used to. Getting the design right can be tricky for people new to the concept.
Database per service: data owned by a microservice is considered private, i.e. no direct access from other services	Lots of extra complexity and potential performance impact. Reducing the purpose of your database to simple tasks
Microservices can be transactional, but there is no concept of transactions across microservices! (…well, on NonStop you can use them if you want)	The developer is responsible to implement the work that is normally provided by transaction managers. Concepts such as compensating transactions are needed, which is not an easy task. Business data inconsistency risk. Eventual consistency challenge….



ACID Transactions – a Topic of the Past?

- Remember the ACID principle: Atomicity, Consistency, Isolation, Durability. Implemented by NonStop TMF
- ACID transactions are one of the key concepts of OLTP platforms that make programming easy and super robust!
- Why do modern microservices architectures introduce new concepts?
- Local versus distributed transactions
- SAGAs (aka compensating transactions):
  - **You** must provide an "undo" / compensation function for all service functions performing transactional work
  - Big difference to ACID. SAGAs introduce **eventual consistency** i.e. data/system will be temporarily inconsistent!
  - Additional complexity through distributed SAGAs

SAGA

Book Trip

Book Flight

Book Hotel

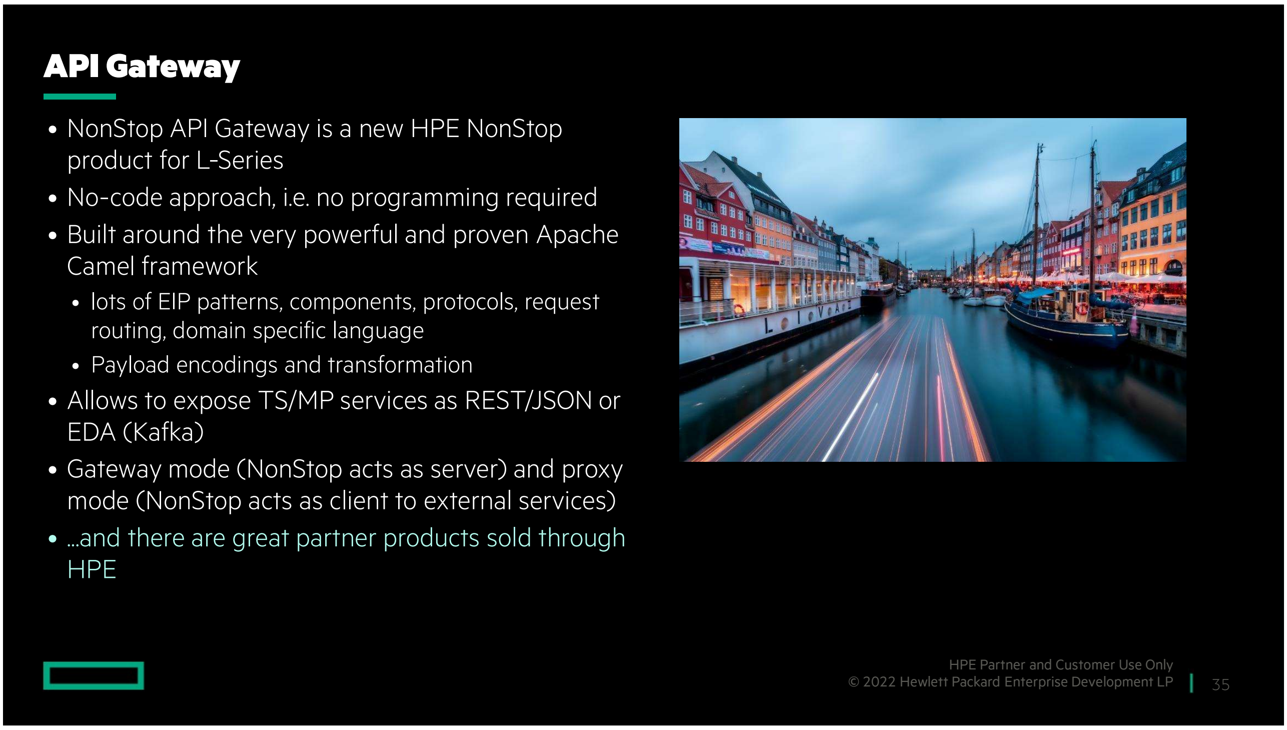
Book Conference

Undo

Undo

Undo

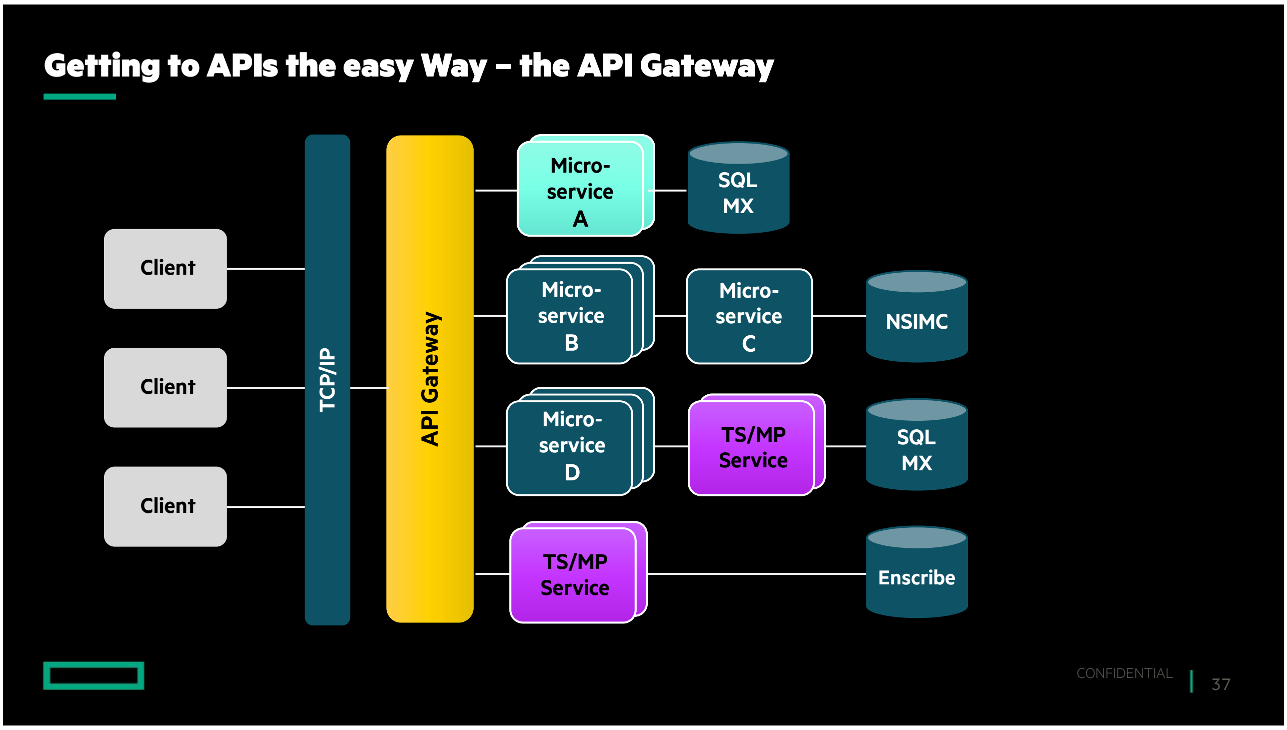
APIs



API Gateway

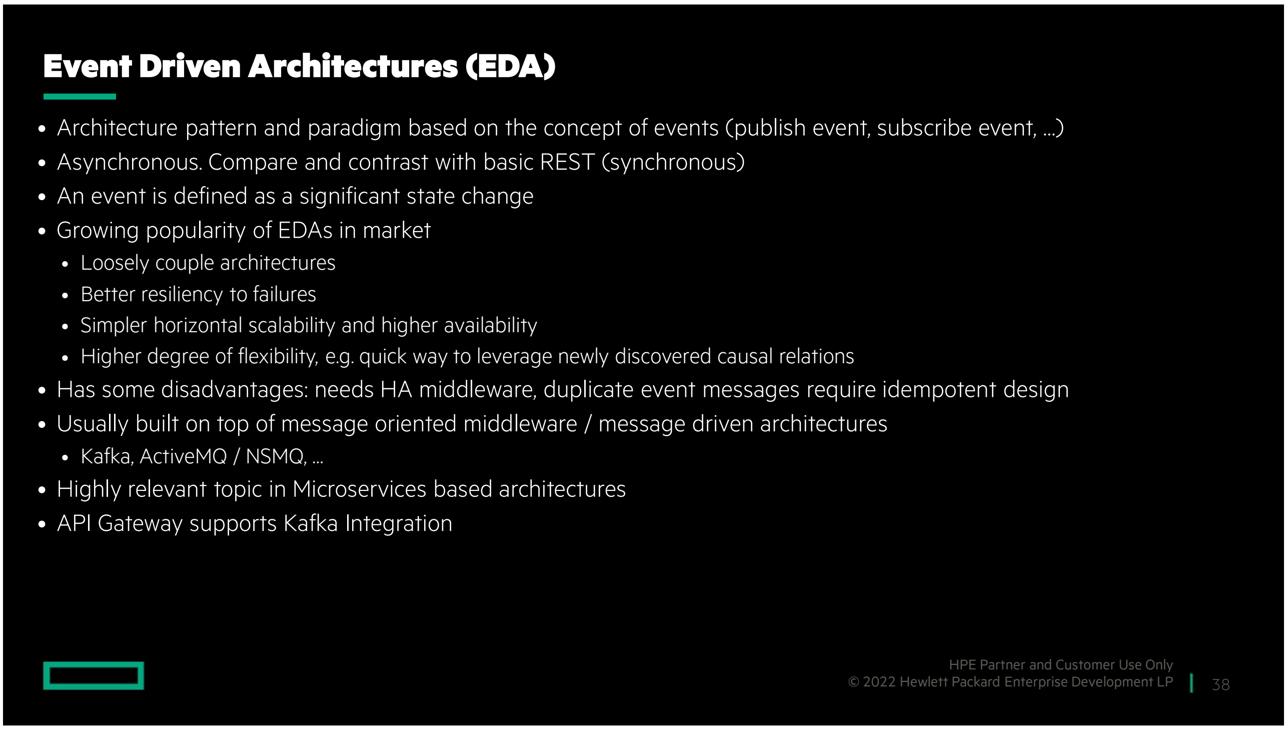
- NonStop API Gateway is a new HPE NonStop product for L-Series
- No-code approach, i.e. no programming required
- Built around the very powerful and proven Apache Camel framework
  - lots of EIP patterns, components, protocols, request routing, domain specific language
  - Payload encodings and transformation
- Allows to expose TS/MP services as REST/JSON or EDA (Kafka)
- Gateway mode (NonStop acts as server) and proxy mode (NonStop acts as client to external services)
- …and there are great partner products sold through HPE

HPE Partner and Customer Use Only



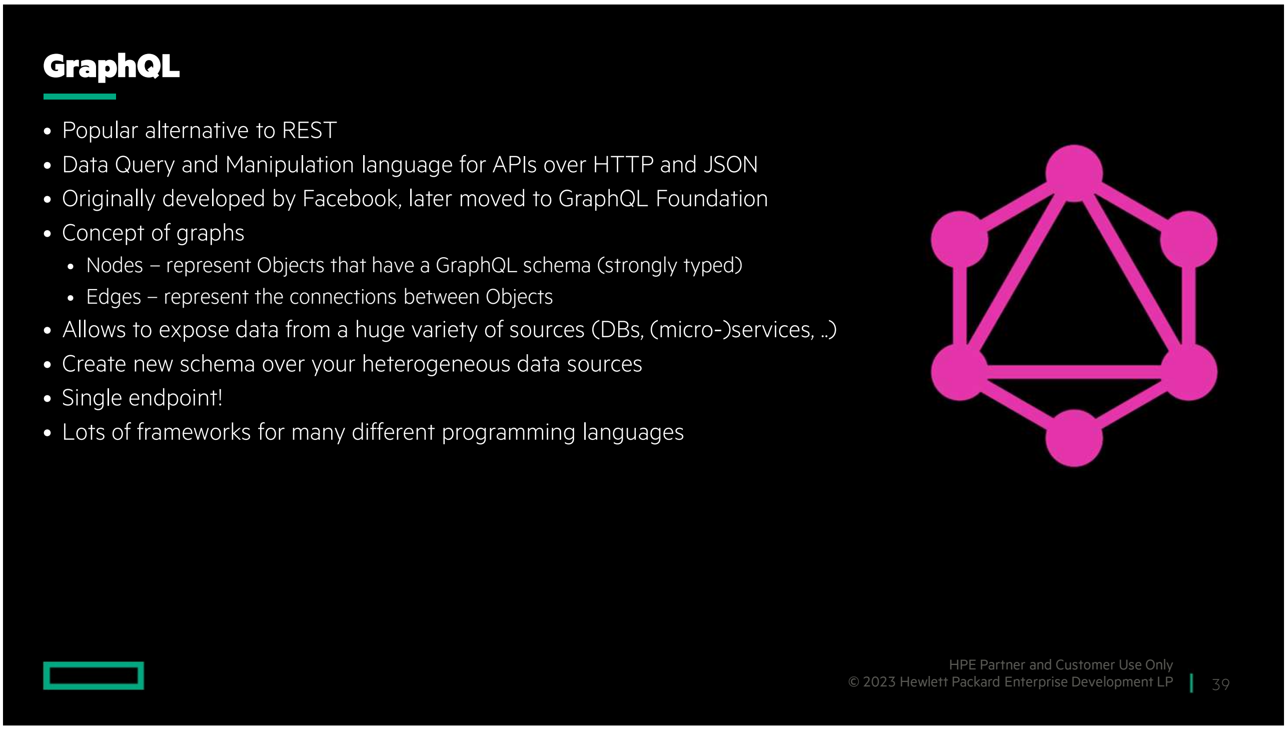
Getting to APIs the easy Way – the API Gateway

CONFIDENTIAL



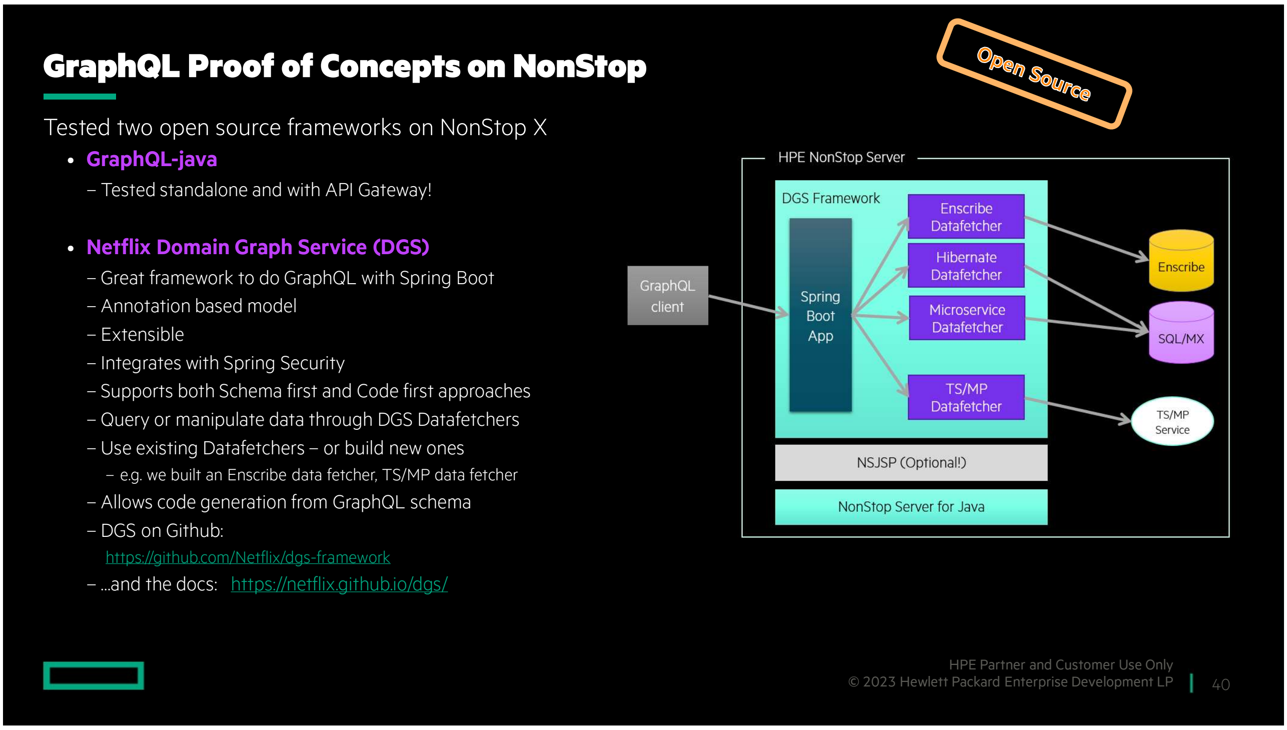
Event Driven Architectures (EDA)

- Architecture pattern and paradigm based on the concept of events (publish event, subscribe event, …)
- Asynchronous. Compare and contrast with basic REST (synchronous)
- An event is defined as a significant state change
- Growing popularity of EDAs in market
  - Loosely couple architectures
  - Better resiliency to failures
  - Simpler horizontal scalability and higher availability
  - Higher degree of flexibility, e.g. quick way to leverage newly discovered causal relations
- Has some disadvantages: needs HA middleware, duplicate event messages require idempotent design
- Usually built on top of message oriented middleware / message driven architectures
  - Kafka, ActiveMQ / NSMQ, …
- Highly relevant topic in Microservices based architectures
- API Gateway supports Kafka Integration



GraphQL

- Popular alternative to REST
- Data Query and Manipulation language for APIs over HTTP and JSON
- Originally developed by Facebook, later moved to GraphQL Foundation
- Concept of graphs
  - Nodes – represent Objects that have a GraphQL schema (strongly typed)
  - Edges – represent the connections between Objects
- Allows to expose data from a huge variety of sources (DBs, (micro-)services, ..)
- Create new schema over your heterogeneous data sources
- Single endpoint!
- Lots of frameworks for many different programming languages

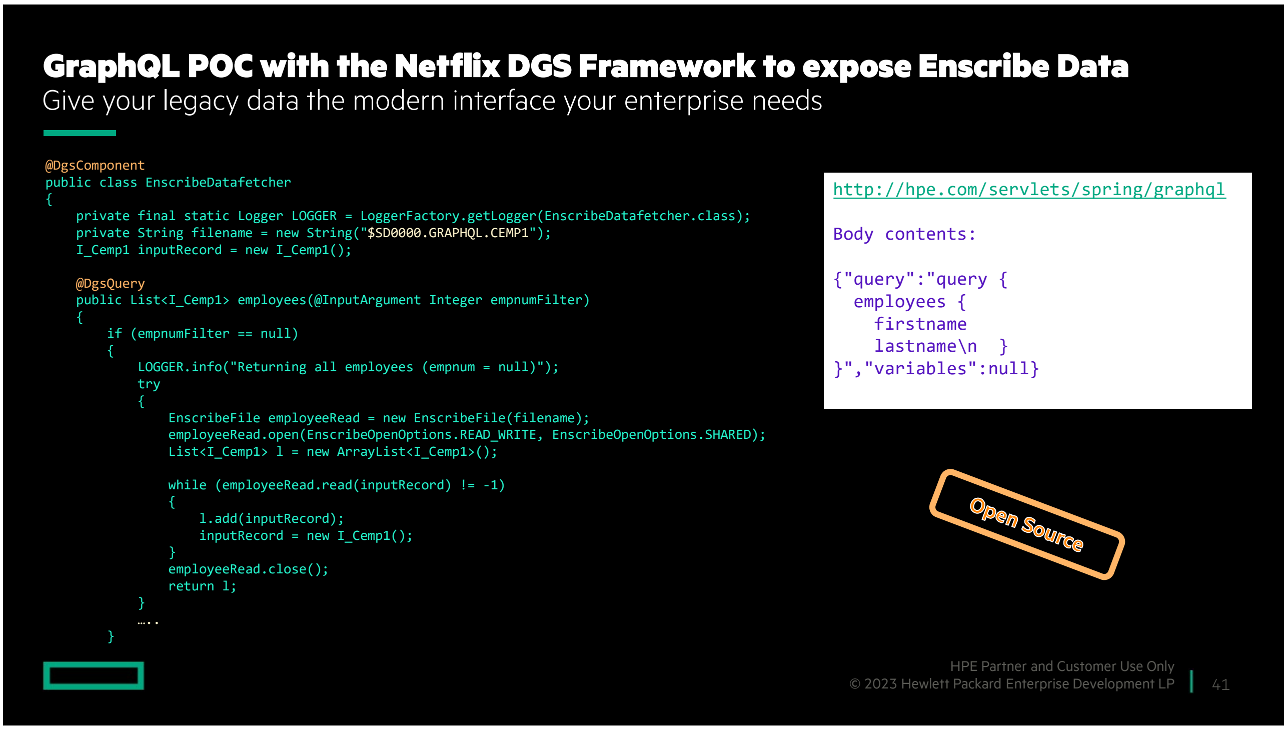


GraphQL Proof of Concepts on NonStop

Open Source

Tested two open source frameworks on NonStop X

- **GraphQL-java**
  - Tested standalone and with API Gateway!
- **Netflix Domain Graph Service (DGS)**
  - Great framework to do GraphQL with Spring Boot
  - Annotation based model
  - Extensible
  - Integrates with Spring Security
  - Supports both Schema first and Code first approaches
  - Query or manipulate data through DGS Datafetchers
  - Use existing Datafetchers – or build new ones
    - e.g. we built an Enscribe data fetcher, TS/MP data fetcher
  - Allows code generation from GraphQL schema
  - DGS on Github:
    https://github.com/Netflix/dgs-framework
  - …and the docs: https://netflix.github.io/dgs/



GraphQL POC with the Netflix DGS Framework to expose Enscribe Data

Give your legacy data the modern interface your enterprise needs

```
@DgsComponent
public class EnscribeDatafetcher
{
    private final static Logger LOGGER = LoggerFactory.getLogger(EnscribeDatafetcher.class);
    private String filename = new String("$SD0000.GRAPHQL.CEMP1");
    I_Cemp1 inputRecord = new I_Cemp1();

    @DgsQuery
    public List<I_Cemp1> employees(@InputArgument Integer empnumFilter)
    {
        if (empnumFilter == null)
        {
            LOGGER.info("Returning all employees (empnum = null)");
            try
            {
                EnscribeFile employeeRead = new EnscribeFile(filename);
                employeeRead.open(EnscribeOpenOptions.READ_WRITE, EnscribeOpenOptions.SHARED);
                List<I_Cemp1> l = new ArrayList<I_Cemp1>();

                while (employeeRead.read(inputRecord) != -1)
                {
                    l.add(inputRecord);
                    inputRecord = new I_Cemp1();
                }
                employeeRead.close();
                return l;
            }
            …..
        }
```

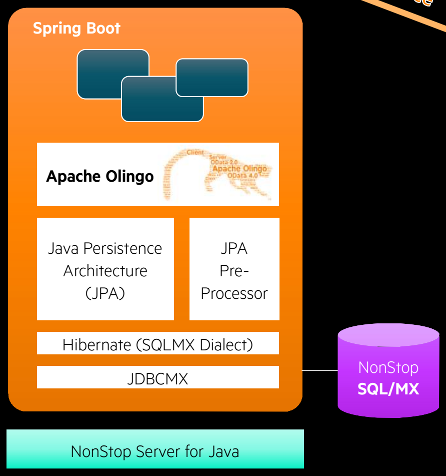
```
http://hpe.com/servlets/spring/graphql

Body contents:

{"query":"query {
  employees {
    firstname
    lastname\n  }
}","variables":null}
```

Open Source

OData

Open Source

- Open Data Protocol (OData) Protocol
- Allows consumption and creation of queryable and interoperable REST APIs
- simple, standardized access
- Standardized – original proposal by Microsoft.
  - OData V4 standardized at OASIS.
  - ISO/IEC published OData 4.0 Core as ISO/IEC 20802-1:2016 and the OData JSON Format as ISO/IEC 20802-2:2016-2:2016
- Quite popular in the industry – vendor backing by Microsoft, Salesforce, SAP, IBM, Tibco, Tableau, Oracle, Mulesoft, …
- Lots of libraries available for many languages, such as Java, Python, Ruby, JavaScript, PHP, ….



OData - Example

Open Source

Entity Object Example:

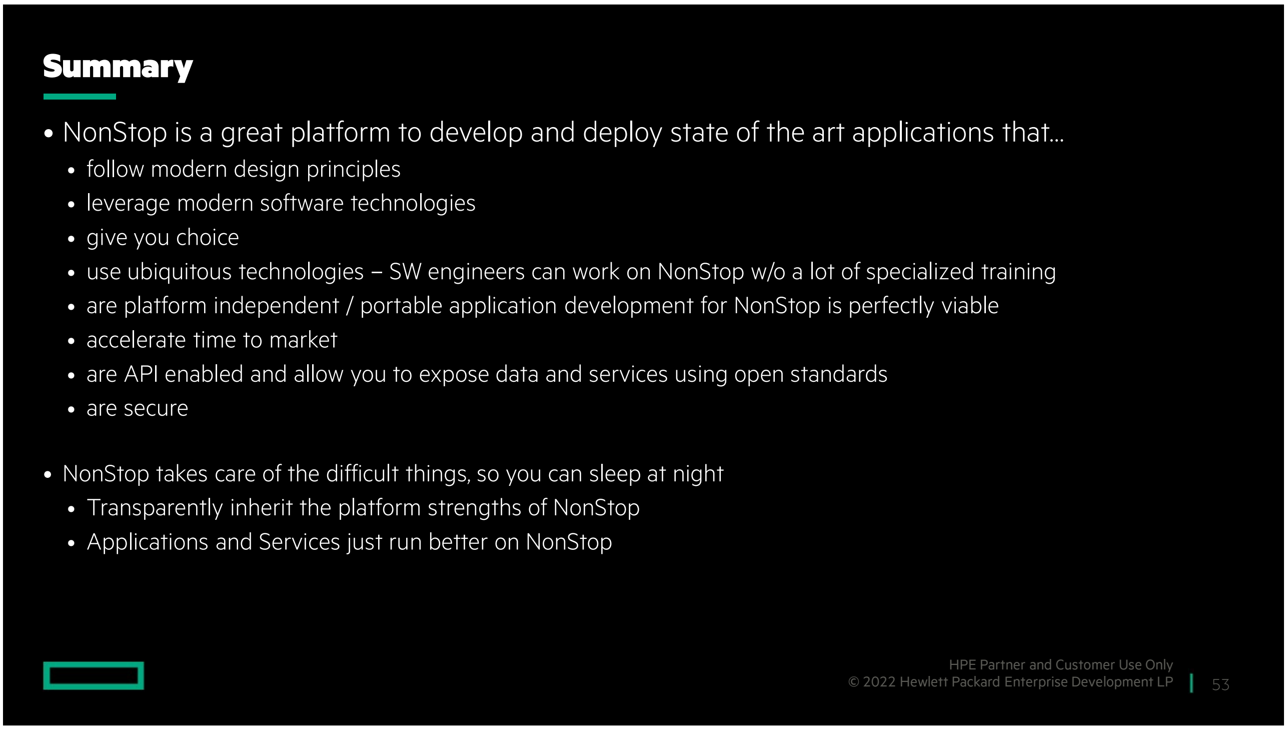
```
@Entity (name = "Employee")
public class Employee
{
    @Column (name = "empid")
    private int empid;

    @Column (name = "firstname")
    private String firstName;

    @Column (name = "lastname")
    private String lastName;

    @OneToOne (cascade=CascadeType.ALL,fetch=FetchType.LAZY)
    @JoinColumn(name="departid", unique= true, nullable=true,
                insertable=true, updatable=true)
    private Department department;
}
```

Request:

http://localhost:12001/jpa.svc/v1/Employees(1)?$format=json

OData Service Response in JSON:

```
{
  "@odata.context": "$metadata#jpa.svc.Employee",
  "Empid": 1,
  "FirstName": "Joe",
  "LastName": "Smith"
}
```

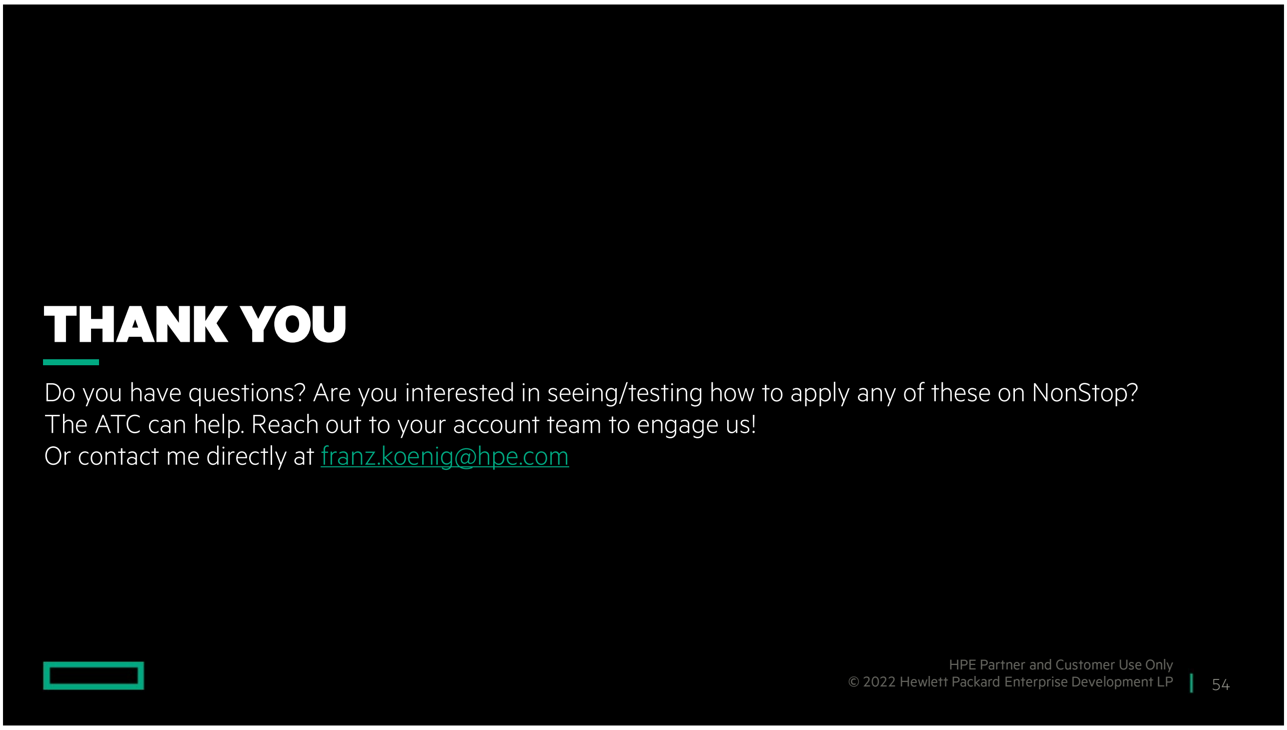


Summary

- NonStop is a great platform to develop and deploy state of the art applications that…
  - follow modern design principles
  - leverage modern software technologies
  - give you choice
  - use ubiquitous technologies – SW engineers can work on NonStop w/o a lot of specialized training
  - are platform independent / portable application development for NonStop is perfectly viable
  - accelerate time to market
  - are API enabled and allow you to expose data and services using open standards
  - are secure

- NonStop takes care of the difficult things, so you can sleep at night
  - Transparently inherit the platform strengths of NonStop
  - Applications and Services just run better on NonStop

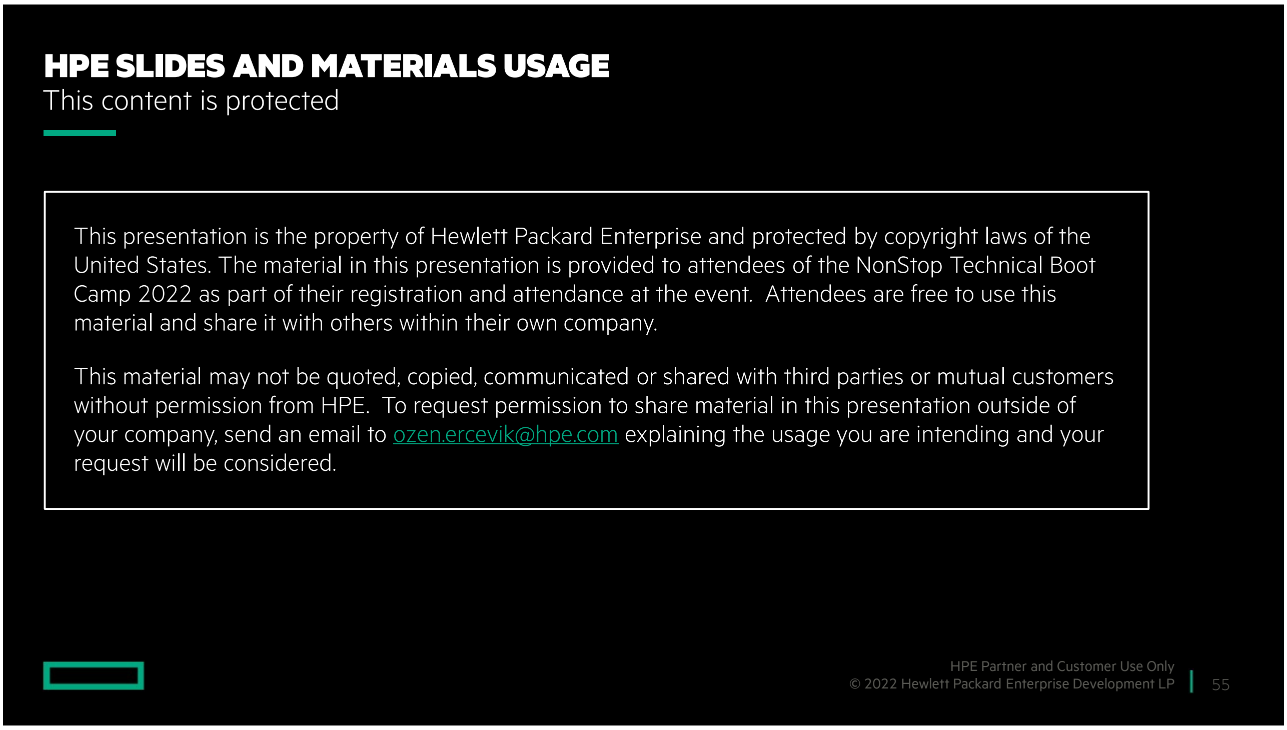
53



THANK YOU

Do you have questions? Are you interested in seeing/testing how to apply any of these on NonStop?
The ATC can help. Reach out to your account team to engage us!
Or contact me directly at franz.koenig@hpe.com

54



HPE SLIDES AND MATERIALS USAGE

This content is protected